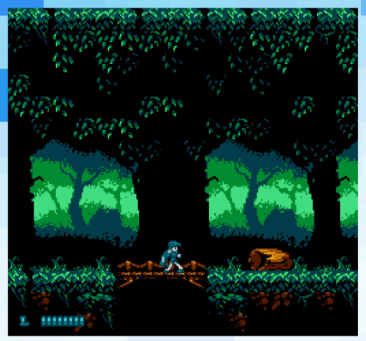
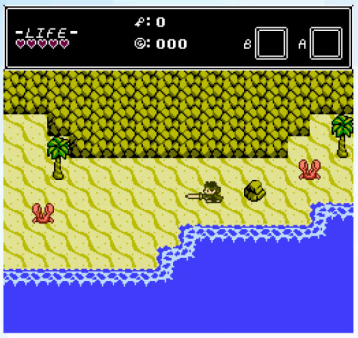
RetroMagazine

semplicemente retro

Numero 20 - Anno 4 - Gennaio 2020 - WWW.RETROMAGAZINE.NET - Pubblicazione gratuita

Crea un gioco NES col tuo PC!





8 bit!

The Dump Club 64

commodore 64

Il dumping da disco

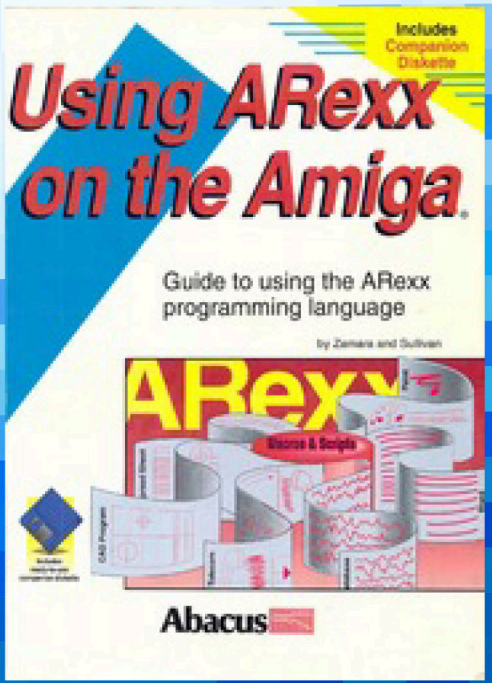
SKIFF - un gioco per Amstrad CPC in Locomotive Basic

Introduzione ad ARexx prima parte

NEW SUPER MARIO LAND (unofficial)

Dal Gameboy al SuperNES uno splendido remake!

Elenco dei negozi fisici che trattano RetroGaming in Italia



Giappone 8^puntata: il salvadanaio di Targa!

Programmare da zero un emulatore: intervista a Valerio Lupi

Intervista a Davide Sattin "Assemblatore cross-platform 6502"

Siamo già al 20, ovvero come passa il tempo quando ci si diverte!

Eccoci qua. Siamo arrivati a pubblicare il numero 20 di RetroMagazine grazie soprattutto al sostegno e all'entusiasmo dimostrato dai lettori. Per tutta la redazione e per tutti coloro che nel tempo hanno contribuito con idee, articoli e con le loro esperienze dirette raccontate per iscritto, si tratta di una mèta tutt'altro che semplice da raggiungere. Nonostante la sua grande passione per retrocomputing e retrogaming, forse neppure il nostro principale "istigatore a delinquere" (vero, Francesco?) si sarebbe aspettato di arrivare a tagliare questo traguardo con un'inerzia tanto intensa e con un'energia che incessantemente proviene dall'ottima accoglienza che ad ogni uscita RM riscuote. Sì, perché la chiusura di ogni numero comporta moltissimo lavoro editoriale e grafico, senza dimenticare tutte le attività che mirano ad aumentare la visibilità e la circolazione della rivista fra i potenziali target. E senza un riscontro positivo da parte di tanti retro-fan, i nostri sforzi verrebbero meno. Ma per fortuna i numeri dei download (mentre scriviamo abbiamo abbondantemente superato i 50mila) e le statistiche di diffusione forniscono continuamente quel carburante che fa marciare il lavoro della redazione e dei collaboratori in un clima di allegria e di divertimento.

Il numero 20 esce nel 2020, a distanza di circa 40 anni – sembra ieri – dalle origini del fenomeno di informatizzazione di massa reso possibile dall'avvento e dalla commercializzazione di console e home computer. E reca con sé quei contenuti e quelle rubriche che sin dal primo numero, secondo le nostre intenzioni, dovevano rifarsi alla nutrita tradizione editoriale (non solo italiana) delle riviste per computer, che da ragazzi tutti noi appassionati correvamo ad acquistare in edicola. Contenuti e rubriche che, visto il momento di grande successo che il fenomeno del retrocomputing sta vivendo da qualche anno, cercano di stimolare i lettori – speriamo come un tempo – a informarsi, a (re)imparare linguaggi e tecniche di programmazione, a scoprire le novità e a "sporcarsi le mani" investendo tempo libero in progetti grandi e piccoli a 8 e 16 bit!

Rileggendo oggi le riviste di un tempo (che, ricordiamolo, allora rappresentavano forse l'unico vero veicolo d'informazione autorevole), ci si stupisce a volte per l'ingenuità di alcuni articoli o al contrario per l'elevato livello di profondità e di esaustività raggiunte su argomenti hardware o software. Noi di RM cerchiamo da sempre di strizzare l'occhio a quella tradizione proponendo una rivista "generalista" (sono pochi ormai i marchi o i modelli che nel corso del tempo non abbiamo trattato), perché anche il lettore tipo di RM lo è. Fra gli appassionati sono ormai la maggioranza coloro che posseggono, conoscono approfonditamente e utilizzano attivamente più macchine. Difficilmente anche il più agguerrito "monoteista" di oggi – e ce ne sono, per carità – sfoglierà un numero di RM senza trovare uno spunto di riflessione, un argomento valido o una recensione che non attragga la sua attenzione.

Certo, si può e si deve migliorare. Per esempio aumentando il livello d'interattività con chi ci legge (scriveteci!) o proponendo corsi a puntate e rubriche fisse, avviando la discussione su temi d'attualità o, ancora, lanciando nuove sfide ai lettori basate sui giochi o sulla programmazione. Insomma, non ci fermiamo al numero 20, sebbene sia già un buon traguardo, ma proseguiremo con la nostra iniziativa editoriale, almeno fintanto che il nostro obbiettivo principale sarà raggiunto: appagare la nostra passione, condividerla con un po' di nostalgia con tutti coloro che vorranno partecipare e, in ultima analisi, divertirci!

David La Monaca/Cercamon
Scrivete a: redazione.retromagazine@gmail.com

SOMMARIO

Hanno collaborato alla stesura di questo numero di RetroMagazine

- Marco Pistorio
- Francesco Fiorentini
- Gianluca Girelli
- Alberto Apostolo
- The Dump Club 64 Team
- David La Monaca (Cercamon)
- Giorgio Balestrieri
- Starfox Mulder
- Carlo Nithaiah Del Mar Pirazzini
- Daniele Brahimi
- Querino Ialongo
- Michele Ugolini
- Federico "Arzak1" Gori
- **Copertina a cura di Flavio Soldani**

SKIFF - un gioco per Amstrad CPC in Locomotive Basic

di Francesco Fiorentini feat. RPI - RetroProgramming Italia

Quando si dice il caso... Ma e' realmente il caso oppure sono una concatenazione di scelte fatte precedentemente a guidarci verso una destinazione diversa da quella intrapresa in partenza ma, almeno in questo caso, infinitamente piu' soddisfacente? Cerchiamo di scoprirlo insieme...

Ci sono macchine che non ho avuto la fortuna di possedere o utilizzare quando ero ragazzo (E vorrei vedere, mica potevi averle tutte... NdR), ma adesso, grazie alla maggiore disponibilita' economica e soprattutto alla flessibilita' e duttilita' degli emulatori, ho deciso di riscoprire piano piano. Una di queste e' l'Amstrad CPC, di cui ammiravo in modo particolare la grafica ed i colori brillanti. Questa macchina, come ho avuto modo di scrivere nel numero 9, e' inoltre equipaggiata di un Basic di tutto rispetto, il **Locomotive Basic**.

Il Locomotive Basic e' un Basic decisamente avanzato, forse uno dei piu' completi e veloci dell'epoca, con una lista di comandi e funzioni da far letteralmente impallidire il Basic V2 contenuto nel Commodore 64... Ebbene mi ero riproposto da tempo di metterci le mani e provare a farci qualcosa di interessante per poi scriverne un articolo, ma come tutti ben sappiamo, il tempo e' la cosa piu' difficile da trovare ultimamente. Cosi, negli ultimi giorni dell'anno scorso, approfittando di qualche giorno di ferie e del fatto che la bimba fosse spupazzata dai nonni, decido di dare un'occhiata un po' piu' da vicino al Locomotive Basic coadiuvato da un interessante libro contenente listati di giochi in Locomotive Basic. Il libro si intitola '**The Amstrad Program Book**' di Peter Goode, edito da Phoenix Publishing Associates nel 1984.

Tra tutti i giochi riportati, vengo subito attratto da uno inititolato **Asteroid Storm**. Una prima occhiata al listato e la sua esigua lunghezza mi fanno subito propendere per questo gioco. L'idea iniziale per il mio articolo di RM era di partire da un giochino e, aggiungendo qualche elemento, farne un tutorial ad uso e consumo di chi, come me, volesse velocemente imparare i primi rudimenti del Locomotive Basic.

Il gioco originale riguarda un'astronave che deve affrontare un campo asteroidale infinito ed evitare gli asteroidi sino a quando morte non sopravvenga (...morte non vi separi... NdR) o il giocatoree non si stanchi della ripetitivita' del gioco.

Il concept quindi e' piuttosto banale, ma introduce anche diversi spunti interessanti. Proviamo ad analizzarli insieme:
- gioco a caratteri: e' un gioco che utilizza i caratteri standard dell'Amstrad CPC, nessun carattere ridefinito.
- scorrimento verticale: lo scrolling verticale e' realizzato semplicemente con un a capo nella riga 25, l'ultima riga dello schermo, simulando cosi' il movimento degli asteroidi verso l'astronave
- il movimento dell'astronave: l'astronave si muove solo orizzontalmente nella prima riga utile dello schermo. Il movimento e' controllato dalle frecce destra e sinistra.
- gestione delle collisioni: le collisioni sono controllate da una routine in Assembly che ritorna il valore del carattere contro il quale l'astronave si scontra.

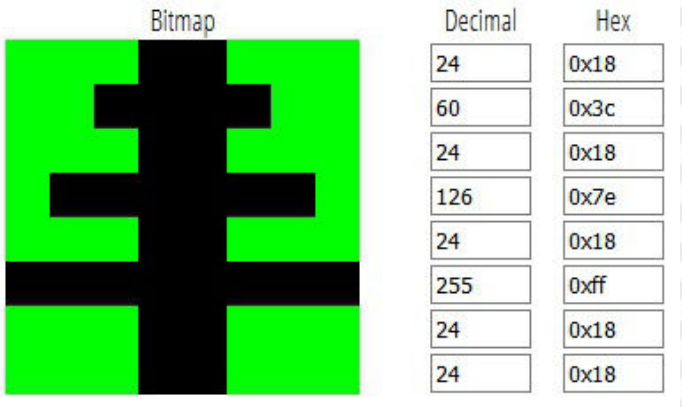
Come anticipato, l'idea iniziale era quella di aggiungere qualche elemento al gioco, magari ridefinendo i caratteri dell'astronave e degli asteroidi ed inserendo oggetti da collezionare per aumentare il punteggio e rendere il gioco un po' piu' difficile e meno noioso... Gia', questa era l'idea iniziale, ma poi guarda caso leggo su RPI il post di Davide Fichera sul contest del mese di gennaio riguardante il gioco Skiing rilasciato per Atari 2600 nel 1980 e mi viene un'idea...

Guardo il video allegato al contest e immediatamente mi accorgo delle similitudini con il giochino che stavo manipolando con il Locomotive Basic.
E se invece di un'astronave il protagonista del nostro gioco fosse uno sciatore?
E se al posto degli asteroidi ci fossero degli alberi? E lo sfondo? Posso farlo bianco per ricordare la parete innevata di una montagna?
Come vedete i cambiamenti iniziali sono piuttosto minimi, un paio di righe di codice aggiuntive ed una prima versione giocabile del nostro Skiing in Locomotive Basic e' praticamente pronta.

Ridefinire un carattere in Locomotive Basic e' veramente semplice, basta utilizzare il comando SYMBOL seguito dal numero del carattere e da una classica mappa 8x8; notate che soltanto il primo valore e' obbligatorio, eventuali linee mancanti verranno trattate come vuote.

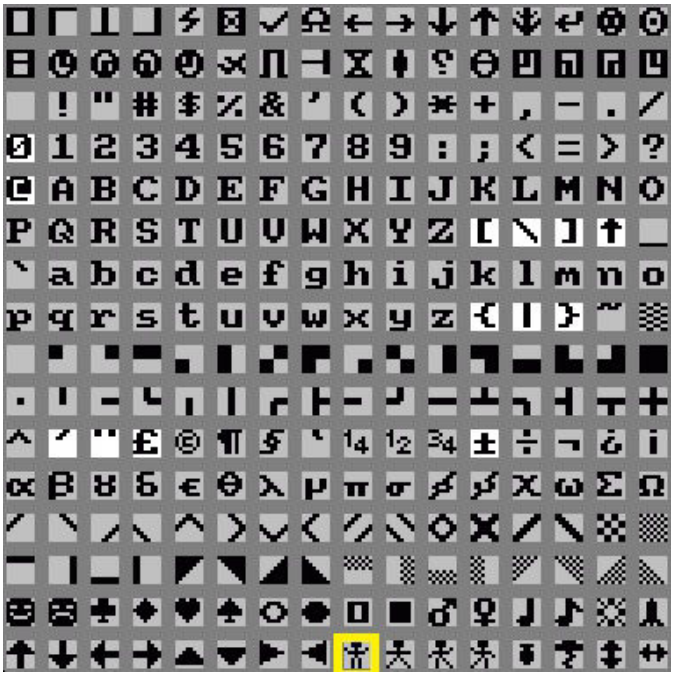
SYMBOL n,i1[,i2,i3,i4,i5,i6,i7,i8]

```
610 REM DISEGNA UN ALBERO RIDEFINENDO IL CHAR(255)
615 SYMBOL 255,24,60,24,126,24,255,24,24
```



Potete utilizzare questo tool online:
http://xlr8.at/8x8hexbin/

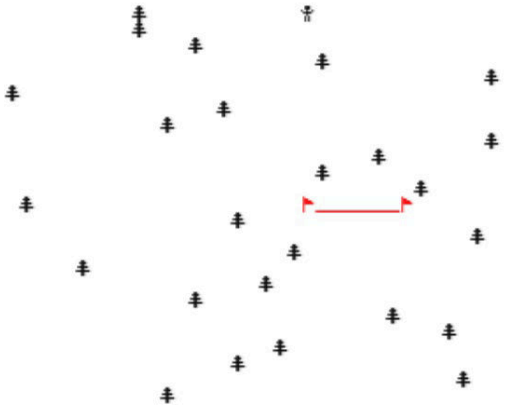
Per fortuna il character set standard dell'Amstrad CPC e' piuttosto ricco di suo, cosi' per lo sciatore ho potuto riutilizzare il carattere standard 248.



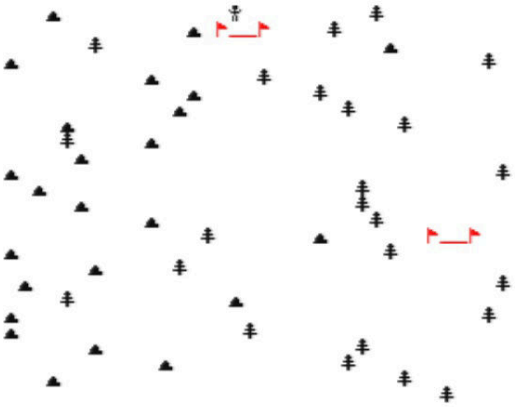
Effettivamente, con un po' d'immaginazione, ha una vaga somiglianza ad uno sciatore accucciato sgli sci. Non dovevo far altro che aggiungere le bandierine ed una versione base del gioco era pressoche' pronta... E questa e' stata la prima versione del gioco di cui ho pubblicato un veloce video su RPI ricevendo commenti che mi hanno spinto a continuare un po' piu' seriamente ad ampliare il giochino.

La versione iniziale del gioco era carina a vedersi, ma decisamente noiosa. Un gioco infinto senza nessun cambio di difficoltà e senza nessun scopo preciso.
Decido quindi di aggiungere qualche elemento per rendere il gioco almeno un po' più interessante.

La prima cosa che ogni gioco che si rispetti deve avere è un nome. Dopo un breve conciliabolo con me stesso partorisco il nome SKIFF... SKI di Francesco Fiorentini ...e poi perchè il gioco fa schi... Vabbè ci siamo capiti dai.
Sull'onda dell'entusiamo decido di creare un menù iniziale ed aggiungere nello stesso ben 5 livelli di difficoltà crescente. Inoltre decido di dare anche un senso al gioco, inserendo una Finish Line e la necessità di passare attraverso le porte per completare i livelli senza averne omessa nessuna. Finalmente cominciamo a ragionare.



Come potete facilmente intuire dal listato allegato i 5 livelli influiscono pesantemente nella difficoltà del gioco. La distanza verticale delle porte varia infatti da un massimo di 15 caratteri (livello rookie ed easy) ad un minimo di 13 (livello impossible). Anche la larghezza delle porte e la loro distribuzione casuale è influenzata dalla diffcoltà del livello. Inoltre nei livelli medium, hard ed impossible, lo sciatore non dovrà evitare soltanto gli alberi ma anche delle rocce che compariranno casualmente nel suo percorso. Onestamente, completare il livello impossible centrando tutte le porte è veramente sfidante.



Ovviamente tutti i parametri del gioco sono gestiti da delle variabili il che rende il gioco facilmente modificabile per renderlo ancora più difficile o più semplice, dipende dai gusti.

Qui potete trovare il listato completo del gioco con tanto di commenti:

```
10 REM SKIFF by FRANCESCO FIORENTINI - Gennaio
2020
15 GOSUB 600:REM DISEGNA BANDIERA E ALBERO
20 GOSUB 410:REM CODICE MACCHINA COLLISIONI
```

```
24 REM DEFINISCE OGGETTI
25 SHIP$=" "+CHR$(248)+" "
26 TREE$=CHR$(255)
27 STONE$=CHR$(253)

30 REM INIZIALIZZA I PARAMETRI DEL GIOCO
35 GOSUB 2000:REM MENU INIZIALE
36 GOSUB 2400:REM PARAMETRI LIVELLO

38 REM INIZIALIZZA I VALORI DEL GIOCO
40 SCORE=0
41 DSPFLAG=0
45 DOORS=0
50 FLINE=3050
55 MODE 1
60 BORDER 26
70 SHIPX=18
80 YY=1
100 PAPER 0
110 INK 0,26
115 INK 1,0
120 CLS

130 REM MAIN LOOP
140 A=INT(RND(1)*38+1)
150 PEN 1
155 LOCATE A,25:PRINT TREE$;
159 REM ROCCE SOLO PER LIVELLO M-H-I
160 IF LVL$="R" OR LVL$="r" THEN GOTO 180
162 IF LVL$="E" OR LVL$="e" THEN GOTO 180
170 B=INT(RND(1)*(38-A)+1)
172 LOCATE B,25:PRINT STONE$
180 IF DSPFLAG=VDOORS THEN GOSUB 1000:REM
DISTANZA PORTE IN VERTICALE
186 DSPFLAG=DSPFLAG+1
187 LOCATE 40,25:PRINT" "
190 REM MOVE SKI
200 SCORE=SCORE+10
210 IF INKEY(8)=0 AND SHIPX>1 THEN SHIPX=SHIPX-1
220 IF INKEY(1)=0 AND SHIPX<38 THEN SHIPX=SHIPX+1
230 XX=SHIPX+1
240 GOSUB 550
250 IF DD=255 THEN GOTO 300
252 IF DD=253 THEN GOTO 350
255 IF DD=95 THEN DOORS=DOORS+1
260 IF SCORE = FLINE THEN LOCATE 1,25:PRINT "
= = = = = = F I N I S H = = = = = = ="
265 IF SCORE = FLINE+250 THEN GOSUB 1600
280 LOCATE SHIPX,1
281 PEN 1
282 PRINT SHIP$
290 GOTO 130
300 REM UCCISO DA UN ALBERO
302 PAPER 1
304 PEN 3
305 PRINT CHR$(7)
306 LOCATE 14,6:PRINT"Oh Cavolo..."
308 FOR Q=1 TO 1000: NEXT Q
310 LOCATE 7, 8:PRINT"Sei finito contro un
albero..."
312 LOCATE 5, 9:PRINT"Come sciatore sei proprio
scarso!"
314 FOR q=1 TO 1000: NEXT q
316 LOCATE 7, 12: PRINT "Hai percorso: " ;SCORE;"
metri"
318 LOCATE 7, 13: PRINT "Passando solo: " ;DOORS;"
porte!"
320 IF INKEY$=" " THEN GOTO 320
322 IF INKEY$<>" " THEN GOTO 322
324 GOTO 30

350 REM UCCISO DA UNA ROCCIA
352 PAPER 1
354 PEN 3
355 PRINT CHR$(7)
356 LOCATE 14,6:PRINT"Oh Cavolo..."
358 FOR Q=1 TO 1000: NEXT Q
360 LOCATE 7, 8:PRINT"Sei finito contro una
roccia..."
362 LOCATE 5, 9:PRINT"Come sciatore sei proprio
scarso!"
364 FOR q=1 TO 1000: NEXT q
366 LOCATE 7, 12: PRINT "Hai percorso: " ;SCORE;"
metri"
368 LOCATE 7, 13: PRINT "Passando solo: " ;DOORS;"
porte!"
370 IF INKEY$=" " THEN GOTO 370
372 IF INKEY$<>" " THEN GOTO 322
374 GOTO 30

410 REM PUT MACHINE CODE JUST ABOVE RE-ADJUSTED
HIMEM
411 REM CODICE MACCHINA CHE GESTISCE LA COLLISIONE
DEI CARATTERI
420 MEMORY FRE(0)-&80
430 MC=HIMEM+1
440 TA=HIMEM+&7F
450 TH=INT(TA/256)
460 TL=TA-256*TH
470 POKE MC,&CD
480 POKE MC+1,&60
490 POKE MC+2,&BB
500 POKE MC+3,&32
510 POKE MC+4,TL
520 POKE MC+5,TH
530 POKE MC+6,&C9
540 RETURN

549 REM CONTROLLO COLLISIONI
550 LOCATE XX,YY
560 CALL MC
570 DD=PEEK(TA)
580 RETURN

600 REM DISEGNA LA BANDIERINA RIDEFINENDO IL
CHAR(254)
605 SYMBOL 254,64,112,124,126,64,64,64,64
610 REM DISEGNA UN ALBERO RIDEFINENDO IL
CHAR(255)
615 SYMBOL 255,24,60,24,126,24,255,24,24
620 REM DISEGNA UNA ROCCIA RIDEFINENDO IL
CHAR(253)
625 SYMBOL 253,0,0,0,24,60,124,127,255
630 RETURN
```

```
1000 REM DISEGNA LE BANDIERINE ED IL CARATTERE
DI CONTROLLO
1001 REM PER CONTARE LE PORTE
1005 C=INT(RND(1)*(LSUP-LINF)+LINF)
1010 PEN 3
1020 LOCATE C,25:PRINT GAP$
1040 DSPFLAG=0
1050 RETURN

1600 REM LIVELLO COMPLETATO
1610 PAPER 1
1620 PEN 3
1630 LOCATE 15,6:PRINT"BRAVO!!!!"
1640 FOR Q=1 TO 1000: NEXT Q
1650 LOCATE 7, 8:PRINT"Hai completato il
livello..."
1660 FOR q=1TO 1000: NEXT q
1670 LOCATE 8, 11: PRINT "Hai percorso :
" ;SCORE;" metri"
1675 LOCATE 9, 13: PRINT "Hai passato : " ;DOORS;"
porte"
1677 MISSED = INT(FLINE/(VDOORS*10)) - DOORS
1678 LOCATE 9, 14: PRINT "Hai mancato :
" ;MISSED;" porte"
1679 LOCATE 7, 16:PRINT"Un'altra sfida? (premi
SPAZIO)"
1680 IF INKEY$=" " THEN GOTO 1680
1690 IF INKEY$<>" " THEN GOTO 1690
1700 GOTO 30

2000 REM MENU INIZIALE
2005 CLS
2010 LOCATE 10,2:PRINT
CHR$(143);CHR$(143);CHR$(143);"  ";CHR$(143);"
";CHR$(143);"  ";CHR$(143);"
";CHR$(143);CHR$(143);CHR$(143);"
";CHR$(143);CHR$(143);CHR$(143)
2015 LOCATE 10,3:PRINT CHR$(143);"
";CHR$(143);" ";CHR$(143);"   ";CHR$(143);"
"CHR$(143);"    "CHR$(143)
2020 LOCATE 10,4:PRINT " ";CHR$(143);"
";CHR$(143);CHR$(143);"    ";CHR$(143);"
";CHR$(143);CHR$(143);"   ";CHR$(143);CHR$(143)
2025 LOCATE 10,5:PRINT "  ";CHR$(143);"
";CHR$(143);" ";CHR$(143);"   ";CHR$(143);"
"CHR$(143);"    "CHR$(143)
2030 LOCATE 10,6:PRINT
CHR$(143);CHR$(143);CHR$(143);"  ";CHR$(143);"
";CHR$(143);"   ";CHR$(143);"   "CHR$(143);"
"CHR$(143)
2040 LOCATE 15,8:PRINT "di Francesco Fiorentini"
2050 LOCATE 10,12:PRINT "Livello:"
2051 LOCATE 10,13:PRINT "(R)ookie"
2052 LOCATE 10,14:PRINT "(E)asy"
2053 LOCATE 10,15:PRINT "(M)edium"
2054 LOCATE 10,16:PRINT "(H)ard"
2055 LOCATE 10,17:PRINT "(I)mpossible"
2056 LOCATE 1,20:PRINT "Istruzioni:"
2058 LOCATE 1,21:PRINT "Passa tra le porte
evita alberi e rocce"
2057 LOCATE 1,22:PRINT "Usa le frecce
";CHR$(242);CHR$(243);" per spostarti"
2100 LVL$= INKEY$
2102 IF LVL$="R" OR LVL$="r" THEN GOTO 2120
2104 IF LVL$="E" OR LVL$="e" THEN GOTO 2120
2106 IF LVL$="M" OR LVL$="m" THEN GOTO 2120
2108 IF LVL$="H" OR LVL$="h" THEN GOTO 2120
2110 IF LVL$="I" OR LVL$="i" THEN GOTO 2120
2115 GOTO 2100
2120 RETURN

2400 REM INIZIALIZZA VALORI PER LIVELLO
2401 IF LVL$="R" OR LVL$="r" THEN GOTO 2500
2402 IF LVL$="E" OR LVL$="e" THEN GOTO 2600
2403 IF LVL$="M" OR LVL$="m" THEN GOTO 2700
2404 IF LVL$="H" OR LVL$="h" THEN GOTO 2800
2405 IF LVL$="I" OR LVL$="i" THEN GOTO 2900
2500 REM LIVELLO ROOKIE
2502 GAP$=CHR$(254)+"______"+CHR$(254)
2503 VDOORS=15
2504 LINF=10:REM LIMITE INF DISTANZA H PORTE
2505 LSUP=25:REM LIMITE SUP DISTANZA H PORTE
2599 RETURN
2600 REM LIVELLO EASY
2602 GAP$=CHR$(254)+"_____"+CHR$(254)
2603 VDOORS=15
2604 LINF=9
2605 LSUP=27
2699 RETURN
2700 REM LIVELLO MEDIUM
2702 GAP$=CHR$(254)+"____"+CHR$(254)
2703 VDOORS=14
2704 LINF=8
2705 LSUP=29
2799 RETURN
2800 REM LIVELLO HARD
2802 GAP$=CHR$(254)+"___"+CHR$(254)
2803 VDOORS=14
2804 LINF=8
2805 LSUP=31
2899 RETURN
2900 REM LIVELLO IMPOSSIBLE
2902 GAP$=CHR$(254)+"__"+CHR$(254)
2903 VDOORS=13
2904 LINF=7
2905 LSUP=33
2999 RETURN
```

Pensavate che fosse finita qui? Certo che no! Altrimenti il caso dell'inizio dell'articolo sarebbe stato poco efficace. :-)

Mentre stavo lavorando alla versione finale del mio giochino mi vedo taggato sempre sul gruppo RPI da qualcuno che mi invita a provare a fare una versione 10-liner dello stesso. Non mi ero mai cimentato in un ten-liner prima di allora, cosicche' l'idea mi ha stuzzicato non poco ed ho quindi deciso di cogliere la sfida e buttarmi nell'impresa.

Ci sono riuscito? Si', sono riuscito a generare una versione funzionante del gioco in solo 10 righe. L'ho anche postata nel gruppo RPI, quindi potete trovarla li', oppure, se preferite avere qualche informazione in piu' su come sia stato realizzato quel un ten-liner, vi do appuntamento al prossimo numero di RetroMagazine.

Introduzione ad ARexx - prima parte

di Gianluca Girelli

Questo articolo è comparso la prima volta sulle pagine di Bitplane nel Novembre del 2011.

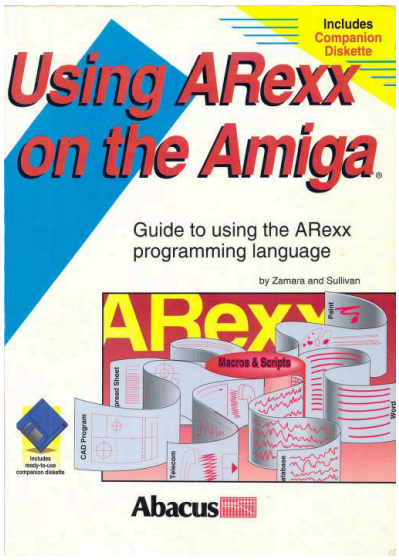
ARexx e' un linguaggio di "scripting" nato come diretta implementazione su Amiga del linguaggio REXX. Include una serie di caratteristiche peculiari per Amiga che ne estendono le funzionalita' al di la' del REXX standard.
Come la maggior parte delle implementazione di REXX, ARexx e' un linguaggio interpretato. I programmi scritti per ARexx sono chiamati "script" o "macro" e diversi software offrono tuttora la possibilita' di eseguire script ARexx dalla loro interfaccia principale (es: Personal Paint) o nel loro codice (Es: Hollywood). ARexx puo' infatti facilmente comunicare con tutti i software di terze parti che implementino una "porta" per lo scambio dei dati attraverso un sistema chiamato "IPC" (Inter Process Communication).

1. Un po' di storia

La prima comparsa del linguaggio di programmazione ARexx risale al 1987, anno in cui William S. Hawes ne effettuo' il "port" per Amiga dal mondo IBM.
ARexx e' basato sul linguaggio REXX (REstructured eXtended eXecutor) descritto da Mike Cowlishaw nel suo libro "The REXX Language: A Practical Approach to Programming". ARexx e' stato incluso da Commodore in AmigaOS2.0 nel 1990. Questa ultima versione di ARexx segue il linguaggio REXX "ufficiale" molto fedelmente, tanto che Hawes fu successivamente incluso nel gruppo di lavoro che pose le basi "standard ANSI" di REXX.

ARexx e' scritto in Assembly per M68000, e quindi le sue prestazioni non sono ottimizzate per lavorare alla massima velocita' permessa dalle nuove CPU PPC, per le quali una nuova versione del linguaggio non e' mai stata sviluppata. Oggigiorno William Hawes non e' piu' coinvolto nello sviluppo del linguaggio (probabilmente a causa di trascorsi legali avuti in passato con Commodore) e nessuna ditta che attualmente lavora in campo Amiga ne sta finanziando nuove versioni.
Non e' neppure possibile estendere il linguaggio a cura degli utenti, come ad esempio succede in ambiente GNU/Linux, poiche' i sorgenti non sono disponibili.
Il futuro di questo tipo di linguaggio risiede oggi quindi probabilmente in Python (gia' presente su Amiga) ma, in attesa di un successore ufficiale, ARexx viene ancora usato ed ha ancora tanto da dire. E' emblematico il fatto che, come riportato sul sito ufficiale di AmigaPython (vedi bibliografia), questo linguaggio supporti ARexx per poterlo poi sostituire direttamente in fase di scripting. Sempre secondo il sito, Python e' inferiore ad Arexx in quanto l'occupazione di memoria richiesta e' di ben 10 volte superiore anche se, occorre dirlo, questo fattore non e' piu' un problema sui computer odierni.

2. Perche' ARexx



ARexx puo' inviare comandi a funzioni ad applicazioni diverse per mezzo di un unico script, offrendo cosi' l'opportunita' di combinare funzioni appartenenti a diversi programmi.
Ad esempio, uno script ARexx puo' estrarre dati da un database, inserirli in un foglio di calcolo per elaborarli, prelevare da esso le tabelle ed i grafici risultanti ed impaginarli con un word processor, infine spedire per e-mail il documento risultante ai vostri contatti in rubrica. Se tutto questo vi sembra familiare non vi sbagliate: VBA (Visual Basic for Applications) di Microsoft lo fa da anni. Ancora una volta, pero', possiamo dire: "Amiga lo faceva gia' prima!".

Pur essendo un software a sorgente chiuso, il linguaggio REXX, si e' consolidato ed espanso enormemente grazie ad una comunita' di utenti estremamente attiva e competente. Ecco perche' anche ARexx continua a tuttoggi a non sentire il peso degli anni. Forgiato su esigenze di lavoro/utilizzo reali della "community", incorpora un set di istruzioni estremamente potenti e flessibili in grado di operare con efficenza ed efficacia sulla formattazione dell'input/output (soprattutto nel campo delle stringhe) consentendo una estrema facilita' di lettura/trasmissione dati.

E' anche a causa di questo tipo di capacita' che Commodore decise, a partire dalla Release 2 di AmigaDOS, di inserire ARexx nel sistema operativo. Cio' escluse di riflesso AmigaBASIC rivelatosi lento, affazzonato e, soprattutto, incapace di sfruttare al meglio le innate capacita' multitasking di Amiga.

Come tutte le implementazioni di REXX, ARexx usa una rappresentazione dei dati di tipo "typeless", cioe' non fa distinzioni tra numeri interi, reali (a virgola mobile), stringhe, caratteri, vettori etc. Tutti i dati vengono rappresentati internamente come stringhe di caratteri, semplificando la scrittura di espressioni ed algoritmi. Come spesso succede nei linguaggi dinamici, le variabili non necessitano di essere dichiarate prima di poterle usare, ma vengono create al loro primo utilizzo.

Il set di comandi di ARexx e' semplice, ma in aggiunta ai

comandi propri del linguaggio ci sono le funzioni incluse nella relativa libreria Amiga (rexxsyslib.library) presente in "System:Libs/". E' altresi' facile aggiungere altre librerie o funzioni poiche' gli script ARexx posso essere invocati come funzioni da altri script. Qualsiasi programma per Amiga che incorpori una "ARexx port" puo' condividere le sue funzioni con i programmi ARexx.

3. Un po' di pratica

Anziche' iniziare con il classico "Hello World!", proviamo un semplice programma che calcola l'eta' in giorni.

```
/* eta.rexx */
say 'scrivi eta'
pull eta
say 'hai circa' eta*365 'giorni'
```

Per quanto semplice, questo frammento di codice ci dice che:
- tutti i programmi ARexx devono iniziare con un commento (la sequenza /* */). Il contenuto effettivo del commento non ha importanza;
- le variabili (in questo caso "eta") non devono essere esplicitamente dichiarate in quanto allocate dinamicamente;
- la concatenazione delle stringhe in uscita (comandate da "say") viene effettuata dal carattere "spazio" (o blank) a differenza di altri linguaggi che usano operatori ben definiti.

Arexx e' un linguaggio "case insensitive". Maiuscolo o minuscolo non ha importanza. Vale la pena notare, pero', che il comando "pull" usato per leggere l'input e' la forma abbreviata di "parse upper pull". Ogni stringa alfanumerica letta con questo metodo sara' sempre restituita in lettere maiuscole, cioe' le stringhe in input "Amiga" o "AmIgA" (o qualsiasi altra combinazione) verranno sempre restituite da "say" come: "AMIGA".

Per utilizzare il programma basta aprire il vostro text editor preferito, digitare il codice e salvarlo (in modalita' testo) come nome_file.rexx (esempio: eta.rexx). A questo punto basta aprire la shell e digitare ">rx eta.rexx" (o piu' brevemente ">rx eta") ed il gioco e' fatto. Il sistema invochera' in automatico l'interprete di ARexx "RexxMast" ed eseguira' lo script.
Chiaramente, se la shell viene aperta in una directory diversa da quella che contiene il vostro codice, bisognera' digitare il percorso completo.

Passiamo ora al famigerato "Hello World!", che in ARexx e' semplicemente:

```
say "Hello World"! (da riga di comando)
```

oppure

```
/* eta.rexx */
say "Hello World!" (da script)
```

Come si vede, l'utilizzo dei doppi apici e' alternativo al singolo apice e torna utile in quelle occasioni in cui dobbiamo usare accenti o apostrofi all'interno di una stringa. Per capirci meglio, scrivere "eta' " risulta meno suscettibile di errori rispetto allo scrivere 'eta'''.

ARexx puo' aprire finestre indipendenti dalla shell in cui stiamo lavorando. Vediamo un esempio:

```
/* prova costrutti di base ARexx */
clear
  if~open('console','con:0/0/640/200/RexxWindow/
CLOSE','W')
     then exit 20
  call writeln'console','Salve a tutti. Questa e''
una prova di uso delle finestre.'
  call writeln'console','Premere INVIO per
continuare.'
  call readln'console'
  call close 'console'
end
```

In questo codice apriamo prima di tutto una finestra (parametro 'console') di 640x200 pixel, chiamata RexxWindow e munita di gadget di chiusura.
Se per qualche motivo la finestra non puo' essere aperta, l'esecuzione si interrompe restituendo un messaggio di errore.
'Console' rappresenta un file logico verso il quale viene rediretto l'input/output attraverso i comandi "writeln" e "readln", invocati dall'istruzione "call". La finestra viene poi chiusa (call close 'console') ed il programma terminato (end). Attenzione a non confondere "end", che solitament chiude un blocco di codice con l'istruzione "exit" che, se usata, termina immediatamente lo script da qualsiasi punto venga invocata.

Usiamo ora lo stesso tipo di sistema per lavorare sui file fisici. Il codice seguente crea un semplice file di testo nel disco logico "Ram" (prefissi.dat) contenente dei prefissi telefonici:

```
/* scrive prefissi telefonici in prefissi.dat */
datafile='datafile'
filename='ram:prefissi.dat'
if open(datafile, filename, 'w') then do
   call writeln(datafile, '02 Milano')
   call writeln(datafile, '06 Roma')
   call writeln(datafile, '045 Verona')
   call writeln(datafile, '081 Napoli')
   call close(datafile)
   end
else
   say "Unable to open " filename
```

Essendo un file di testo, esso puo' essere letto semplicemente digitando da shell il comando

Shell> type ram:prefissi.dat

Se invece vogliamo leggere il file dall'interno di un altro script basta scrivere:

```
/* legge i dati di prefissi.dat */
tfile='typefile'
name ='ram:prefissi.dat'
if open(tfile, name, 'r') then do
  do while ~eof(tfile)
     say readln(tfile)
     end
  call close(tfile)
  end
else
   say 'Unable to open file' name '.'
```

Questo script, che non differisce dal precedente comando "type", puo' pero' essere modificato a piacere per caricare i dati letti dal file in variabili (ad esempio un array di testo) per poterli poi elaborare a seconda delle esigenze.

4. Conclusioni

Come si e' visto in questa breve introduzione Arexx, nonostante il peso degli anni, risulta tuttora un linguaggio versatile e moderno in grado di competere con prodotti piu' blasonati. Al momento in cui questo articolo fu originariamente scritto, per quanto riguarda il mondo Amiga, ARexx era (ed e') presente su ogni versione del sistema operativo a partire dalla 2.0; implementazioni di REXX conformi allo standard ANSI esistono comunque per tutte le piattaforme (Unix/Linux, Mac, Windows etc) e la piu' popolare tra queste era senza dubbio Regina_REXX, diventato l'ARexx di base di AROS. MorphOS, invece, avendo una libreria rexxsyslib.library per vari motivi non funzionante, non era in grado di utilizzare ARexx se non sostituendo la libreria nativa MorphOS con una della famiglia AmigaOS 3.x.
Il progetto Regina_REXX si poneva quindi come obiettivo anche quello di diventare il REXX standard per MorphOS, affrancandosi cosi' dalla dipendenza del sistema dalla library di AmigaOS 3.x.
La facilita' con cui si scrivono gli script in ARexx e' tale che non e' necessario essere un programmatore per poterli implementare.
Tuttavia la potenza di ARexx (e la sua capacita' di interfacciarsi con quasi ogni applicazione Amiga sia moderna che "legacy") rende questo linguaggio essenziale e utilizzabile proficuamente anche dal programmatore piu' esperto, soprattutto per quei lavori di routine per i quali la velocita' di elaborazione non e' un fattore importante.

5. Next issue preview

Nel prossimo tutorial entreremo piu' nel dettaglio dell'uso di stringhe, matrici, istruzioni condizionali e procedure. Alla fine del tutorial avremo posto le basi per la costruzione di un semplice motore di gioco per avventure testuali.

6. BIBLIOGRAFIA

Mike Cowlishaw "The REXX Language: A Practical Approach to Programming" (1985) Prentice Hall. ISBN 0-13-780651-5.
Chris Zamara, Nick Sullivan "Using Arexx on the Amiga" (1991) Abacus Software Inc. ISBN 1-55755-114-6.
AmigaOS 2.0 Manuale di sistema

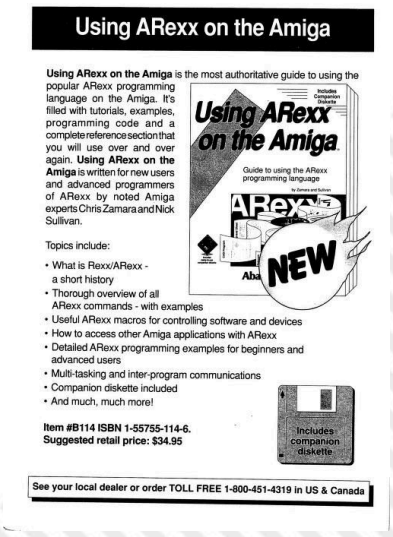
http://it.wikipedia.org/wiki/Linguaggio_di_scripting
http://it.wikipedia.org/wiki/Linguaggio_interpretato
http://en.wikipedia.org/wiki/ARexx
http://www.monkeyhouse.eclipse.co.uk/amiga/python/
http://regina-rexx.sourceforge.net/

Linguaggi di scripting
In informatica un linguaggio di scripting è un linguaggio di programmazione interpretato, destinato in genere a compiti di automazione del sistema operativo (batch) o delle applicazioni (macro), o a essere usato all'interno delle pagine web.
Gli script sono generalmente semplici programmi il cui scopo è l'interazione con altri programmi, molto più complessi, in cui avvengono le operazioni più significative. Gli script si distinguono dai programmi con cui interagiscono, solitamente implementati in un linguaggio differente e non interpretato. Inoltre, spesso gli script sono creati o modificati dall'utente finale.

Linguaggi interpretati
I linguaggi che prevedono l'esecuzione dei listati di codice per mezzo di un interprete sono detti linguaggi interpretati.
Un interprete ha lo scopo di eseguire un programma in un linguaggio di alto livello, senza la previa compilazione dello stesso.
Esso, infatti, ha il compito di eseguire le istruzioni nel linguaggio usato, traducendole di volta in volta in istruzioni in linguaggio macchina.

Programmi "legacy"
In informatica, il termine "legacy" si riferisce a tutti quei dati ed applicazioni che sono stati ereditati da linguaggi, piattaforme e tecniche di programmazione sviluppate prima della tecnologia attuale. La maggior parte delle aziende che utilizzano estensivamente i computer hanno applicazioni e database di questo tipo che sostengono aree critiche del servizio. Tipicamente la sfida e' tenere il codice "legacy" in uso mentre si sta migrando verso codice piu' nuovo, piu' efficiente e che utilizzi le nuove tecnologie e le aumentate capacita' dei programmatori.

From the ARexx manual
"ARexx was developed on an Amiga 1000 computer with 512k bytes of memory and two floppy disk drives. The language prototype was developed in C using Lattice C, and the production version was written in assembly-language using the Metacomco assembler. The documentation was created using the TxEd editor, and was set in TeX using AmigaTeX. This is a 100% Amiga product.



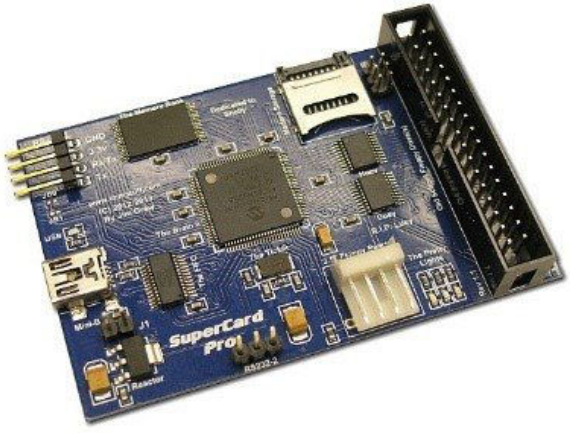
Il dumping da disco

DUMP CLUB 64

di The Dump Club 64 Team

Guardiamo in faccia la realtà: Il dumping dei supporti magnetici (in particolare del Commodore 64) è la quintessenza del retrocomputing! Una tappa obbligata per chiunque si consideri un vero appassionato e che può regalare quella stessa sensazione che devono aver provato i primi avventurieri, all'inizio del secolo scorso, ad entrare nelle sale buie e inviolate delle misteriose piramidi d'Egitto. Immaginate infatti di riuscire a recuperare un disco illeggibile, scovato in qualche cantina umida e polverosa e di adoperarvi con cura e dedizione fino a riuscire a renderlo leggibile e infine a trasferirlo su file per poi caricare la directory e ritrovare dei piccoli tesori dimenticati che prima del vostro lavoro erano irrimediabilmente compromessi se non destinati a scomparire per sempre. Non fraintendiamoci però, l'attività di dumping è un vaso di Pandora dal quale possono uscire centinaia di insidie ed è una strada irta di ostacoli e costellata di imprevisti che possono seriamente mettere a rischio la stabilità dei vostri nervi ma le emozioni e le ricompense che è in grado di offrirvi ripagheranno comunque ogni vostro sforzo. Tuffiamoci allora senza ulteriori indugi nel merito della questione: per non mettere troppa carne al fuoco tratteremo più avanti, sulle pagine di RetroMagazine, l'aspetto tecnico del dumping per lasciare questa volta maggiore spazio all'aspetto prettamente pratico del trasferimento da disco e poterci così immediatamente catapultare nel fantastico mondo del dumping.

Attualmente esistono diversi metodi molto efficaci per il trasferimento dei dischetti floppy del Commodore 64. Di seguito un breve elenco dei più popolari:



SuperCardPro: Il SuperCardPro è un dispositivo harware di lettura "Flux" e come altri lettori simili gestisce la lettura di dati di flusso non elaborati (RAW). È in grado di effettuare copie a livello di settore e può convertire i dati RAW in formati che possono essere usati sugli emulatori.
Link: http://www.cbmstuff.com/proddetail.php?prod=scp

KryoFlux: Il KryoFlux è un dispositivo hardware che legge il disco floppy non come un insieme di dati ma come flusso magnetico effettivo dei dischi stessi. Quello che si ottiene, nella stragrande maggioranza dei casi, è un'immagine del disco che include qualsiasi protezione dalla copia, schemi di scrittura insoliti ed altri stratagemmi usati per mischiare le carte.
Link: https://www.kryoflux.com/

ZoomFloppy / XUM1541: Lo ZoomFloppy è una soluzione molto pratica ed economica per il trasferimento dei dischetti e oltre alla modalità seriale supporta anche, via connessione parallela, un protocollo Burst Nibbler per copiare e trasferire le protezioni dei dischi. In più consente di collegare le stampanti Commodore al PC.
Link: http://www.go4retro.com/products/zoomfloppy/

In questo articolo parleremo esclusivamente di quest'ultimo hardware, essendo il più diffuso tra gli appassionati.

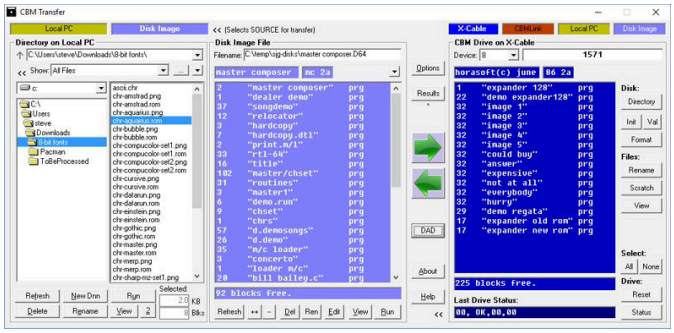
Lo **ZoomFloppy** consente il trasferimento dei dati e la creazione di immagini disco d64, d71, d81, g64, nib dall'unità disco 1541, 1541-II e compatibili, 1571 o 1581 al PC e viceversa, mediante la connessione con il cavo di connessione seriale IEC (lato ZoomFloppy) e microUSB (lato PC).
Utilizzando invece il cavo parallelo (lato ZoomFloppy) si possono trasferire con successo anche dischi protetti nella modalità **Nibble**. Con il 1571 la modalità Nibble è possible anche con il solo cavo di connessione IEC seriale.

Collegato con il cavo seriale lo ZoomFloppy leggerà il contenuto di un dischetto floppy CBM settore per settore creando un file .d64 attraverso il programma

d64copy del pacchetto software **OpenCBM** https://github.com/OpenCBM.

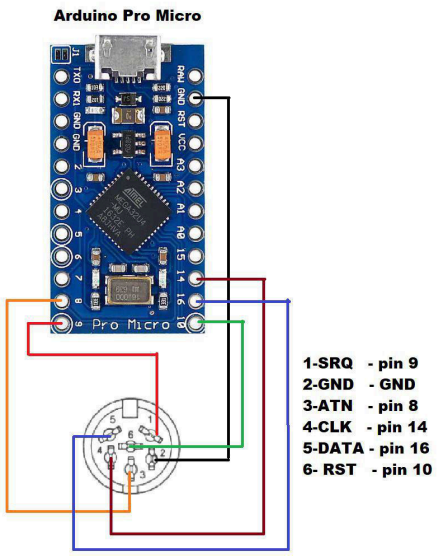
Attraverso la connessione parallela invece, come abbiamo accennato in precedenza, sarà possibile utilizzare la modalità **Nibble** che permetterà di copiare anche i floppy disk protetti. In questa modalità lo ZoomFloppy leggerà i dati non elaborati (raw bits) di ogni traccia senza alcuna interpretazione creando dei file .nib / .nbz (che poi potranno essere convertiti in .g64 per utilizzo con gli emulatori) attraverso il programma **Nibread** del pacchetto software **Nibtools** https://github.com/markusC64/nibtools.

Per chi non ama la linea di comando e preferisce utilizzare una interfaccia grafica è disponibile **CBMXfer** che offre una GUI molto intuitiva: http://www.6502.org/users/sjgray/software/cbmxfer/cbmxfer.html.

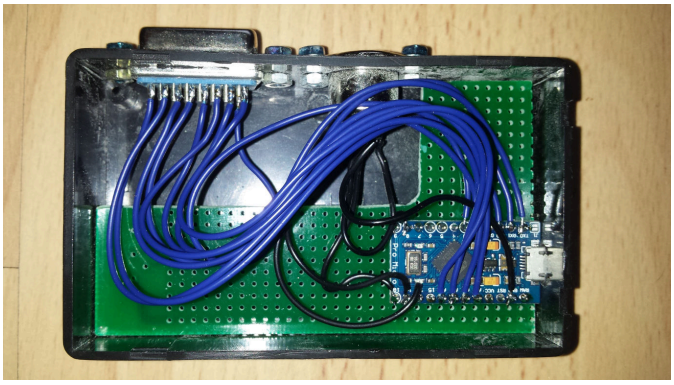


È opportuno dire che è preferibile utilizzare un 1571 perché, mentre con il 1541 si può abilitare la modalità nibble solo attraverso la connessione con il cavo parallelo, con il 1571 è sufficiente il cavo seriale. Inoltre il 1571 solitamente ha meno problemi di allineamento rispetto al 1541. Infatti quest'ultimo trova la traccia zero cercando indietro le tracce 35 o 40 dalla sua posizione corrente: questo spesso porta il meccanismo della testina a picchiare duramente l'arresto e a lungo andare tutto ciò ha un impatto decisamente negativo sull'allineamento dell'unità. Il 1571, invece, non mostra un simile comportamento consentendo di avere problemi di disallineamento molto più bassi.

XUM1541



Concludiamo il discorso segnalando una soluzione "fai da te" per realizzare una versione "a basso costo" del progetto originale dello ZoomFloppy. E' possibile infatti costruire un XUM1541 utilizzando una board Arduino ProMicro che può essere acquistata al costo di circa 6€ nel mercato europeo. Flashando il ProMicro con un opportuno firmware e collegandolo all'unità seguendo il giusto pinout avremo un affidabile ed economico dispositivo per la nostra attività di dumping da disco.
Link: https://myoldcomputer.nl/commodore-64/xum1541-promicro/



Come di consueto chiudiamo il nostro articolo invitandovi a visitare la nostra pagina web https://www.dumpclub64.it/ dove potrete trovare per l'occasione una sessione di dumping interamente dedicata ai dischetti originali trasferiti con il 1571 collegato nella modalità **Nibble** con lo XUM1541 realizzato utilizzando l'Arduino Pro Micro riprogrammato.

Vi invitiamo come al solito a dumpare sempre i vostri dischetti per preservarli dall'inevitabile decadimento, in modo da poterli in futuro ripristinare e poi condividere con la comunità internazionale secondo lo spirito che accomuna gli autentici appassionati di retrocomputing e anche per poter disporre di un backup allargato nel caso doveste perdere i vostri dumps. Ci sono ancora molti applicativi, giochi, cracks, intros, immagini e file musicali che aspettano di essere dumpati (specialmente per quanto riguarda tanta produzione italiana) che portano con sé storia, ingegno, cultura e anche i ricordi di chi li ha vissuti: forse proprio tu che ora ci stai leggendo puoi aiutarci a recuperarli per preservarli non solo per te stesso ma anche per tutti quelli che in futuro li vorranno apprezzare.

Chiunque fosse interessato ad aiutarci in questa eroica missione o volesse aiuto per dumpare o semplicemente volesse sostenerci in qualche modo può contattarci scrivendoci un email all'indirizzo dumpclub64@gmail.com oppure attraverso la nostra pagina web https://www.dumpclub64.it/ o se predilige l'uso dei social iscriversi al nostro gruppo facebook: https://www.facebook.com/groups/dumpclub64/.

Lacrime di Dragon... 32

di Alberto Apostolo

Credo che fosse il 1985 quando qualcuno pensò di donare all'I.T.I.S. di Foligno (oggi I.T.T. Da Vinci) un Dragon 32 nuovo di zecca.
Un bidello aveva sistemato il computer sopra uno dei banconi del Laboratorio di Informatica in attesa di essere inaugurato ufficialmente.
Proprio quando la mia classe aveva l'ora di Laboratorio di Informatica, l'esercitazione fu interrotta dall'ingresso del Vice-Preside seguito da alcuni professori e da colui che aveva fatto la donazione (non ricordo più chi fosse).
Durante l'attesa dell'accensione del computer, si era creata un po' di confusione e (senza volerlo) mi ero ritrovato nelle ultime file accanto ad alcuni compagni, tra i quali M.T. da Cave di Foligno (che saluto!).
I secondi passavano e, vedendo che il computer non ne voleva sapere di accendersi e funzionare, M.T. bisbigliò a noi che gli stavamo accanto: "Secondo me non hanno attaccato la spina alla presa...".
Figurarsi se fosse stata vera una cosa del genere! Ma un ghigno beffardo cominciò ad apparire sui nostri volti perché M.T. aveva ragione. Nella solennità del momento, nessuno si era preoccupato di eseguire quella elementare operazione (a scusante va detto che il cavo di accensione e la presa di alimentazione si celavano alla vista dietro il banco di laboratorio).
Infine il computer fu collegato, si accese regolarmente e la "cerimonia" finì tra le risate di tutti i presenti.

Tony Clarke nel Novembre 1982

Le origini del Dragon 32

Il Dragon 32 era un computer prodotto in Galles con il sostegno dell'ente parastatale W.D.A. (Welsh Development Agency) agli inizi degli anni '80 del XX secolo.
Tony Clarke, Direttore Generale di una fabbrica di giocattoli chiamata Mettoy (con sede a Swansea) aveva intuito il crescente interesse dei bambini per i computer rispetto ai giocattoli tradizionali.
Da notare che tra i prodotti tradizionali di quella azienda erano incluse le macchinine Corgi Toys in metallo presso-fuso.
Nel 1980 per competere nel mercato degli home-computer venne fondata la controllata Dragon Data Limited (con il logo in fig.1 che richiama il Red Dragon, l'antico simbolo del Galles).

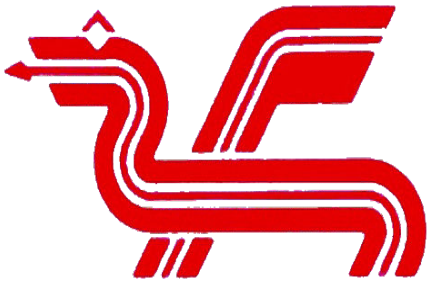


Figura 1

Nell'Autunno del 1981 iniziò il progetto "SAM" con la consulenza della PA Technology di Cambridge. Il gruppo di lavoro era diretto da Ian Thompson-Bell. Fu deciso di puntare sul micro-processore Motorola 6809 anziché sui più conosciuti Z80 e 6502. Questa scelta destò l'interesse del pubblico ma non quello delle software house che non avevano esperienza di sviluppo con quel sistema.
Alcuni dettagli del progetto erano in comune con il Tandy TRS80 Colour Computer; ciò significava che alcuni programmi scritti per il "Co-Co" funzionavano anche sul Dragon.
L'aspetto ricordava quello di una

robusta macchina da scrivere, dotata di 32 KB di RAM, grafica a colori. Lanciato alla fine del 1982 con il prezzo di 199 Sterline, ne furono venduti inizialmente circa 32mila esemplari a partire dall'inizio del 1983. I computer venivano prodotti nello stabilimento di Port Talbot (a pochissimi chilometri dalla sede legale di Swansea).
Le ragioni di questo iniziale successo erano dovute al fatto che la Dragon Data poteva contare sulla rete di vendita della Mettoy e sulla indisponibilità di altri modelli di computer in quel momento.

Le caratteristiche tecniche

Il sistema era costituito da una CPU 8 bit Motorola 6809E (con un clock di 0.84 MHz) e da 32 KB di RAM espandibile a 64. In 16 KB di ROM si trovava il software di base comprendente Sistema Operativo e BASIC (fig.2).

Figura 2

Sulla mother board erano presenti anche i seguenti chip Motorola: il 6883 Synchronous Address Multiplexer (SAM) memory controller, il 6847 video controller e due 6821 Peripheral Interface Adaptor (PIA) per gestire l' I/O.
Il Dragon 32 disponeva di una porta parallela Centronics, due porte per i joystick, uno slot per cartridge ROM e una interfaccia per un registratore a cassette da 1500 baud (con un controllo separato del motore del registratore).
Una uscita UHF permetteva di collegarlo alla TV di casa mentre l'uscita RGB era prevista per collegarlo a un monitor a colori.
Lo schermo aveva una risoluzione di 192 x 256 pixel (16 righe x 32 colonne) ed erano disponibili 8 colori più il nero (fig.3).
Le lettere disponibili sullo schermo erano solo quelle maiuscole (le minuscole comparivano solo in fase di stampa su carta). La tastiera QWERTY incorporava 53 tasti.
Il suono era dato da un solo oscillatore ed era modulato sul segnale video per essere disponbile sull'altoparlante della TV. Non era previsto nessun tipo di modellamento del segnale d'onda per cui la sintesi di strumenti musicali era impossibile, così come la realizzazione di effetti da video-gioco per la mancanza di un generatore di rumore.
Si potevano usare due drive per floppy disk da 5 e 1/4'' a faccia singola comandati da un controller inserito sullo slot esterno (fig.4).
Il BASIC (Microsoft extended colour BASIC) in ROM era largamente compatibile con quello del TRS80 e includeva alcune routine driver per usare la maggior parte di stampanti Centronics.
Per l'utilizzo dei disk drive era previsto il sistema operativo

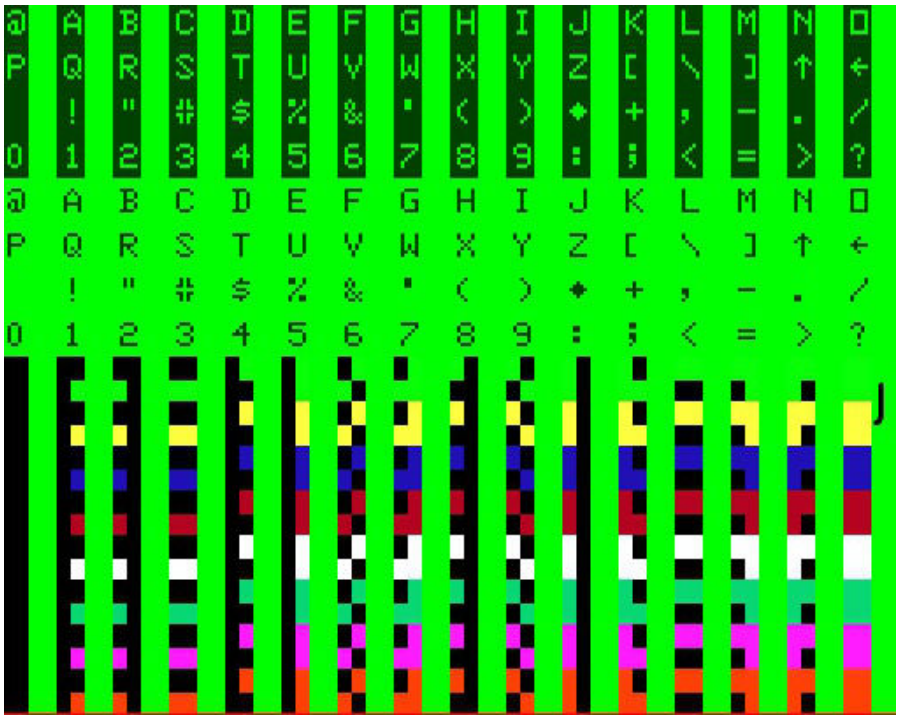


Figura 3

Figura 4

denominato OS/9 (su dischetti).
Per la realizzazione del BIOS, la PA Technology aveva chiamato Duncan Smeed, uno specialista in linguaggio assembler della University of Strathclyde (Glasgow, città nella quale Motorola aveva una forte presenza).

Duncan Smeed nel 1985

La fine della Dragon Data

Nello stesso periodo gli affari della Mettoy non stavano andando bene e l'azienda accumulava perdite (11.5 milioni di Sterline alla fine del 1982). Questo obbligò la Mettoy a cedere la Dragon Data a un consorzio di 6 società (tra cui la W.D.A. e la Pru-Tech) per colmare una parte delle perdite.
Fu messo in produzione un computer più evoluto del Dragon 32, il Dragon 64, ma era troppo tardi e non riuscì ad avere il successo sperato contro una concorrenza che si era fatta molto agguerrita (ZX Spectrum , Acorn BBC, Commodore 64, ecc.).
Nel Settembre 1983 la GEC (una azienda del settore elettronico) aveva rilevato la Dragon Data e nel corso del 1984 l'aveva rinominata GEC Dragon. Brian Moore sostituì Tony Clarke. Furono messi in cantiere due modelli: "Alpha" e "Beta".
"Alpha" aveva lo stesso aspetto di un Dragon 32 ma 256 KB di memoria e due drive per floppy disk da 3 e 1/2' mentre "Beta" assomigliava di più a un PC. Ma nel Maggio 1984 La GEC Dragon finì in liquidazione a causa delle perduranti difficoltà nel realizzare profitti. Fu tentato un salvataggio con la start-up spagnola Eurohard ma nel Dicembre 1984 il boom degli home-computer era finito.

Conclusioni

Il Dragon 32 che avevamo a Scuola era poco utilizzato (con un certo disappunto degli assistenti di laboratorio che lo avevano provato e ne avevano intravisto le potenzialità grafiche).
Le ragioni principali erano la mancanza di periferiche collegate (stampante e drive per floppy disk) unite alla scarsa documentazione che impediva di sfruttare al meglio l'hardware con il processore Motorola 6809 (ritenuto superiore al 6502 e allo Z80 dalla stampa specializzata).
Il sistema era reclamizzato per uso casalingo e didattico mentre l'uso professionale era limitato dallo schermo 16 x 32 caratteri a meno di ricorrere a "trucchi" con la grafica in alta risoluzione (e poi la "O" a forma di rettangolo era una cosa inguardabile rispetto al design delle altre lettere, n.d.A.).

```
(C) 1982 DRAGON DATA LTD
16K BASIC INTERPRETER 1.0
(C) 1982 BY MICROSOFT

OK
10 PRINT "HELLO!"
20 GOTO 10
```

Vi lascio con un invito all'ascolto della canzone "Tears of The Dragon" di Bruce Dickinson, che ha ispirato il titolo dell'articolo (vedi sezione "Link Utili").

Link utili

Tears of The Dragon: https://www.youtube.com/watch?v=nzwV9cW1aaI

Un manuale completo del Dragon 32: http://www.dragondata.co.uk/Publications/InsideTheDragon.pdf

Emulatore in Python: https://pypi.org/project/DragonPyEmulator/

Emulatore XROAR: https://www.linuxlinks.com/XRoar/

Bibliografia

[Dra82] AA. VV. https://nosher.net/archives/computers/dragon_1982-09_001 consultato il 2020/01/05.

[Lea16] T. Lean, "Electronic Dreams: How 1980s Britain Learned to Love the Computer", Bloomsbury Publishing, 2016.

[LS83] D. Longley, M.Shain, "Microcomputer User's Handbook: The Complete and Up to Date Guide to Buying a Business Computer", Springer 1983.

[Smi12] T. Smith, "Dragon 32 is 30", consultato il 2020/01/05, https://www.theregister.co.uk/Print/2012/08/01/the_dragon_32_is_30_years_old/

[Sor84] L.Sorge, "Dragon 32", MC Microcomputer n.27, Febbraio 1984 pp. 46-53, https://issuu.com/adpware/docs/mc027

Intervista a Davide Sattin - "Assemblatore cross-platform 6502"

di Marco Pistorio

M.P: Come ti è venuta l'idea di realizzare proprio un assemblatore, oggi, nel 2020, per generare codice macchina 6502?

D.S: A dire il vero l'idea è nata mettendo insieme alcune esigenze e curiosità, e ovviamente la passione per il mitico Vic 20, o più in generale per le macchine Commodore.
Ho sempre avuto l'idea di costruire un framework per il Language Recognition, e nonostante esistano già dei framework consolidati, ho provato a costruire il mio.
La conseguenza di questo è stata la seguente domanda: "Che linguaggio posso provare ad usare per il mio framework?". Volevo una cosa semplice e l'Assembler mi sembrava la scelta giusta. Ho messo quindi insieme questa domanda con la passione per il retrocomputing.

M.P: Quale linguaggio di programmazione hai scelto per il tuo assemblatore e perchè? L'assemblatore gira allo stesso modo su Mac, Win, Linux?

D.S: Ho scelto di usare tecnologia Microsoft .Net Core (3.1) per la precisione. Questa tecnologia permette (come Java) di poter produrre programmi cross-platform. Per cui usandola già ampiamente nel mio lavoro, mi è venuto naturale usarla per il mio progetto. L'assemblatore gira nello stesso modo, anche se comunque devi tener conto che varie piattaforme hanno specifiche differenze. Queste differenze sono già praticamente coperte da NetCore, ma ad esempio certe piattaforme possono produrre file di testo omettendo alcuni caratteri di controllo, e questo ad esempio potrebbe causare dei problemi nel parsing del file.

M.P: Consideri il progetto già chiuso oppure prevedi un ciclo di sviluppo ampio, con correzione di eventuali bugs (segnalati e non) ed implementazione di nuove features?

D.S: Il progetto non è chiuso, ho già rilasciato due fix, nel frattempo ;). Il problema principale che mi trovo ad affrontare è la documentazione. Mi annoia, ma devo farla. L'assemblatore per il momento offre tutte le funzionalità di base, ed alcune funzionalità più complesse come le espressioni, ma manca di alcune features come le Macro ed altre cose. Ho una roadmap nella quale ho segnato le implementazioni che farò in futuro. Quindi il progetto è vivo e sarà in evoluzione.

M.P: Puoi descriverci le caratteristiche di questo assemblatore? (Ad esempio, è prevista la creazione di macro, il supporto degli opcodes non documentati -illegal opcodes-, l'importazione di files grafici standard oppure ancora è previsto il supporto per eventuali plugins, debug interattivo e/o prodotti di terze parti?)

D.S: Come ti dicevo, l'assemblatore nasce con funzionalità di base che comunque permettono la realizzazione di programmi, ma svilupperò le altre funzionalità con il tempo. Volevo uscire circa nello stesso periodo in cui era prevista l'uscita del "The C64". Questo era il mio obiettivo. Inoltre se avessi dovuto fornire tutte le funzionalità alla prima release i tempi sarebbero stati più lunghi. Comunque si le macro sono previste nella RoadMap, per quanto riguarda gli "Illegal OpCode", sono preventivati come features da implementare nelle prossime release. L'assemblatore essendo a riga di comando può essere semplicemente agganciato a qualsiasi editor che preveda l'esecuzione di comandi esterni. Ad esempio io uso 6502 Overdrive assieme a VsCode. Ovviamente bisogna configurare l'ambiente, ma questo è abbastanza normale per chi usa VsCode.

M.P: In cosa differisce il tuo assemblatore da altri disponibili sul mercato, come ad esempio KickAssembler, cc65, C64Studio, DASM, CBM.Prg Studio (oltre al fatto di essere il "tuo" assemblatore chiaramente...) Quali sue particolari caratteristiche giustificherebbero la scelta di adoperare il tuo assemblatore piuttosto che uno dei già citati?

D.S: Sono tutti ottimi assemblatori, ed a conti fatti anche migliori del mio. In quanto ovviamente hanno un bagaglio di features che ora mi mancano, ma che svilupperò nel tempo. L'obiettivo è consolidare e farlo crescere. Ad ogni modo 6502Overdrive nasce anche per un fatto semplice, tante volte copiando ed incollando del codice, alcuni assemblatori sono molto rigidi nel riconoscere la sintassi. Ad esempio alcuni di questi se metti un opcode senza uno spazio ed un tab, falliscono. Questa è una cosa che mi infastidisce molto per cui ho cercato di creare un Assemblatore che interpretasse le cose senza troppi patemi d'animo. Inoltre ho lavorato molto sulla gestione degli errori. Questa è una peculiarità del mio assemblatore, molto utile soprattutto per chi si cimenta per le prime volte con il linguaggio macchina. Alcuni assemblatori riportano ad esempio un errore generico, nel caso un cui si trovano OpCode con dichiarazioni di Addressing errate.

M.P: Il codice sorgente deve essere scritto con una sintassi particolare per essere correttamente compilato, (come per esempio nel caso di cc65) oppure no?
Può essere facilmente compilato con altri assemblatori che hanno regole sintattiche simili (o del tutto analoghe) a quelle che rispetta il tuo assemblatore oppure segui un tuo standard?

D.S: Ho cercato di seguire il più possibile lo standard, ma ci sono alcune differenze (minime), anzi penso che alcune saranno del tutto opzionali in futuro. Ad esempio le Label sono riconosciute mettendo {nomevariable} seguito dal carattere ':' (due punti). Questo in realtà l'ho fatto volutamente per avere un colpo d'occhio maggiore per chi legge il codice, mi sembrava più chiaro ed elegante. Per il resto le differenze sono esigue. La maggior differenza si trova nel comandi macro built-in, ma ho visto che ogni compilatore ha la sua nomenclatura. Mi riferisco ad esempio al comando CTEXT che serve per definire una stringa Petscii. Altri compilatori hanno il loro nome.

M.P: Questa domanda è un pò di parte in quanto sono, da sempre, un "sessantaquattrista" incallito. Partendo dal fatto che il microprocessore del c64 ovvero il 6510 è una variante estremamente simile del 6502, il tuo assemblatore potrebbe essere usato senza particolari problemi anche per creare codice per 6510?

D.S: Viva il 64! Beh, di fatto il 6510 è una evoluzione del 6502, per cui non ci sono particolari controindicazioni per usarlo, ma dovrò comunque dare un supporto maggiore per chi usa il C64, o più in generale alla possibilità di creare un codice che sia multi target.

M.P: Si tratta di un progetto open-source oppure no? Pubblicherai oppure hai intenzione di pubblicare il codice sorgente del tuo assemblatore oppure ancora pensi di tenere per te il codice sorgente del tuo assemblatore? Sono due scuole di pensiero entrambe valide...

D.S: Per il momento il progetto è ClosedSource, ma non escludo il fatto che possa diventare Open. Questo dipende anche dal fatto che non ho mai gestito un progetto OpenSource, e probabilmente non ne conosco appieno le modalità con cui lavorarci.

M.P: Hai pensato a quali strade percorrere per finanziare il progetto? Anche portando avanti progetti opensource ad esempio, sarebbe possibile chiedere delle libere donazioni oppure offrire del supporto "mirato" dietro congruo compenso.

D.S:Per il momento questo progetto è interamente autofinanziato. Pensavo ad un sistema di donazioni, giusto per rientrare nelle spese del dominio. Però ho altre idee in testa, sempre collegate ad 6502Overdrive. Mi piacerebbe però avere maggiori risorse per poter fare quello che vorrei. Pensavo al Crowdfunding e vedere se qualche anima generosa mi sostiene, ma ne riparleremo più avanti.

M.P: Hai in cantiere altri progetti simili a questo? Focalizzati magari su altre piattaforme?

D.S: Si, pensavo ad un Ide, in questo caso però focalizzato solo ed esclusivamente su Vic20 e C64. Non ho ancora le idee chiare in merito, ci sto pensando!

M.P: Ti ringrazio per la tua pazienza e..in bocca al lupo per il tuo progetto :) Ciao e grazie per averci concesso questa intervista, Davide.

D.S: Ringrazio te e lo staff di RetroMagazine, ed un saluto a tutti i lettori!

Riferimenti WEB

Sito WEB:
https://www.6502overdrive.com/

Gruppo FB:
https://www.facebook.com/groups/2707706952655676/

Programmare da zero un emulatore: intervista a Valerio Lupi

di David La Monaca (Cercamon)

Salutiamo Valerio Lupi e lo ringraziamo per aver accettato il nostro invito per un'intervista. Fra gli interventi al recente evento **Once Upon A Sprite**, tenutosi a Milano il 29 ottobre 2019, uno di quelli che ha maggiormente destato la curiosità del pubblico è stato quello di Valerio che ha presentato il frutto del suo lavoro durato qualche mese tutto incentrato sulla programmazione *from scratch* di un emulatore per Commodore 64. Sebbene il suo intento iniziale fosse soltanto quello di imparare a scrivere un emulatore del C64 in grado di far girare solo un paio di giochi di Jeff Minter/Llamasoft di cui è grande fan, procedendo nel suo lavoro, Valerio ha finito per mettere insieme un prodotto a suo dire completo per metà, ma in grado di far girare la maggior parte dei giochi che ha personalmente testato. Ci sono molti emulatori software per C64 disponibili gratuitamente e funzionanti sui più diffusi sistemi operativi, ma l'approccio di tipo "learn by doing" di Valerio ci ha incuriosito non poco. Siamo tutti abituati a dare per scontata la nostra copia di VICE x64, CCS64, Micro64, VirtualC64 (solo per citarne alcuni), sempre pronta all'occorrenza per giocare ad un vecchio titolo o testare una nuova demo. Quante volte siamo rimasti a bocca aperta dopo il RUN di un gioco e ci siamo chiesti come diavolo hanno fatto gli autori di questi emulatori ad ottenere una riproduzione pressoché perfetta del sistema più conosciuto di casa Commodore? Beh, dev'esserselo chiesto anche Valerio, il quale, a differenza della maggior parte di noi, non si è limitato allo stupore ma ha preso carta, matita e.. no, volevo dire, ha preso schermo, tastiera e IDE e ha cominciato a buttare giù le idee su come costruire da zero un emulatore per C64. Ma prima di addentrarci, con le nostre domande, nelle ragioni che hanno mosso Valerio a cimentarsi nella sua impresa e nel racconto di come ha affrontato e risolto le diverse problematiche cui è andato incontro durante lo sviluppo, cerchiamo di conoscere meglio il protagonista di quest'intervista.

DLM – Salve Valerio, siamo felici di scambiare quattro chiacchiere con te sul retrocomputing in genere e in particolare su vC64-Emu, l'emulatore di C64 cui ti sei dedicato nel corso del 2019. A beneficio dei nostri lettori puoi brevemente presentarti e darci qualche notizia sulla tua formazione (quando e dove sei nato, scuole frequentate, crescita personale e professionale, le tue passioni ed inclinazioni, interessi personali al di là del lavoro, ecc.)?

VL - Dunque, sono nato a Firenze il 31 dicembre del 1974 (in realtà sono proprio l'ultimo nato a Firenze del 1974!) quindi ho da poco compiuto 45 anni. Ho vissuto gran parte della mia vita a Livorno e ora vivo in campagna, in provincia di Arezzo, con la mia compagna, 2 gatti e 2 cani (io sono il gattaro, i cani appartengono a lei e tocca sopportarli!). La mia formazione è tipica di molti della vecchia guardia: niente laurea in informatica, ragioneria alle superiori (diplomato ragioniere programmatore, usavamo Turbo Pascal e anche il Cobol, un vero lusso!), poi Biologia all'università (che non ho finito). La mia 'carriera fra bit e byte' è iniziata, come per molti a quei tempi, come 'pirata informatico'. Attraverso diverse peripezie (ci vorrebbe un'intervista a parte per parlarne!) mi appassionai al reversing e alla sprotezione del software su PC, dopo che per lungo tempo ero rimasto affascinato dalle intro dei gruppi che precedevano i giochi piratati su C64 e Amiga (dei quali ero utilizzatore avanzato e smanettone, ma su quelle macchine non mi ero ancora dedicato allo sviluppo/hacking). Quando avevo circa 23 anni, tramite passaparola, trovai lavoro in un'azienda che aveva bisogno di uno sviluppatore low-level su Windows, lavoro che lasciai presto per dedicarmi a quello che poi è diventato il mio lavoro di sempre e che svolgo tutt'oggi, ossia lo sviluppo di strumenti software nell'ambito della sicurezza informatica per diverse aziende, sia italiane sia estere. Per quel che riguarda i miei hobby, beh... sono una persona semplice e molto abitudinaria. Mi piacciono la buona tavola (e si vede!!!), il buon vino, i buoni sigari, la natura, visitare posti sperduti e ricchi di storia. In un'altra vita avrei studiato storia antica e sarei diventato un archeologo!

DLM – Immagino che tu sia stato un appassionato di computer fin da ragazzo. Che cosa ti ha spinto ad avvicinarti al mondo dell'informatica e qual è stata la tua prima esperienza diretta con un computer? Scommetto che il tuo primo home computer è stato proprio il C64...

VL - Scommessa vinta! Certo che è stato il C64! Come ho raccontato anche durante il mio talk al recente OUAS 2019 di Milano, il mio primissimo approccio con i computer è stato quando il mio babbo mi portò a casa un C64. Io volevo giocare al gioco del tennis che aveva il mio amichetto sull'Intellivision e così scrissi 'voglio il gioco del tennis' sul computer e il computer rispose ovviamente con un bel 'syntax error'. Ricordo che andai dal mio babbo in lacrime e dissi: "Babbo, il computer non funziona, è rotto!". Povero babbo, chissà, se mi avesse veramente comprato l'Intellivision come desideravo, magari la mia vita avrebbe preso tutt'altra piega e oggi sarei davvero un archeologo e non un informatico - ahhahah!

DLM – Quando hai cominciato a guardare al "lato oscuro" della programmazione di un computer? Ovvero quando hai iniziato a scoprire che esistevano comandi BASIC differenti da LOAD e RUN necessari per caricare i giochi?

VL - In realtà ho cominciato molto tardi, solo col PC. Ma ero ancora piccolo quando il mio babbo mi portava a casa la domenica, insieme all'immancabile Topolino, riviste come Commodore Computer Club (con tutti quei

listati da copiare), MC Microcomputer, ma soprattutto i fascicoli dell'enciclopedia 'Il Mio Computer' della DeAgostini, che usciva a dispense in edicola. Ne custodisco ancora gelosamente gli 8 volumi che mio padre fece rilegare. Non so spiegare come mai mi portasse le riviste sui computer. In fondo, mio padre lavorava in banca e non ci capiva una virgola di informatica!

Ne 'Il Mio Computer' trovavo corsi di assembler per CPU 65xx e Z80 avanzatissimi e molto chiari. Non che ci capissi molto a quei tempi, ma oggi posso dirti che davano e danno la paga a molti tutorial e corsi che trovi su Internet, Erano davvero altri tempi. Per finire, visto che hai suscitato in me il ricordo di Commodore Computer Club, beh... non sai che emozione a OUAS 2019 nel conoscere di persona il direttore di quella rivista "storica". Ero poco più che un bambino quando leggevo quella rivista ed ero affascinato da tutti quei listati. Li copiavo e li lanciavo sul C64 ma ci capivo ancora molto, molto poco!

DLM – Hai subito trovato interesse nella programmazione e nello scoprire come un computer funziona intimamente?

VL - Sì, assolutamente! Anche se la mia carriera è nata come cracker (mi occupavo di sproteggere giochi/programmi per vari gruppi - UCF, XForce, Class, Quartex - della scena warez fine anni 90/primi 2000 e anche con un discreto successo), in seguito mi sono spostato totalmente sullo sviluppo, che dà immensamente più soddisfazione. Ma non sarei certamente dove sono adesso senza il passato da cracker. E' stata quella 'formazione' che mi ha insegnato, in fin dei conti, come funzionano le CPU e come ragionano l'hardware e i sistemi operativi internamente. La tecnologia va avanti, ma alla fine i concetti a basso livello sono per il 90% sempre i soliti. Per chi sviluppa, ciò che è cambiato negli anni è essenzialmente la miriade di layer di astrazione/complicazione che di volta in volta sono stati aggiunti.

DLM – Ricordi qual era la configurazione completa del tuo primo computer (incluse periferiche, memorie di massa, espansioni ed accessori)? Lo possiedi ancora o ne hai presi altri nel tempo?

VL - Certo che mi ricordo! Allora: Commodore 64 con Datassette, la cassetta di "Frogger" della Interceptor Micros e la cartuccia di "Radar Rat Race" originale Commodore, tutto acquistato nel 1983 per 720.000 lire. Un paio d'anni dopo (mi pare) sono arrivati il floppy drive 1541 e la stampante MPS-803. Mai avuto un monitor, sempre usato la TV. In seguito montai lo SpeedDos (operazione effettuata da un fotografo di Livorno che mi feci amico e che mi passava regolarmente i giochi gratis; lui era nel giro di Fantasoft, gli "avversari" dei milanesi Niwa e Pier). Provai anche le varie cartucce OMA, NIKI-II, ISEPIC: non sapevo programmare, ma grazie a queste ero riuscito a capire, andando completamente a caso, come 'rippare' le intro dei pirati, rimarchiarle maldestramente 'LUPISOFT' e ripatcharle nei giochi! Purtroppo no, non ho più niente del materiale di quei tempi :(Nel tempo invece ho accumulato un sacco di retroconsole ma, stenterai a crederci, nessun retro-computer!!!

DLM – Che cosa ti ha spinto ad avvicinarti alla programmazione in concreto (amici, riviste, libri, curiosità)? Con quale linguaggio o strumenti software hai cominciato?

VL - Faccio prima a raccontarti come mi sono avvicinato al cracking, la programmazione è venuta dopo, come una naturale evoluzione. Dunque, preparati a ridere: avevo 17 anni ed ero reduce da un intervento chirurgico per cui mi muovevo poco. Mi capitò in mano (tra i giochi dell'abbonamento mensile al piratone di fiducia) un gioco porno per PC che era sprovvisto di crack. Sappiamo tutti che in quel periodo dell'adolescenza noi maschietti siamo molto 'sensibili' a certi argomenti, quindi DOVEVO vedere quel gioco in funzione. Così mi armai di pazienza (tempo a quell'età di solito se ne ha molto, e poi io non avevo molto altro da fare visto che ero costretto a restare a casa) e ricordai che su un altro dischetto avevo un file .txt che spiegava come usare DEBUG.COM di MS-DOS. Quindi, senza sapere niente di assembler x86 (conoscevo solo, e poco, il Pascal che avevo studiato a scuola) e contando solo sul fatto che un programma deve comunque andare avanti (e questo più o meno era giusto), cominciai a smanettare sull'eseguibile del gioco. Notai che gli opcode avevano nomi simil-inglesi (JMP=jump ?, JNZ=jump not zero ?, MOV=move, di questo ero sicuro perché vedevo che i registri cambiavano valore) e poi andai avanti così perlustrando il codice del gioco. Beh, che ci crediate o no, ho cominciato proprio così. Andando completamente a caso, riuscii poi a far partire quel benedetto gioco (per la cronaca, di donne nude nemmeno l'ombra, ahimè, il gioco si rivelò di una mezza ciofeca) e mi appassionai alla cosa, iniziando a crackare altri giochi e imparando sempre di più mentre smanettavo. Da lì nacque il mio pseudonimo "Xoanon", termine trovato per caso sul vocabolario, come da tradizione. Mi piaceva la parola (è un tipo di totem greco), che diventò poi "xOANINO" (diminutivo) e infine "valerino", in fin dei conti tutti mi hanno sempre chiamato così, quindi perché non usarlo come nickname? Come racconto sempre, quindi, tutto è nato per una sega - ahahahahah! Per finire la domanda, ho iniziato con l'assembler x86, per poi passare al C dopo qualche anno. All'inizio programmavo solo in assembler (e facevo qualcosina in Pascal). Oggi posso dire di conoscere parecchi linguaggi di programmazione inclusi diversi assembler. In ogni caso, conoscendo il C, il C++ e magari un assembler, in fin dei conti i concetti base sono sempre gli stessi: tra un linguaggio ed un altro cambia la sintassi e poco più, approcciarne uno nuovo da zero (quantomeno per iniziare a scriverci qualcosa e poi andando avanti col mio solito metodo per acquisire esperienza) è questione di poche ore.

DLM – Sapevamo (abbiamo indagato un po' su di te) della tua adesione a gruppi della scena cracking internazionale per PC. Come ti è venuta l'idea di far parte di questo vero e proprio stile di vita? Puoi

parlarci dell'atmosfera respirata in quei giorni? Come eravate organizzati e come comunicavate l'uno con l'altro?

VL - Sì, sono nato come cracker e me ne vanto pure :). Anzi, posso dire di essere stato il primo a pubblicare un manualetto di cracking in italiano agli albori di Internet. Ho sempre avuto questa cosa di cercare di 'divulgare' quello che imparavo, un po' come un Aranzulla ante-litteram. Lo facevo solo per la gloria, però, e non per soldi :). Sto parlando di "The Xoanon's Guide To Cracking', un manuale che scrissi a metà anni 90. Cercalo pure, ancora è possibile trovarne una copia online. Veramente ridicolo a rileggerlo oggi, ma a quei tempi era oro e chi ha iniziato in quel periodo difatti lo ricorda con affetto. Come spesso dico scherzosamente a tanti ex-reverser italiani di quei tempi, che oggi magari hanno la loro azienda e/o sono stimatissimi researcher nel campo della cybersecurity, "siete un po' tutti figli miei!"

Allora, come mi è venuta l'idea...? Vediamo, sono sempre stato affascinato dalle intro dei gruppi pirata che precedevano i giochi. Non so perché in realtà, mi affascinavano e basta. Come ho detto prima, da ragazzino col C64 ero persino riuscito a capire come ripparle e cambiarne la grafica. Ero davvero quello che all'epoca si definiva un "lamer" :). L'atmosfera che si respirava? Beh, niente di paragonabile ad oggi, Internet è venuta dopo. Allora ci si scambiava i dischetti a mano, la pirateria non era un male, si partecipava ai copy-party dove andavi per scambiare giochi (i più famosi party erano tenuti dai gruppi del Nord-Europa), tempo di phreaking, BBS, e molto, molto altro. Continuando così mi dilungherei veramente troppo. Qualche anno fa scrissi un articolo per il sito Lega Nerd. E ne abbiamo parlato a lungo anche nelle passate edizioni di OUAS. Io e Antonio (Randall Flagg di Razor1911) abbiamo fatto un paio di talk a riguardo, anche durante l'ultimo OUAS del 2019!

DLM – Hai mai frequentato corsi di informatica professionali o di livello universitario? Oppure sei, come pensiamo, uno straordinario autodidatta del design di software e di coding nei vari linguaggi in cui ti cimenti?

VL - Per carità, Dio ce ne scampi e liberi! :) Ho imparato sia a crackare che a programmare con puro metodo sperimentale, tipo trial & error, andando completamente a caso e aggiustando il tiro quando sbagliavo, prendendo nota degli errori e imparando grazie a questi. Ed è ciò che faccio tuttora, quando devo fare qualcosa in un linguaggio che non conosco (come credi che abbia imparato l'assembler 65xx per lavorare sull'emulatore?!): prendo il compilatore e inizio a scrivere codice. Non ho mai usato un libro di programmazione in vita mia, lo giuro! Oggi naturalmente c'è Google a disposizione, ma anche ai tempi ho sempre fatto così, magari dando uno sguardo a quel che trovavo sulle riviste, ma leggere/studiare prima di mettermi a programmare, mai fatto. O meglio, qualche volta ho provato a seguire quest'approccio, ma dopo poche pagine di solito mi annoio e lascio perdere. Guardo un manuale o un libro soltanto quando mi serve davvero, in caso. Detto tra noi, i coder (ma anche i reverser, che oggi si fanno chiamare cybersecurity researcher) più bravi in assoluto che conosco hanno il diploma in tutt'altra materia, alcuni addirittura solo la 3a media, nessuno o proprio pochissimi hanno studiato informatica o ingegneria.

DLM – Nella programmazione la tecnologia e gli strumenti disponibili sono certamente cambiati col tempo. Come affronti il processo di sviluppo di un'applicazione? Usi metodi diversi rispetto al passato? Le architetture hardware di oggi offrono abbastanza stimoli alla creatività di un coder, secondo te?

VL - Come ho detto prima, io ho un approccio tutto mio.

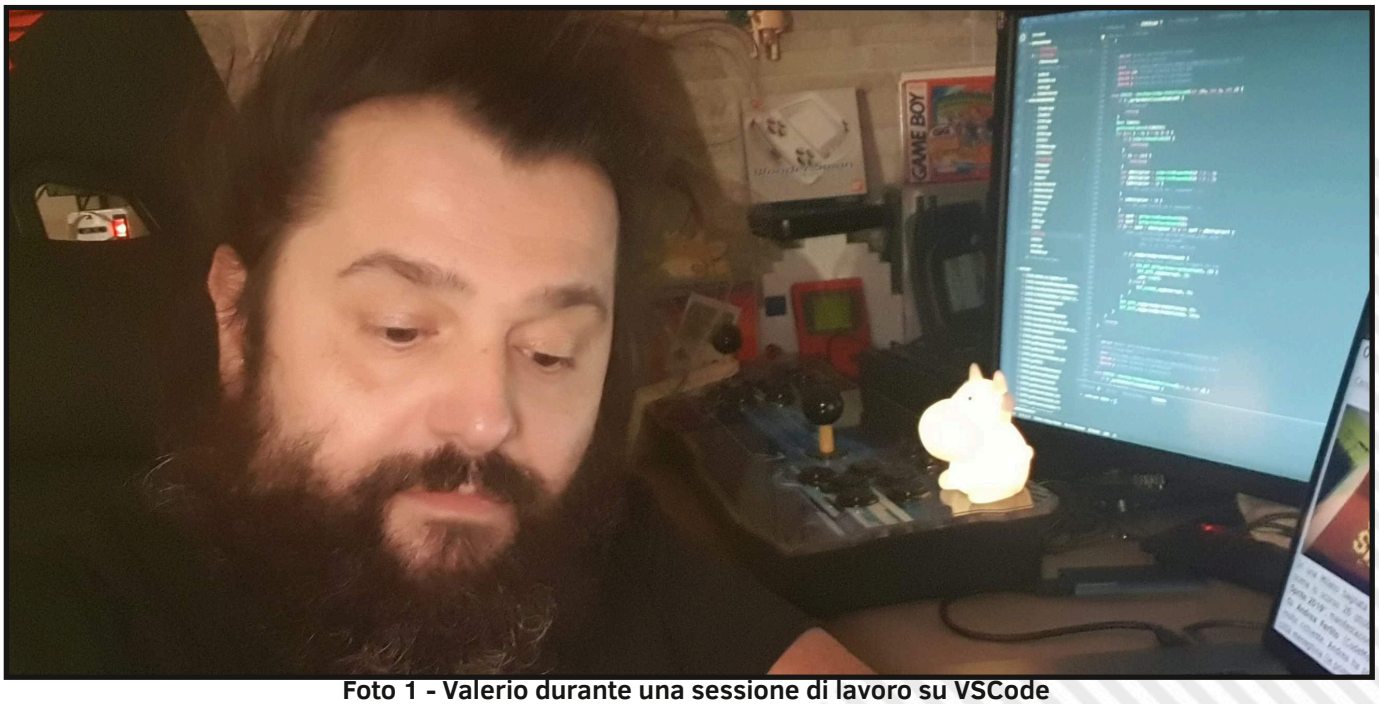


Foto 1 - Valerio durante una sessione di lavoro su VSCode

Inizio prendendo il kit di sviluppo o il compilatore e provo a compilare un semplice "Hello World!". Leggo la documentazione solo quando mi blocco o quando mi serve: ho fatto sempre così, e non mi metterò certo ora a leggere libri e libri prima di provare a programmare su un sistema che non conosco :) Il bello è anche questo. Se dovessi mettermi a cercare di sapere tutto prima di cominciare a programmare perderei anche lo stimolo, credo: io ho bisogno di sbagliare per imparare :). Quanto alle architetture hardware, beh, oggigiorno c'è più o meno solo un'architettura hardware, che è quella del PC. Anche le console come Xbox One e PS4, alla fine sono architetture PC x86. Prima era diverso, c'erano architetture una diversa dall'altra (si pensi ad Amiga coi suoi chip custom, fino ad arrivare alla PS3 col Cell), si sviluppava soprattutto in ASM per ragioni di performance ed era tutto diverso passando da una CPU all'altra. Oggi l'utilizzo dell'ASM è veramente limitato, sono disponibili kit di sviluppo e librerie di alto livello che astraggono l'hardware. Hardware che, gira e rigira, sostanzialmente è quasi sempre un PC x86. Direi che la creatività si è spostata più sui contenuti che sullo spremere l'hardware al massimo come si faceva un tempo. Si pensi ad esempio alle diavolerie spettacolari che si inventavano nel passato, tipo per far sembrare che il C64 avesse più di 16 colori. Anche i vari engine come Unity ed Unreal vanno in questa direzione. Sono tool di alto livello in cui si è più liberi di sviluppare le proprie idee dal punto di vista del game design, pensando un po' meno a fare magie, il che appunto è compito di chi sviluppa gli engine "liberando" così il game designer. Per quanto mi riguarda, beh, io sono di un'altra parrocchia ovviamente. Sono all'antica e farei sempre come Jeff Minter che sviluppa il suo engine (in realtà l'engine è di Ivan, Jeff ci sviluppa i giochi sopra) e si diverte molto di più sicuramente. Mi ha sempre più affascinato il lato tecnico, il cosa stava dietro ai giochi, più che il lato di game-design in sé.

DLM – Ricordi qual è stato il primo videogame che hai giocato sul C64? Ti piace ancora utilizzare l'hardware reale di un tempo?

VL - Me lo ricordo molto bene! Sono stati i primi giochi che babbo mi portò a casa. Primo fra tutti "Radar Rat Race" su cartridge della Commodore (che funziona anche sul mio emulatore!). Poi la versione dell'arcade "Frogger" della Interceptor Micros, che avevo su cassetta (ma questo purtroppo non funziona). Purtroppo non ho C64 reali, ma ho un sacco di retroconsole che spesso uso. Quelle portatili (tipo GB, GBC, NeoGeoPocket, ecc.) me le sono portate a casa e le ho usate per decorare la scrivania sulla quale lavoro. Quelle "da tavolo" (tipo NES, MegaDrive, ecc.) ahimè, sono in cantina a casa di mia madre a Livorno a prender polvere. Devo decidermi a riesumarle e dar loro degna sistemazione. Ne ho veramente tante, tutte le maggiori sicuramente.

DLM – E veniamo nello specifico al tuo progetto di retrocomputing di creare un emulatore per C64 da zero. Come ti è venuta quest'idea? Perché hai scelto il Commodore 64 come computer da emulare? Si è trattata anche di una specie di sfida con te stesso?

VL - Beh, come avevo già accennato, oltre al fatto che il c64 è stato il mio primo computer, sono un fan sfegatato di Llamasoft e di Jeff Minter (da quando il mio babbo mi prese la cassetta di Hover Bovver, nel 1984). Per me Jeff rappresenta quello che dovrebbe essere uno sviluppatore: libero, indipendente, magari non milionario ma vive nella sua fattoria in mezzo al Galles, coi suoi animali, fa quel che gli piace di più e a quasi 60 anni ha sviluppato su qualsiasi sistema esistente e spacca ancora il culo ai paperi (chi ancora non ha provato Polybius corra a comprarsi una PS4 con il kit PSVR!). Io in realtà sono rimasto intrappolato (per soldi, non lo nego) in questa storia della cybersecurity mentre il mio sogno da piccolo era quello di sviluppare giochi e diventare come Jeff. Per me è sempre stata una figura

Foto 2 - Valerio immortalato con un C64 autografato da Jeff Minter.

mitologica e ancora non ci credo che siamo diventati amici e ci sentiamo spesso!! Comunque, ad oggi, siccome scarseggio di fantasia, creare un gioco sarebbe stato troppo complicato. Da tempo poi volevo provare a cimentarmi con un emulatore: mi ha sempre affascinato la cosa, dai tempi di quand'ero ragazzino e bazzicavo Giovanni Bajo (Rasky) e Stefano Crosara (Moonshado) che lavoravano a Psyke, uno dei primi emulatori di Playstation. Ma a quei tempi non ero ancora pronto. Tra l'altro, poi li ho trascinati ambedue nella cricca di OUAS :). Così, l'anno scorso, mi sono detto: bando alle ciance, per questa edizione di OUAS programmo un emulatore di C64 completamente da zero e senza sapere proprio da che parte cominciare (tipico stile mio insomma!). Ci faccio girare quantomeno un gioco di Jeff come mio personale tributo a Llamasoft e a colui che sarei voluto/potuto diventare se, appunto, non mi fossi venduto al vile denaro!

Ovviamente non ho mai pensato, neanche lontanamente, di mettermi in competizione con VICE o con altri emulatori blasonati, figuriamoci... Volevo (DOVEVO) riuscire a fare qualcosa di funzionante senza sapere UNA VIRGOLA di come funzionava il C64, il suo hardware, il 6510, il VIC, insomma, niente di niente! Hai ragione: una sfida con me stesso, sì. Ed era veramente la prima volta che mi cimentavo con qualcosa di grafico, io che ho sempre fatto software 'nascosti'! Beh, in realtà, ci fu un'altra occasione, su Gameboy Advance tanti anni fa... Chi mi conosce sa di che parlo, ma tralasciamo, la storia è un po' imbarazzante - ahahahahah e comunque non mi ricordavo più quasi niente di come si disegna su un framebuffer, pixel per pixel!

DLM – Che tipo di approccio hai utilizzato per rappresentare le diverse componenti hardware del C64 (CPU, memoria RAM, chip dedicati alla grafica e al suono, gestione input/output, ecc.) all'interno dell'emulatore?

VL - Puoi immaginarlo, ho usato l'approccio che uso di solito con ogni software che programmo, ossia dividere il problema in sottoblocchi (oggetti) e sviluppare ognuno di questi nel modo più indipendente e riutilizzabile. In totale ci sono 3 progetti separati (sono proprio dei repository separati), tutti e 3 sviluppati in C++: 1) la CPU (v65xx) è un progetto a sè stante, riutilizzabile in qualsiasi altro progetto. 2) Una libreria di utilities, sempre come progetto a sè stante (emushared): in realtà mi è servito molto poco astrarre utilities per gestire la portabilità, visto che già utilizzavo SDL, ma tant'è, per favorire il riutilizzo di codice in futuri emulatori ho pensato di separare certe utilities che mi sarebbero potute servire anche altrove. 3) Il progetto principale, vC64-Emu, contiene infine l'implementazione dell'emulatore vero e proprio, che linka le due sopracitate librerie (la CPU e la libreria di Utilities generiche) e va a completare il puzzle implementando i vari chip che compongono l'hardware del C64 e li 'orchestra' dando vita all'emulazione. Tutto questo sempre secondo una logica il quanto più possibile indipendente e riutilizzabile: ogni chip implementa dei metodi read, write e update comuni. Da notare che con un'architettura simile, se domani decidessi di utilizzare un altro emulatore di CPU per vC64-Emu, basterebbe fare un bridge in maniera che questo esponga l'interfaccia IMemory esposta dal mio v65xx: et voilà, ecco cambiato emulatore di CPU. Lo stesso discorso vale per le implementazioni dei vari chip all'interno del progetto (guardandomi indietro effettivamente dovevo fare un'interfaccia anche qui come per IMemory, per esporre read, write, update. Va beh, il concetto non cambia).

DLM – Puoi spiegarci a grandi linee quali sono state le tue scelte di design e d'implementazione dell'hardware reale?

VL - In sostanza, ho creato un design a blocchi quanto più indipendenti possibili, più o meno come accennato sopra: OOP for the win! :) Ho creato diversi oggetti ed ognuno di essi rappresenta un 'pezzo' del c64: la memoria (CMemory), il PLA (CPLA), il VIC-II (CVICII), i due CIA (CCIABase e le derivate CCIA1 e CCIA2), il SID (CSID, che però non ho implementato, quindi non c'è audio nell'emulatore). Ho ricostruito quindi, ad oggetti, qualcosa che assomiglia alla mainboard del C64 :). La funzione main() dell'emulatore non fa altro che instanziare per prima cosa CMemory ed inizializzarla con le ROM del C64 (che sono da scaricare a parte online, le 3 roms KERNAL, CHARGEN e BASIC).

Come secondo step, viene instanziata la CPU (CMOS65xx) alla quale si passa l'oggetto memoria. Poi, si passa ad inizializzare i vari chip (i 2 CIA, VIC e SID) passando loro l'istanza della CPU, che diventa un canale di comunicazione con cui ogni chip può accedere a sua volta all'oggetto memoria (in pratica così ho simulato il BUS di sistema che interconnette i vari blocchi). Si inizializzano poi i vari sottosistemi/manager CInput per gestire l'input attraverso il CIA1, CDisplay per gestire l'output a video attraverso il VIC, CAudio che al momento non fa niente. Questi blocchi interconnessi permettono di gestire a più alto livello concetti come audio, input e display, che sono di fatto generici... Volevo un mainloop bello pulito.

Infine, si avvia la CPU e si inizializza il main loop che:
• Esegue un'istruzione della CPU e conteggia i cicli.
• Aggiorna lo stato degli altri chip, tenendo conto dei cicli di CPU che questi possono consumare.
*• Una volta raggiunto il numero di cicli per frame (che, a farla breve - ma non è proprio così - su un C64 PAL è circa 312 linee * 63 cicli di clock per linea = 19656), viene fatto l'update dello schermo e si chiede a SDL di fare il poll dei suoi eventi interni (tra cui controllare lo stato della tastiera per poter 'reagire' agli input).*
• Nonostante il tempo occupato da SDL per gestire i suoi eventi, l'update ahimè avverà sempre troppo velocemente, a causa dell'elevata velocità dei PC moderni. Ma non dobbiamo dimenticare che il C64 (PAL) gira a 50hz, 50 frame per secondo: serve quindi effettuare una sleep per i msec (millisecondi) necessari a 'lockare' il tutto a 50hz.

Et voilà, l'emulatore è fatto! :) Ora dico "et voilà", come se fosse una cosa semplice, ma per capire sta cosa del locking col framerate c'ho messo giorni, visto che non provengo dal mondo del gamedev. Le prime versioni dell'emulatore funzionavano con la velocità del framerate completamente sbagliata :). Per questo ringrazio tanto Giovanni che mi ha spiegato per bene questo concetto del locking ad un tot di FPS. Col senno di poi sembra banale, ma c'ho perso veramente tanto tempo!

DLM – Quale piattaforma (sistema operativo, linguaggio, IDE) hai scelto quando hai deciso di buttarti nell'impresa? E come mai proprio questa piattaforma e non un'altra?

VL - Il mio approccio, in generale, usato non solo nello sviluppo di vC64-Emu, è di rendere i miei progetti quanto più portabili tra le varie architetture e OS. Quindi va da sè che ho scelto SDL come libreria per astrarre video/input/audio (ricordo che l'audio non è stato implementato, eheh). Dopo la pubblicazione ho ricevuto feedback che il progetto compila e funziona pure su un Raspberry PI, per cui direi che ho centrato l'obbiettivo quantomeno da questo lato :). Avevo iniziato il progetto in Golang (stavo imparando Go e farci un emulatore mi sembrava una buona 'palestra') usando GoLand di JetBrains, un ambiente di sviluppo integrato molto carino (IntelliJ, sul quale è sviluppato GoLand, è alla base pure di Android Studio) comprensivo di debugger e tutto il resto. Purtroppo, dopo un po', mi sono accorto che il debugger di Go (Delve) non supportava la visualizzazione degli int come hex: non chiedetemi perché un debugger non debba supportare questa feature, ci sono centinaia di bestemmie in merito, tra cui le mie, sul loro GitHub! E IntelliJ, da buon layer di astrazione (alla fine è soltanto un IDE che usa i vari compilatori/debugger esistenti) se ne sbatteva, quantomeno a quei tempi, di implementarlo. Programmare un emulatore senza avere un debugger che mi permettesse di visualizzare la memoria in hex era fuori discussione, quindi ho buttato alle ortiche quel poco che avevo fatto e sono ripartito da zero in C++ utilizzando CLion (sempre di IntelliJ, ma che non ha quel problema appoggiandosi a GDB/LLDB). Nel mentre rimasi folgorato sulla via di VSCode (che ora è diventato il mio ambiente di sviluppo per TUTTO), quindi, aiutato anche dal fatto che CLion alla fine usa progetti cmake, switchare su VSCode è stato abbatanza indolore. Ah, ora che mi ricordo... apro una simpatica parentesi: mentre passavo a VSCode m'era ripresa la mattana di fare tutto in Go. C'era lo stesso problema del debugger, ovviamente, ma mi dissi: ok, aggiungo il supporto al plugin ufficiale di Microsoft per Go dentro VSCode... Apriti cielo, la pull request che lasciai loro non fu reputata valida per delle segate: in pratica, mi dicevano fai così invece di cosà, oltretutto postandomi proprio il codice corretto e rifai il commit. Io gli risposi "Beh, perché devo rifare il commit? Correggete pure che non mi offendo mica!". Niente da fare, non fu accettata, a me non andava di usare un plugin che comunque sarebbe rimasto solo mio e avrei dovuto mantenermelo aggiornato con le nuove versioni e così ritornai ai vecchi santi del C++. Per finire, il plugin di Microsoft per Go su VSCode NON ha il supporto per visualizzare gli int come hex, pur essendo banale da implementare e la mia pull request è ancora lì appesa per chi vuole andare a cercare e correggerla, non io, chiudo parentesi.

DLM – Immagino che all'inizio tu abbia proceduto nello sviluppo inserendo nel progetto quanta più logica possibile, senza badare alla fase di test o debug, oppure hai trovato un modo di verificare la bontà del tuo lavoro anche nelle prime fasi di sviluppo, magari creando un ambiente di test ad hoc?

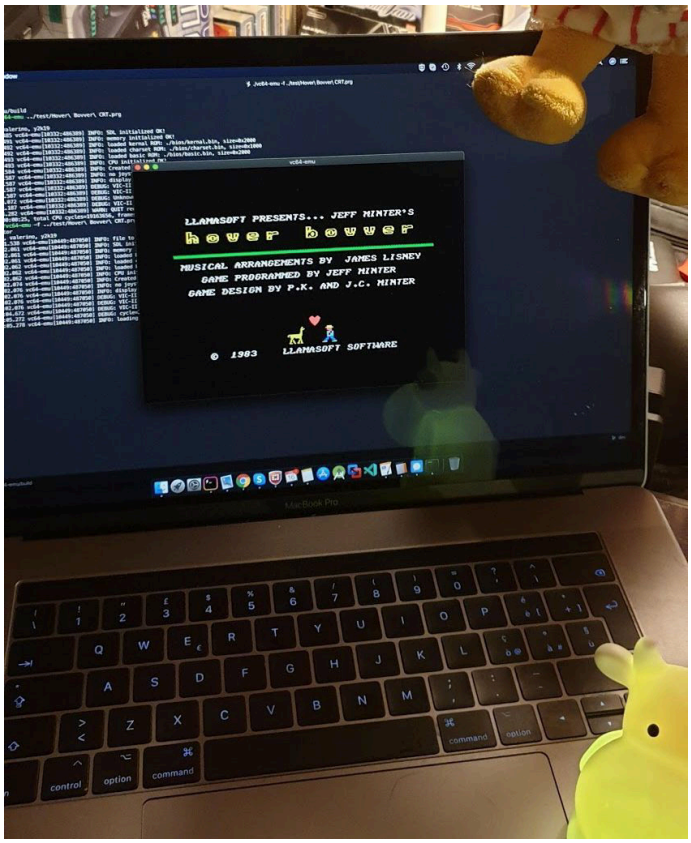


Foto 3 - vC64-Emu in azione con Hovver Bovver

VL - Per prima cosa ho sviluppato l'implementazione della memoria (un semplice array di 64k), la logica per caricarci le ROM, e poi sono partito con lo sviluppo di v65xx (la CPU) facendo un semplice main() di test che istanziava il tutto e avviava la CPU con le ROM caricate. Lo scopo iniziale era, dopo aver effettuato il 'boot', ritrovarsi nella memoria video a $0400, dove il VIC va a 'pescare' per disegnare lo schermo, la famosa stringa 'C64 BASIC V2 ... 38911 Basic bytes free'. Questo avrebbe voluto dire che l'emulazione della CPU funzionava già abbastanza bene e che il C64 emulato aveva almeno completato il boot. Prima di implementare altro (quindi iniziare l'implementazione del VIC e il display a schermo) ho voluto comunque accertarmi che l'implementazione della CPU fosse giusta: mi sono avvalso quindi di alcuni test 'ufficiali' che ho trovato su 6502.org, tra cui il 'Klaus Test' che da quanto ho capito è un po' il test di riferimento per il 6502 (il 6510 del C64 è praticamente un 6502 con un paio di differenze trascurabili). Con questo test si va a testare ogni possibile opcode con ciascuna modalità di

indirizzamento del 6502 (ce ne sono parecchie!) e passato questo possiamo essere sicuri che la CPU è implementata bene al 100% o quasi. In questo modo si può essere certi che, in caso di errori, il bug è altrove e possiamo scartare la CPU. In pratica, ho implementato in v65xx un metodo che esegue il test di Klaus e ne controlla il risultato.
Dai e dai alla fine (ADC e SBC m'hanno fatto parecchio penare!!) ci son riuscito e la mia CPU ha passato il test. **Ecco qua il momento: :)** *Da notare che nel mentre ho anche implementato un debugger all'interno di v65xx, in maniera da poter controllare lo stato dei registri e della memoria, eseguire istruzioni passo passo, mettere breakpoint su indirizzi o interrupt ecc...* **(foto debugger)** *- Una volta finito v65xx, beh, dovevo pur vedere qualcosa!*
Ho iniziato quindi lo sviluppo del VIC e dei due CIA (come vi ho detto ho tralasciato l'emulazione del SID), utilizzando le callback di v65xx on read/on write per eventualmente leggere/scrivere, invece che dalla memoria emulata, dai loro registri 'memory mappati'. A seconda di come viene configurato il PLA, infatti, accessi in lettura/scrittura di determinati range di memoria vanno a leggere/scrivere nei registri di VIC/CIA1/CIA2 invece che nel/dalla memoria. Inoltre, sempre a seconda di come è configurato il PLA, l'accesso a determinate aree di memoria in lettura è mappato sulle ROM del BIOS (KERNAL, BASIC e CHARGEN), mentre in scrittura sempre sulla memoria. Anche qui, ad un certo punto, una volta che le routine di draw dello schermo in character mode (la modalità video 'standard' del C64, quella settata al boot) hanno iniziato a funzionare e la basilare emulazione dei registri del VIC e del CIA2 pure (il CIA2 dice al VIC dove è mappata la sua memoria video), sbam!!! - lo schermo era pieno di caratteri! E, sinceramente, da lì in poi ho capito che in qualche modo ce l'avrei fatta ad arrivare dove volevo. Dopo questo momento si è trattato più o meno di fare solo bugfixing e implementazione di cose mancanti, ma per lo meno riuscivo a vedere meglio quel che succedeva :). Per inciso, **il momento è questo** *e sono stato veramente tanto felice quel giorno!*

Come avrete capito (e come anche ho specificato nelle slide di OUAS che trovate in [R1]) i momenti salienti dello sviluppo sono stati tutti filmati e rimangono a imperitura memoria sul mio twitter, che vi invito a visitare (e a followarmi, così divento influencer ricco e famoso - ahahhaah!!). Basta che scorriate indietro fino ai primi del 2019 e trovate tutti i tweet fino a fine ottobre quando era in programma OUAS :) In realtà avrei voluto sviluppare l'emu in live coding su Twitch o YouTube. Ma la banda che ho a casa purtroppo non me l'ha permesso.

DLM – Qual è stata la tua fonte per trovare tutte le informazioni di carattere tecnico relativo alla circuiteria e all'architettura dell'elettronica usata da Commodore attorno al 6510?

VL - Ho fatto qualche ricerca su Google, ormai si trova di tutto, ma direi che https://www.c64-wiki.com/ è un ottimo punto di partenza per farsi un'idea, si trovano riferimenti e info un po' su tutto quel che riguarda il C64. Ad esempio si trovano i datasheet originali del 6510 e dei vari chip (ma alcuni sono scannerizzati abbastanza da cani) e link a una miriade di siti con documentazione.

Ecco qualche esempio:
- *https://fms.komkon.org/EMUL8/HOWTO.html (non si può iniziare a scrivere un'emulatore senza aver letto l'howto di Marat Fayzullin!)*
- *https://www.c64-wiki.com/wiki/Memory_Map (dove è mappato cosa nella RAM del C64)*
- *https://www.c64-wiki.com/wiki/CIA (descrizione dei registri di CIA1/CIA2)*
- *https://www.c64-wiki.com/wiki/Page_208-211 (i registri del VIC-II)*
- *http://www.zimmers.net/cbmpics/cbm/c64/vic-ii.txt (un must-read per l'emulazione del VIC-II, la roba su c64-wiki non è troppo esaustiva)*
- *http://c64os.com/post/?p=45 (senza questo non sarei mai riuscito a capire la Keyboard matrix, necessaria per emulare il CIA1 e l'input)*
- *https://dustlayer.com/vic-ii/2013/4/25/vic-ii-for-beginners-beyond-the-screen-rasters-cycle (e questo mi ha aiutato a capire il concetto di raster!!!)*

DLM – Come hai deciso di emulare la CPU e come gli altri IC dedicati del C64 come il SID o il VIC-II?

VL - La CPU è implementata rappresentando un array di 256 elementi che rappresentano ognuno il puntatore all'implementazione dell'opcode comprensivo di Addressing Mode e numero di cicli che quell'opcode consuma (come ho spiegato prima, contare i cicli è essenziale). Il metodo step() prende un'istruzione dalla memoria alla locazione indicata dal Program Counter (PC), la esegue (il che aggiorna lo stato interno della CPU, ovvero tutti gli altri registri), incrementa il Program Counter, se necessario, e ritorna il numero di cicli occupati. Ci sono anche dei metodi irq() e nmi() che servono all'emulatore (o meglio, agli altri chip alla quale la CPU è virtualmente connessa) a triggerare in alcuni casi degli interrupt mascherabili (IRQ) o non mascherabili (NMI). Ad esempio quando i timer dei chip CIA 'scadono' (che termine brutto, in inglese sarebbe 'expire', non so come si possa tradurre meglio di così!) triggerano un IRQ (CIA1) ed un NMI (CIA2). Altro esempio è il famoso 'raster interrupt': quando il raster (il pennello elettronico che disegna lo schermo) raggiunge una certa scanline settata nel registro $d012, viene triggerato un IRQ. In questi casi, la CPU 'sospende' temporaneamente l'esecuzione del codice e 'salta' all'handler dell'IRQ alla locazione specificata dall'IRQ o del vettore NMI del 6510, dopodiché riprende da dove aveva interrotto. Riguardo agli altri chip, beh, come ho accennato prima, ho implementato degli handler di read/write che vengono chiamati a seconda degli accessi a certe locazioni, in base alla configurazione del PLA (in modo trasparente, tramite delle callback che v65xx mette a disposizione e che permettono di trappare read/write sulla memoria), più un metodo update() che viene chiamato durante l'esecuzione del mainloop.

DLM – Com'è andata con la gestione della memoria emulata? Quali difficoltà hai incontrato nello sviluppo di un modello che riproducesse l'accesso alla memoria del C64 reale?

VL - Quella è la parte più semplice, perché la rappresentazione della memoria è un semplice array. Con il metodo che ho utilizzato delle callback read/write nella CPU (in pratica, ogni istruzione che legge dalla memoria chiama la callback read e ogni istruzione che scrive nella memoria chiama la callback write) è abbastanza semplice gestire gli accessi. Ora che mi ricordo, parlando di accessi alla memoria, uno dei problemi iniziali che ho avuto (e c'ho messo un sacco a capire il perché!) è stato che avevo cannato totalmente l'implementazione del PLA! Così, tutto sembrava funzionare se il PLA era in configurazione standard (come al boot, con i registri del VIC mappati a $D000): anche alcuni giochi di Jeff funzionavano, compreso Gridrunner che mi ero proposto come primo obiettivo, quindi inizialmente non notai la cosa provando sempre i soliti giochi. Ma una volta che il PLA veniva configurato in maniera diversa, ad esempio avendo a $D000 mappate la ROM CHARGEN o la RAM), il display veniva ovviamente corrotto o crashava semplicemente perché al posto di leggere i registri del VIC si andava a leggere memoria a caso.

DLM – Quindi è stato semplice implementare le ROM di sistema del C64 (Kernal e BASIC su tutte) in base alle tue scelte progettuali iniziali?

VL - Assolutamente sì, semplicissimo perché non ho implementato proprio niente :). Essendo un emulatore, viene emulato l'hardware su cui gira il software. E il BIOS del sistema (ossia le ROM) è un software. In base alla struttura del mio progetto, basta caricare KERNAL, CHARGEN e BASIC in memoria alla locazione alle quali il sistema se le aspetta (è bastato consultare la Memory Map del C64 su c64-wiki), settare la CPU all'indirizzo contenuto nel reset vector del 6502 ($FFFC, che è la prima locazione che viene controllata quando si accende il C64) e infine far partire la CPU e quindi l'emulazione. Se tutto funziona, come per magia, apparirà la mitica schermata blu :). Tra parentesi, indovinate un po' che indirizzo c'è a $FFFC, quando viene caricato in memoria il KERNAL? Ebbene sì, $FCE2 = 64738. Ed ecco spiegato perché il C64 si resetta con SYS64738, semplicemente si chiama la routine (SYS) alla locazione 64738. Io l'ho scoperto dopo quasi 40 anni!

DLM – Come hai implementato i task ciclici come il refresh dello schermo, gli interrupt VBlank ed HSync, l'aggiornamento dei timer, il controllo della tastiera e delle porte joystick?

VL - Come ho ricordato nelle slide di OUAS, per la storia dei task ciclici e la sincronizzazione generale del sistema emulato, mi è stato di enorme aiuto prima leggere l'howto di Marat Fayzullin, che è stato uno dei primi a scrivere emulatori. Dopodiché, con l'aiuto di Giovanni, come ho detto prima, ho affinato la tecnica :). Ma veramente, la prima cosa da fare è leggere la guida di Marat. Scrivere un emulatore può sembrare quasi una magia, ma alla fine è più o meno come scrivere un gioco. C'è un mainloop come nei giochi, che gira finché non esci. All'interno di questo loop, si eseguono le istruzioni della CPU e si aggiorna di conseguenza lo stato dell'hardware. In modo ciclico si fa una lista di compiti ricorrenti. Tutto (si fa per dire) qui. La tastiera e il joystick sono un discorso a parte, per capire come viene gestita la Keyboard Matrix c'ho messo parecchio e c'avrei messo ancora di più se non avessi trovato la documentazione che ho linkato sopra, ma soprattutto senza questo post trovato su Lemon64.

Nel mio input manager (CInput) devo innanzitutto mantenere aggiornato lo stato premuto/rilasciato di ogni tasto nella keyboard matrix emulata (un array) ogni volta che si preme/rilascia un tasto sulla tastiera 'vera'. Dopodiché, nell'emulazione del CIA1 (al quale è connessa la tastiera) va gestito il fatto che il kernal, ciclicamente, attraverso le locazioni PRA ($DC00) e PRB ($DC01) va a ricavarsi lo stato di ogni tasto della tastiera scorrendo la Keyboard Matrix e usando PRA come maschera per accedere alle righe della matrice, ottenendo così il valore di PRB che corrisponde alle colonne. Per quanto riguarda il joystick, come ho spiegato anche nelle slide, questi sono connessi al CIA1 in parallelo insieme alla tastiera, quindi generano 'keyboard events' ambedue. Per il joystick in porta 1 è semplice: il kernel calcola lo stato del joystick in base a PRB e ad ogni tasto premuto corrisponde una direzione o il fire button. Per il joystick in porta 2 teoricamente dovrebbe essere simile (l'unica differenza è che il joystick in porta 2 è gestito in base a PRA) e corrispondere ogni input del joystick ad una combinazione di tasti invece che ad un tasto singolo. Probabilmente ho sbagliato qualcosa io nell'emulazione, perché provando quest'approccio non sono riuscito a farlo funzionare. Quindi, visto che non ne venivo fuori, ho usato un hack sporchissimo: lo 'converto' al volo in joystick 1 andando a scambiare PRA con PRB (quindi nel mio emulatore non potrai ad esempio mai fare un doppio a International Soccer! Poco male.. tanto si tratta di uno di quei giochi che non funziona - ahhahahaah). Et voilà, problema risolto.

DLM – Dopo quanto tempo dall'inizio dello sviluppo hai ottenuto un eseguibile in grado di far partire il primo semplice file .prg? E quanto questo ti è stato utile per velocizzare lo sviluppo vedendo ciò che non funzionava a dovere?

VL - Il primo programma che ho provato è stato ROX64 (ovviamente di Jeff Minter), scritto in BASIC. Lo scelsi proprio perché pensavo fosse più semplice da emulare!). E' partito subito ma per la storia del PLA che dicevo prima appariva così come appare nel mio tweet (e poi non avevo ancora implementato gli sprite, in ogni caso). Da lì in poi è stato tutto un progredire e fixare cose sempre col mio solito metodo 'trial & error' (specialmente finché non sistemai definitivamente la cosa del PLA, diciamo che cambiavo il bank switching

Foto 4 - Cena fra coder: Giovanni Bajo, Jeff Minter (Llamasoft), Valerio Lupi, Ivan Zorzin (Llamasoft)

a mano ogni volta per far funzionare qualcosa, non capendo bene che avevo sbagliato a monte!). Comunque, aggiungendo di volta in volta un mattoncino alla mia costruzione, in poco tempo, ROX riuscì finalmente a partire anche se zoppicante, poi Matrix per la prima volta (AMC 2, ancora Jeff), quindi il mio scopo iniziale era raggiunto! In pratica, continuai così fino alla fine: provavo a eseguire un gioco (iniziavo a provare anche cose non di Jeff, ovviamente) vedevo che non funzionava qualcosa, andavo a rivedere la documentazione, mi accorgevo degli errori, sistemavo, a volte rompevo qualcos'altro, a volte no e quindi si vedevano i miglioramenti. Naturalmente nel frattempo implementavo anche altre cose, come gli sprite e i modi bitmapped, quindi sempre più giochi iniziavano a partire ed essere giocabili.

DLM – Quale livello di completamento ha raggiunto secondo te vC64-Emu? Sappiamo che al momento non supporta il SID, i file .d64 o .tap e altro ancora, ma che è in grado di far girare molti giochi e programmi in formato .prg. Prevedi di continuare, hai avuto richieste in tal senso da coloro che ti hanno seguito durante lo sviluppo?

VL - Ti dirò... Non lo so ancora :). Considerando i bug propriamente implementativi di tante cose che ho capito solo a posteriori, che sono causa dei tanti malfunzionamenti attuali, della mancanza dell'emulazione del SID, del disk drive, del datassette, dei vari formati, ecc. rimane un sacco di lavoro da fare. Io direi che tutto sommato siamo vicini al 45-50%, non di più, ma per me è stata già una gran cosa. Come ho detto, sono partito COMPLETAMENTE da zero e non conoscevo molto del C64. Quando lo avevo ero solo un bambino e sinceramente pensavo di fermarmi e darmi per vinto molto prima di arrivare allo schermo blu iniziale :). Quando poi ho visto girare Hover Bovver, proprio il giorno che ho implementato gli sprite (anche se ci sono ancora molti bugs) [R2] una lacrimuccia è spuntata.

Riguardo al continuare lo sviluppo, beh, non è mai detta l'ultima parola. Veramente al momento sono sia impegnato in un nuovo lavoro, nella vita 'reale', che mi sta portando via molto tempo. Questa per me è una nuova avventura (finalmente ho smesso di fare lo sviluppatore per lavoro e ho intrapreso un percorso più 'manageriale', in effetti non ne potevo più, per me lo sviluppo dev'essere divertimento!!!) e poi preferirei dedicarmi a qualcos'altro nel campo dell'emulazione. Io volevo solo provare a me stesso (e sì, anche a qualcun altro in realtà) che ero capace di fare altro oltre alle cose che ho sempre fatto per lavoro. D'altro canto la competizione con VICE o altri emulatori è fuori discussione e comunque ho messo il sorgente da subito su GitHub appositamente per chi volesse contribuire, continuare, estendere, ecc. Insomma, il sorgente è lì, fateci quello che volete, magari offritemi da bere quando mi incontrate.

DLM – Hai fatto tutto da solo o hai cercato aiuto presso altri programmatori, magari già con qualche esperienza alle spalle riguardo alla programmazione di emulatori? Hai ricevuto suggerimenti e feedback da parte di fruitori o tester del progetto?

VL - Assolutamente tutto da solo! Certo, ho chiesto a Giovanni qualche dritta sulle cose che non mi tornavano. E me ne ha date anche dopo. In effetti diverse cose le rifarei diversamente ora. Il modo in cui le ho implementate è proprio sbagliato, motivo per cui alcune cose funzionano a metà, non funzionano o sembrano funzionare affatto ma in realtà sono inesatte. Non nego che quando sono stato preso dalla frustrazione ho guardato in giro come altri emulatori hanno implementato certe cose (ad esempio, le istruzioni SBC e ADC della CPU, o la gestione del raster

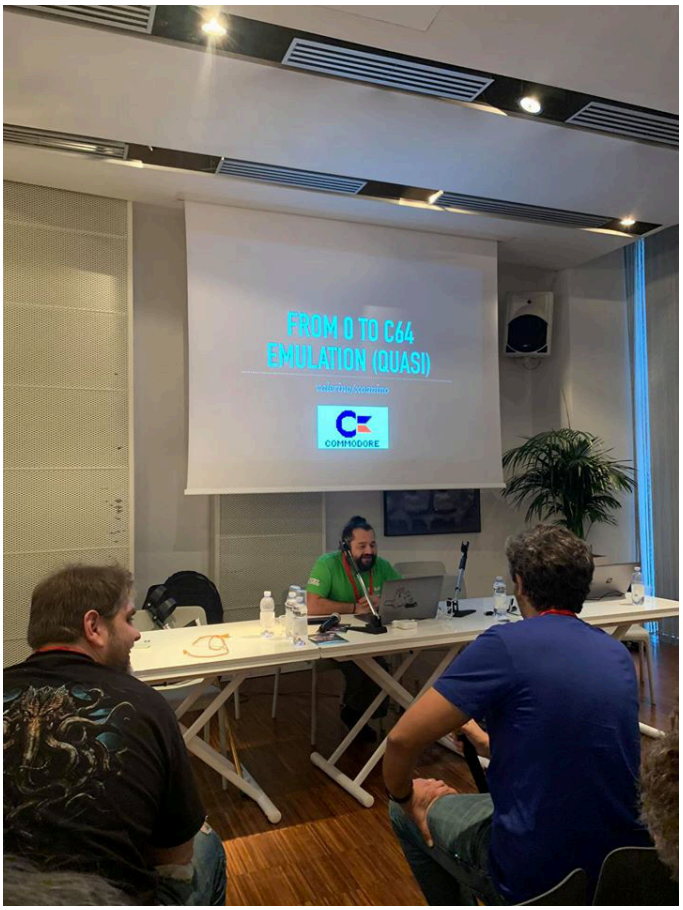


Foto 5 - Il talk su vC64-Emu al recente OUAS 2019.

interrupt che ho comunque sbagliato. Odio rimanere bloccato e alla fine non è un compito in classe, se perdo il divertimento e inizia la frustrazione è finita!). Ma assolutamente ho sviluppato tutto da solo, dalla prima all'ultima riga di codice.
Riguardo ai suggerimenti, beh, sì, ne ho ricevuti. Un ragazzo (ciao Marco, tra l'altro toscano come me!) mi ha scritto e mi ha detto che è riuscito a far girare l'emulatore sul Raspberry praticamente solo ricompilando ed è stata per me una grande soddisfazione. Mi ha anche informato di aver implementato un per ora rudimentale (senza emulazione reale del drive, quindi) supporto ai .D64! Appena avrà tempo, mi ha detto, mi farà la pull request su GitHub per integrare il tutto.

DLM – Qual è stato il più grande ostacolo tecnico/di programmazione che ricordi di aver affrontato durante lo sviluppo di vC64-Emu?

VL - Sicuramente l'emulazione del CIA1 per quanto riguarda joystick e tastiera. Ci ho messo veramente tanto a capire come funzionava, come ho detto prima. Anche l'implementazione del bankswitching, con il quale, una volta sistemata l'emulazione del PLA, molti più giochi hanno cominciato a funzionare. Poi sicuramente l'emulazione della gestione del raster nel VIC (che non sono troppo sicuro di averla ancora azzeccata bene): a mano a mano che la sistemavo, iniziavano a partire più giochi (ad esempio, se non si gestisce bene il raster interrupt, compreso settare il bit 7 di $D019 al momento giusto, Hover Bovver non parte). Per finire, la collisione degli sprite che mi ha fatto penare non poco. In realtà non funziona bene neanche ora, ma sicuramente meglio della prima implementazione: ora quantomeno Attack Of The Mutant Camels è giocabile, prima dovevi sparare al cammello 500000 volte prima di sperare che il sacro codice di Jeff riconoscesse che il cammello era stato colpito!

DLM – Sono certo che ha dovuto lavorare duramente su questo progetto, ma anche che ti sei divertito molto durante la fase di sviluppo e di testing. Qual è il momento più divertente o bizzarro che hai trascorso lavorando all'emulatore?

VL - Un momento divertente è stato sicuramente quando ho replicato, emulando il Koala Paint (il Photoshop di quei tempi), a modo mio la famosa scena di "Sono un fenomeno paranormale" di Alberto Sordi. E' una cosa solo per intenditori, l'originale è in [R3] , la mia versione è questa [R4].
Il momento più bizzarro credo sia stato quando ho eseguito su vC64-Emu l'emulatore di ZX Spectrum [R5].

DLM – Volgendo lo sguardo indietro nel tempo a quando hai iniziato quest'avventura, c'è qualcosa di cui ti penti o che faresti in modo diverso se potessi? Un diverso approccio oppure qualche dettaglio qua e là del progetto?

VL - Dunque, qualcosa di cui mi pento, no, assolutamente. Perché mai? Alla fine ho imparato tantissimo! :) Certo, ora se dovessi ripartire, rifarei le cose diversamente, anche alla luce di quello che mi ha spiegato Giovanni (ad esempio, considerare il numero di righe che devo tracciare ed impostare il runloop in base a questo). Altro esempio, se ho due chip che funzionano uno a 1Mhz e uno a 2Mhz e devo disegnare 200 righe di schermo ogni frame, ad ogni iterazione (quindi for i = 0 to 200) emulo 1000000/200 cicli del primo chip, 1000000/200 cicli del secondo chip, e via dicendo. In questa maniera l'emulazione viene molto più precisa di come faccio io, perché i miei CIA1 e CIA2 funzionano per miracolo giusto perché funzionano a 1mhz e il 6510 del c64 (PAL) funziona "quasi a" 1Mhz. In realtà nella mia emulazione del VIC, conto ovviamente i cicli che devono passare tra una linea e l'altra, ma insomma, a parte che ho cannato completamente i CIA, come dice Giovanni è un approccio molto più chiaro del mio. Ma emulando macchine del genere, dove tutto era sincronizzato PERFETTAMENTE secondo un clock è ovvio che se sbagli la sincronizzazione tra i componenti anche di poco e otterrai comportamenti diversi dall'originale.

DLM – Quale nuova avventura, se ve n'è una, nel mondo del retrocomputing ti sei prefisso per questo 2020 o per gli anni a venire? Un uccellino mi ha detto che forse potresti lavorare ad un nuovo emulatore della console Atari 2600...

VL - Speriamo di trovare il tempo, ehhehehehh. Comunque sì, avevo iniziato a fare (stavolta in Rust, che odio profondamente!) un emulatore di Atari 2600. Al momento è on hold, non ci lavoro da mooooolto tempo,

avevo appena iniziato il rewrite di v65xx in Rust! Sì lo so, sono masochista! Perché l'Atari 2600? Beh, innanzitutto, David Crane (il fondatore dell'Activision e autore di Pitfall e di Ghostbusters su C64) è uno dei miei idoli insieme a Jeff (l'altro componente della mia personalissima trinità è Matthew Smith, autore di Manic Miner su Spectrum). E poi, proprio una sera che ero a cena con Jeff e Ivan quando vennero in Italia a Roma a inizio estate 2019, Jeff mi consigliò di leggere 'Racing the Beam', un libro che spiega lo sviluppo su Atari 2600.

Praticamente, non essendoci memoria video, sull'Atari 2600 non esiste neanche il concetto di risoluzione. Il programmatore disegna le scanline pixel per pixel cercando di stare 'al passo' col fascio di elettroni del tubo catodico che le visualizza, da qui 'racing the beam' perché appunto dovevi far tutto in tot cicli di clock altrimenti la TV passava alla linea successiva e ne disegnava solo una parte della precedente. Una roba veramente allucinante e affascinante allo stesso tempo :) Rimasi folgorato leggendo quel libro (si trova in pdf, basta cercare), e quindi l'Atari 2600 è diventato il mio secondo progetto di emulazione :). Staremo a vedere, spero davvero di trovare il tempo!

DLM - Grazie mille, Valerio, per aver dedicato un po' del tuo tempo alla nostra rivista. Sono certo che i nostri lettori apprezzeranno una voce così preparata ed un argomento che da sempre affascina gli appassionati di programmazione.

VL - Grazie a voi! Spero di essere stato esaustivo nelle mie risposte :). A presto! Ciaooooo!!!

Grazie alle puntuali e variopinte risposte di Valerio, abbiamo appreso molto su come si può progettare e programmare un emulatore con gli strumenti di coding a disposizione oggigiorno. Chiaramente questo non basta per lanciarsi in un'impresa tanto ardua. Servono soprattutto tempo, skill non comuni, flessibilità nell'uso degli strumenti di sviluppo e soprattutto, come ci ha palesemente mostrato Valerio, tanta tenacia, voglia di mettersi alla prova e una discreta dose di incoscienza!

Per adesso salutiamo Valerio ed i suoi progetti. Ma chissà che questo non sia solo un arrivederci. In futuro ci piacerebbe leggere di nuovo qualcosa sulle sue avventure spericolate nel mondo del retrocomputing.

RIFERIMENTI

- Slides su vC64-Emu (OUAS 2019)
- C64 Wiki https://www.c64-wiki.com/
- [R0] Articolo Lega Nerd
- [R2] Hovver Bovver gira su vC64-Emu!
- [R3] KoalaPaint "Sono un fenomeno paranormale"
- [R4] Versione di Valerio
- [R5] ZX Spectrum su vC64-Emu!

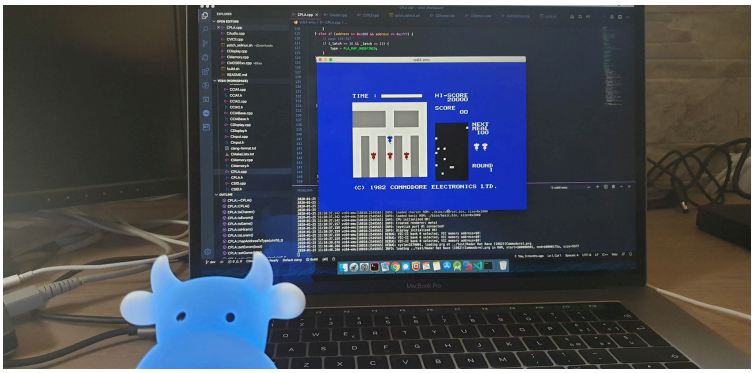


Foto 6 - Radar Rat Race (Commodore)

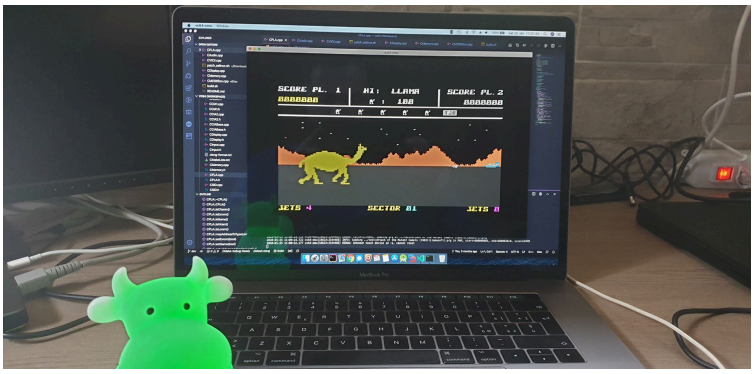


Foto 7 - Attack of the Mutant Camels (Llamasoft)

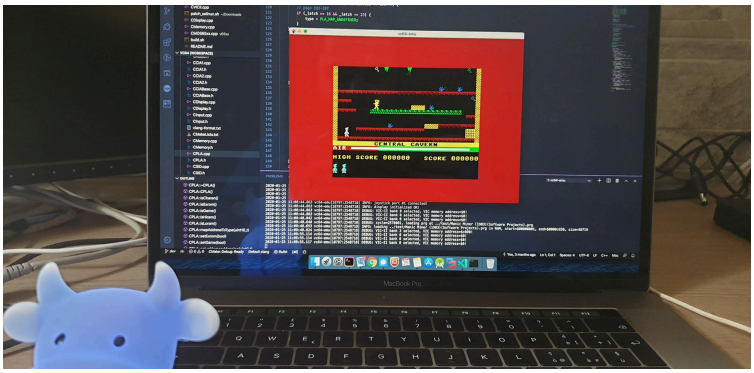


Foto 8 - Manic Miner (Ingame)

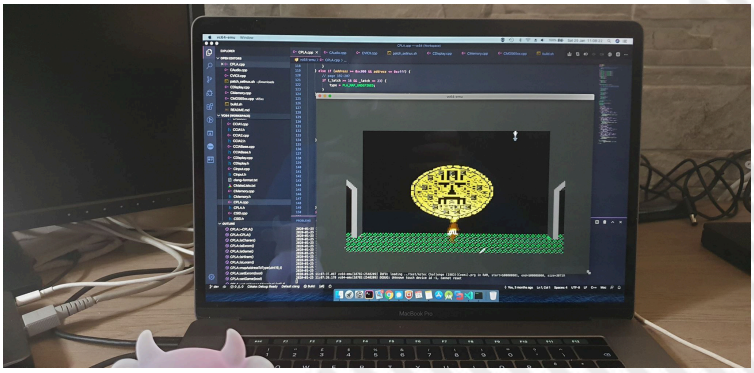
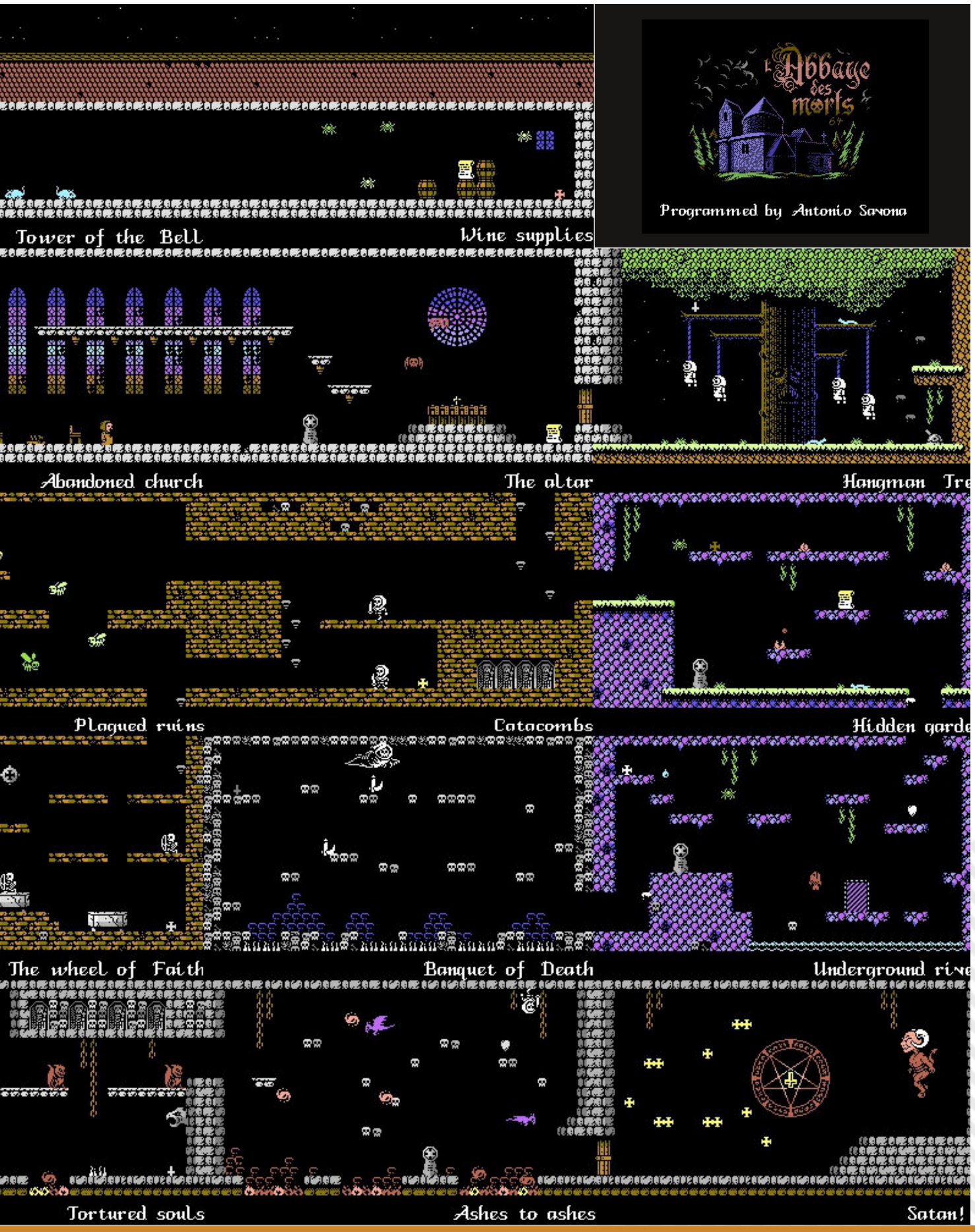


Foto 9 - Aztec Challenge (Cosmi)

L'Abbaye des morts

Nei primi giorni di dicembre, **Francesco Galluccio** fece un post su alcuni gruppi Facebook mostrando la mappa del gioco **L'abbaye Des Morts**, programmato per il Commodore 64 da **Antonio Savona**. L'immagine mi riporto' subito alla mente le mappe pubblicate negli anni 80 sulle riviste di settore e decisi quindi di contattare Francesco per avere il permesso di pubblicare il suo lavoro su RM.
Se l'idea vi piace, fateci pervenire le vostre mappe e le pubblicheremo!



NES MAKER
MAKE YOUR OWN GAME

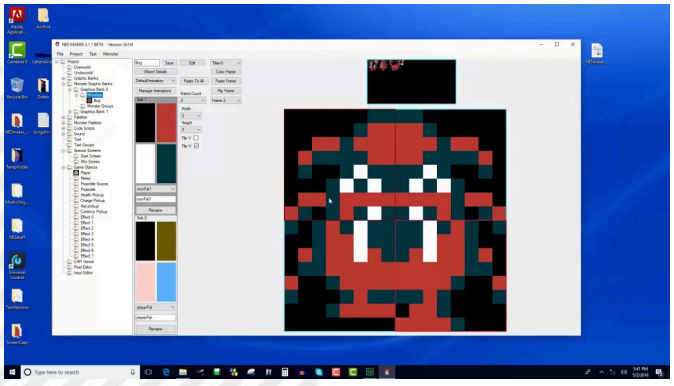
Editore: The New 8-bit Heroes
Sito web: http://beta.thenew8bitheroes.com/

L'amore per il mondo degli 8 bit è popolato da eroi.

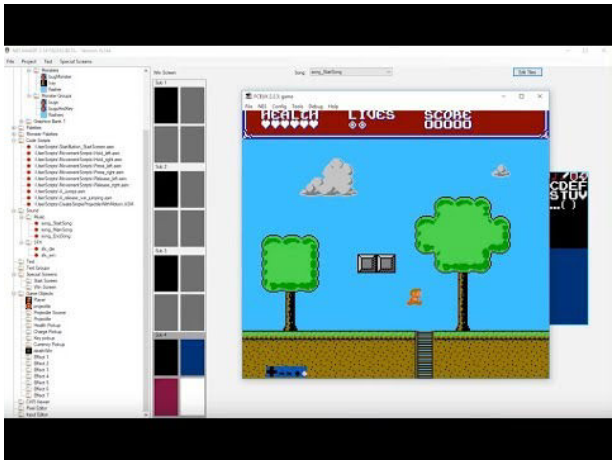
Gli eroi in questione non sono quelli dei nostri amati videogames, ma i New 8-Bit Heroes, piccola software house che ha sviluppato questo bellissimo prodotto per Pc.

Il Nes Maker nasce come progetto su Kickstarter, un progetto che non presenta solo un software su pc dove poter sviluppare il proprio personale "gioco NES", ma come vero e proprio sistema di sviluppo a 360°.
Il successo è immediato! In poco tempo sul mercato escono diverse versioni ed ora andiamo a visionare la versione 4.15.

Il software è di semplice utilizzo esattamente come il più famoso RPG maker, permette attraverso una serie di menù grafici di creare giochi su PC senza avere basi di programmazioni e di coding. Offre quindi la possibilità di creare il proprio gioco ad 8 bit in perfetto stile "NES" e di flasharlo su una cartuccia per poi poterlo giocare direttamente sul piccolo 8 bit di NINTENDO. Un'esperienza che per gli amanti del retrogaming è davvero unica.



Si possono sviluppare diversi tipi di giochi, dai platform agli action adventure, passando per i classici jrpg alla Zelda, senza troppi sforzi. Tutto attraverso utili menù drag and drop.
Si potrà quindi progettare grafica, sprites e layout di colore ovviamente vincolate dallo stile grafico del Nes. Lavorare su un editor di testo che permette di creare stringhe di testo per gli NPC.
Creare il menù iniziale, schermate di presentazioni o di termine gioco, menù mobili, mappe e tutto quello di contorno in un gioco.
Personalizzare l'AI degli NPC
Impostare i parametri per lavorare in playtest su emulatore via PC prima di "flashare" il gioco.
Utilizzare un motore base per partire da progetti "pre impostati" dove possiamo lavorare su giochi di avventura, rpg, platform, picchiaduro e altro ancora.
Infine tramite un click e attraverso un sistema USB pensato apposta, flashare su Cartuccia e … giocarlo su NES.
Insomma Nes Maker è davvero un prodotto accattivante e di facile utilizzo.



Lo sviluppatore è Joe Granato, che lo ha creato con l'idea principale di sviluppare il suo videogame Mystic Seach, gioco di ruolo alla Legend of Zelda.
Nello sviluppo inziale ha pensato di lanciarsi in qualcosa di più "aperto" e ha pensato di rendere il prodotto una specie di "PHOTOSHOP" per NES (parole sue).

Il modulo "Avventura" è quello che ho visionato con maggiore attenzione. Si tratta del modulo più classico che ci permetterà di creare un'avventura alla "Zelda" o "Final Fantasy", anche se con un po di pratica può essere utilizzato e manipolato per permetterci di fare qualsiasi cosa. Possiamo decidere di rendere le quest più o meno complesse sviluppando per loro una difficoltà graduale. Possiamo decidere di inserire eventi random, incontri casuali oppure sviluppare il gioco alla "BALDUR'S GATE" inserendo la possibilità del secondo player e gestendo assieme ad un amico il gruppo di gioco.

Le limitazioni del NES MAKER?
La principale è quella più ovvia, puoi creare solo ciò che è disponibile all'interno del programma stesso. Sei sei un

programmatore con esperienza nel mondo NES con tonnellate di conoscenza del sistema, sicuramente troverai limitante questa cosa.

Nell'ultima release è stata inserita la possibilità di modificare manualmente il codice, ma sarà quasi come decodificare un gioco NES, un lavoro bestiale già di per sé.
Tuttavia, se sei come me un appassionato Hobbista e vuoi creare qualcosa da Zero, ma la codifica è la tua bestia nera.. il prodotto è perfetto!!!

Va detto anche che sul sito degli sviluppatori sono presenti decine di tools video dove ci viene spiegato da 0 ogni singolo passo per realizzare il proprio games. Uno dei tutorial più impressionati è quello dove realizzare in 20 minuti il proprio platform.

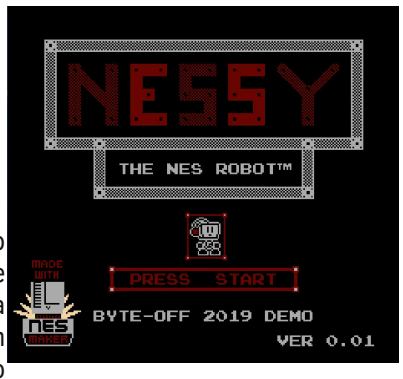
Parliamo della parte "corposa".. ovvero lo sviluppo della cartuccia.
Il prodotto viene venduto sul sito in versione liscia a 36$ e in versione con Kit Hardware a 88$. Quest'ultima ci presenta il software, l'UsbINLRetro con connettore nes, 1 cavo usb da 30 cm, una cartuccia Nes, un mapper Nes da 72 pin progettato per il Nes maker. Attraverso tutto questo è possibile, una volta terminato e testato il nostro prodotto, cliccare su "flash rom" e copiare tutto sulla cartuccia. Ogni singola cartuccia nes costa 20 dollari ed è possibile acquistarne di supplementari sul sito degli 8Bit, su Amazon o su Alibaba. La cartuccia una volta flashata è nativa ntsc, ovvero nasce per sistemi americani ma con un paio di accorgimenti (diversi reset della macchina una volta accesa per poter settare la regione) si riesce a farla girare sui sistemi pal.
Vi consiglio, se siete appassionati e volete cimentarvi in qualcosa di diverso, questo prodotto. Gli sviluppatori ogni 2 mesi circa presentano corposi aggiornamenti e settimanalmente sviluppano tutorial appositi.

Non ci resta che metterci a lavoro!! ;)

Per chi volesse vedere alcune demo o interi prodotti creati con NESMAKER consiglio di visionare il seguente sito.
http://vizivasoftware.com/nes/.

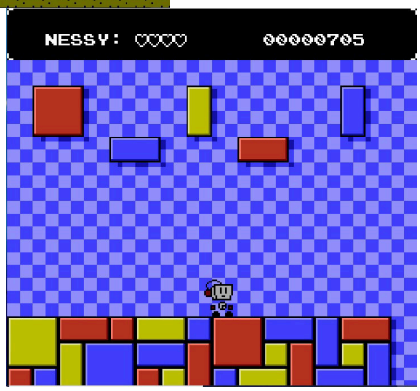
Da questo sito ho visionato due giochi che andrò a presentarvi: Nessy the NES Robot e Dodle Land Demo, due ottimi platform. Vediamoli!

Nessy the Nes Robot



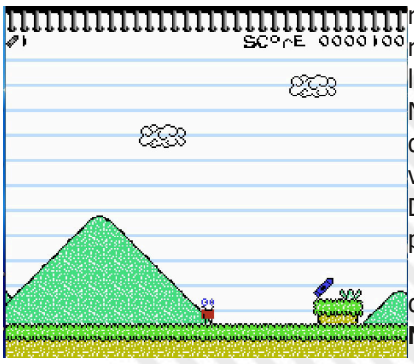
Sviluppatore:
Byte-off 2019
Fase di sviluppo:
demo di 3 livelli

Si tratta di un prodotto molto ben fatto, dove guideremo Nessy una "cartuccia" nes in un mondo coloratissimo sulla falsariga dei più classici platform game a scorrimento. In questo demo di fine 2019 abbiamo un prodotto già completo. Questa versione si sviluppa su tre livelli. Il protagonista può saltare sopra gli avversari e schiacciarli nel classimo sistema alla "super Mario" oppure distruggerli con il classico beam a corto raggio.
Molto colorato e con una discreta AI degli avversari è uno dei prodotti più interessanti sviluppati con il NESMAKER.



Dodle Land Demo
Sviluppatore:
Nate Peters
e Anaceli Peters
Fase di sviluppo:
Un demo di tre livelli con boss finale.

Un piccolo gioiello dalla grafica semplicissima concepito come fosse l'album da disegno di un bambino.
Il protagonista è una gomma da cancellare che, nel classico mondo platform a scorrimento, deve raccogliere dei pastelli colorati e affrontare dei nemici di vario tipo (pastelli, formici, carta da disegno, puntine).
Non mancano le warp zone dove raccoglieremo bonus per far punti o livelli nascosti.
Ciò che colpisce di più è la grafica. Semplice ma davvero accantivante, sembra davvero l'album da disegno di un ragazzino. Utilizzato molto bene anche l'editor sonoro del NesMaker. Speriamo di vederlo presto in versione "finale". Davvero un gran bel prodotto.

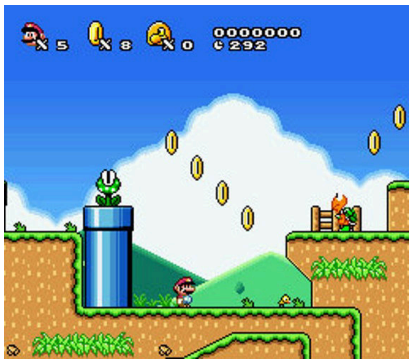


di **Carlo Nithaiah Del Mar Pirazzini**

NEW SUPER MARIO WORLD 2: AROUND THE WORLD

Anno: 2019
Sviluppatore: Pink Gold Peach
Piattaforma: Super Nintendo
Genere: Platform

Super Mario World è universalmente riconosciuto come uno dei migliori platform game a 2d di sempre. Bello, ben bilanciato, curato sotto ogni punto di vista.
Negli anni, numerose hack hanno permesso a diversi appassionati di cimentarsi espandendo il mondo di Super Mario World. Alcune Hack sono davvero bizzarre, altre semplicemente geniali.. Questa che sto per recensire è davvero una bella scoperta.



Questa hack mode nasce come sequel di New Super Mario WorldThe 12 Magic Orbs, bel titolo molto amato dai giocatori che vedeva Mario in una ricerca di 12 sfere per il mondo dei funghi.
Around The World nasce dopo svariati anni di sviluppo (dal 2016 al 2019), vede Mario viaggiare tra diversi mondi (16) con lo scopo di salvare la solita principessa Peach. Ogni mondo è sviluppato in 5 o 6 livelli con una varietà di temi e di level design differente per un totale di 90 piccoli gioiellini da esplorare.
Ci sono come in ogni Mario che si rispetti una marea di segreti, camere nascoste, bonus e uscite nascoste che aumentano la longevità e la voglia di esplorare il prodotto.

Dal punto di vista grafico è davvero molto ben fatto. C'è tutta la bellezza di Super Mario World e tantissime chicche meravigliose che scoprirete livello per livello.

Il gioco è nel complesso davvero difficile, parte con un piccolo tutorial dove capiremo come gestire Mario per poi cominciare a mostrare la rudezza di alcuni livelli.
La rom è scariabile gratuitamente dal web (fino a quando Nintendo lo riterrà opportuno) ed è possibile giocarla su emulatore, Snes Mini tramite hakchi oppure su SNES originale flashandola su Card.

Non posso che consigliarvi il gioco. Un lavoro davvero impressionante che dimostra come la scena del Super Nintendo e dei suoi appassionati sia più viva che mai.

di **Carlo Nithaiah Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 90%

Come tutti i Super Mario, anche questa versione presenta lo stile classico della serie. Moltissimi livelli, bonus stage, la presenza di Yoshi, stage nascosti... Insomma in classico.
Il livello di difficoltà punta verso l'alto. Non è concesso molto spazio agli sbagli.

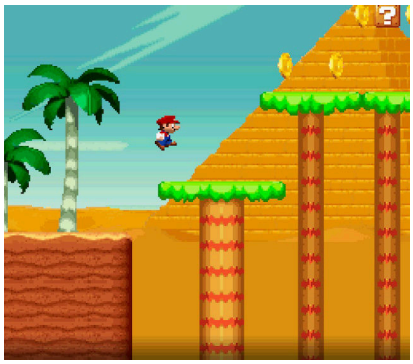
» Longevità 95%

Tantissimi coloratissimi livelli, bonus nascosti, mostri da spatasciare. Un grande grande platform.

NEW SUPER MARIO LAND

Anno: 2019
Sviluppatore: Anonimo
Piattaforma: Super Nintendo
Genere: Platform

Sviluppato recentemente per Super Nintendo e chiamato New Super Mario Land, è apparso in rete grazie al lavoro di un anonimo sviluppatore. Quando ho letto la prima volta che si trattava di un remake dell'originale Super Mario Land che usci nel 1989 per GameBoy, con una nuova grafica in stile Nintendo Ds e animazioni straordinarie, oltre al supporto per 4 giocatori, pensavo fosse una fake news e che il progetto si arenasse in un nulla di fatto o in una demo eterna. Invece sono stato fregato! Disponibile in download gratuito ed è SPLENDIDO!



Spesso vediamo nuovi giochi homebrew rilasciati per vecchie console come NES e Sega Megadrive / Genesis ma non è così comune vedere nuovi giochi SNES, apparentemente è un po più difficili da sviluppare e proprio per questo New Super Mario Land è strabiliante. Chiunque sia dietro questo gioco è estremamente talentuoso e ha dedicato moltissimo lavoro a tutti i dettagli.

Il gioco è semplicemente meraviglioso con uno stile grafico tratto dalla nuova serie di Super Mario bros che conosciamo dalla Wii o, come dicevo, dal Nintendo DS.

Vediamo anche molti dettagli negli sfondi con grandi effetti di scorrimento e riflessione dell'acqua di parallasse e molto altro ancora!

Il gioco supporta anche fino a 4 giocatori con un multitap! Tutto quello che devi fare è premere start su uno degli altri controller per unirti al gioco e il personaggio appare in una bolla proprio come nei giochi Wii... Ogni personaggio è caratterizzato dai colori di Mario, Luigi, Wario e WaLuigi. In cooperativo è divertente l'idea di lavoro in team sfruttando la possibilità di rimbalzare sopra gli altri per raggiungere le piattaforme nascoste o più elevate. Che dire, gioco consigliatissimo e divertente. Bilanciato e molto molto longevo come l'originale e anche di più grazie al modo cooperativo.

Il gioco è stato testato e funziona alla grande usando ed emulatore o una cartuccia flash su un vero SNES. Io l'ho caricato su cartuccia flash e su Snes Mini.. e in entrambi i casi fila liscio come l'olio.

Il file Rom è disponibile per il download qui: https://archive.org/details/newsupermarioland assicurati di scaricarlo prima che venga rimosso :)

Buon divertimento

di **Carlo Nithaiah Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 90%

Un classico per gameboy e un nuovo classico per Snes. Ottimi controlli e grande gameplay. Difficile ma mai punitivo, si sbaglia solo se si è disattenti.

» Longevità 95%

Questa versione su SNES ha la possibilità di essere giocata con da 1 a 4 giocatori in contemporanea su schermo. In singolo è sempre splendido ma in quattro diventa frenetico. In questa modalità si può cooperare per scoprire sempre più cose. Questa cosa allunga l'avventura davvero moltissimo.

BATTLEMORPH

Developer: Attention to Detail
Anno: 1995
Piattaforma: Atari Jaguar CD
Genere: Sparatutto in prima persona

Dopo una lunga assenza, torna la rubrica dedicata ai giochi dimenticati; per festeggiare il ventesimo numero, non potevamo non parlare di qualcosa di davvero particolare. Oggi affrontiamo infatti uno dei migliori titoli disponibili per l'oscuro Atari Jaguar CD, periferica sfortunatissima che ormai vive nella memoria di pochi. Prima di iniziare, facciamo un piccolo excursus storico.

Siamo nel Settembre del 1995; Atari sta provando in ogni modo a supportare il Jaguar, ma la situazione ormai è piuttosto complicata. La concorrenza non ha lasciato alcun scampo al felino, che si trova ormai in bilico. In un ultimo disperato tentativo di migliorare la situazione viene rilasciato il tanto atteso Jaguar CD, che promette di restituire alla console una buona fetta di pubblico. Purtroppo ormai è già tardi; la periferica viene prodotta in quantità molto limitate e vede il rilascio di soli quindici giochi, prima che Atari stacchi la spina all'intero progetto Jaguar.
Tra questi titoli ce ne sono alcuni piuttosto interessanti, in particolar modo Battlemorph della Attention to Detail, uno sparatutto tridimensionale in terza persona.
Il gioco è in realtà il seguito diretto di Cybermorph, uno dei titoli di lancio del Jaguar, pubblicato alla fine del 1993. Pur essendo discreto, il capitolo originale non convince pienamente pubblico e critica delle capacità della nuova macchina Atari, ma nonostante ciò è scelto come portabandiera, in quanto viene venduto in bundle con la console. Il team di sviluppo, accolte le numerose critiche, si rimbocca le maniche e si mette al lavoro sul seguito, che arriva nei negozi nel Dicembre del 1995.

Angolo oscurità

Dopo una lunga assenza, torna la rubrica dedicata ai giochi dimenticati; per festeggiare il ventesimo numero, non potevamo non parlare di qualcosa di davvero particolare.

Battlemorph è ambientato trent'anni dopo Cybermorph; l'impero alieno di Pernitia è stato ricacciato nel proprio sistema dalle forze coloniali terrestri. La scomparsa improvvisa di alcune navi da guerra nei pressi della stella Perseo, insieme ad avvistamenti di vascelli pernitiani, causa preoccupazione e scompiglio nelle colonie sparse per la galassia. Il "Consiglio della Difesa" decide quindi di inviare l'incrociatore Sutherland per far luce sulla situazione; a bordo del vascello c'è il War Griffon, un caccia mutaforma erede del T-Griffon del gioco originale. La situazione volgerà ovviamente subito in peggio e lo scontro con le forze aliene sarà inevitabile.

Graficamente Battlemorph mostra notevoli passi avanti rispetto al predecessore: il gioco presenta livelli molto più complessi, uniti ad una rinnovata veste grafica, con strutture molto più elaborate, maggiore fluidità e la sporadica presenza di texture mapping. La profondità visiva non è ancora una volta particolarmente impressionante, ma risulta migliore dell'originale ed è ammorbidita dalla sfumatura progressiva degli oggetti in lontananza. La presenta di coreografiche sezioni subacquee, direttamente raggiungibili dalla superficie senza alcun caricamento di sorta, unita ad alcune sequenze all'interno di tunnel, dona al tutto un'accattivante impatto estetico. Nota di merito anche alle sequenze video d'intermezzo, ben fatte e mai intrusive o noiose.

A livello audio troviamo una notevole colonna sonora composta da Will Davis, che si sposa perfettamente con l'atmosfera di gioco. Gli effetti sono ben realizzati ed il tutto è accompagnato dall'ottima narrazione vocale di Rob Brydon, unita al ritorno di Victoria Lowe nei panni di Skylar, l'intelligenza artificiale del nostro caccia. Questa volta la suadente voce del computer non sarà più pedante e ripetitiva come nel prequel, ma risulterà anzi particolarmente utile, in quanto ci fornirà informazioni dettagliate sugli obiettivi di missione.

Battlemorph è uno sparatutto in terza persona; dovremo completare una serie di missioni sui vari pianeti controllati dalle forze pernitiane. A differenza del prequel, che era incentrato sulla monotona ricerca di capsule, qui ci sono obiettivi molto più vari ed accattivanti, come ad esempio distruggere un treno in corsa.
La presenza di sezioni subacquee e di tunnel sotterranei riveste un ruolo fondamentale in molte missioni ed aggiunge profondità all'intera esperienza. Potremo inoltre armare il nostro caccia con diverse armi speciali e starà a noi capire quale si adatterà meglio al nostro stile di gioco.
Il sistema di controllo prevede un minimo di curva d'apprendimento, ma in breve tempo saremo perfettamente in grado di cavarcela. Il pad del Jaguar si comporta molto bene anche in questo caso, senza alcun tipo di problema. Nel caso abbiate un Pro Controller, ovviamente, le cose migliorano ancora di più, dato che la disposizione dei tasti è senza dubbio più azzeccata per un gioco di questo tipo.
Il titolo è inoltre piuttosto lungo e prevede ben quarantotto missioni, suddivise in otto sistemi con sei pianeti ciascuno. La presenza di vite extra nascoste nei livelli e di tre diverse opzioni per la difficoltà tra cui scegliere, permetteranno a chiunque si avvicini una sfida equa per le proprie capacità.

Tirando le somme, Battlemorph è senza dubbio un titolo davvero pregevole, che purtroppo ha sofferto dell'essere rimasto esclusivo per la periferica CD del Jaguar. Se siete tra i fortunati che ne possiedono una o avete i soldi per prenderla (beati voi), questo gioco è assolutamente degno di essere recuperato. Per tutti gli altri curiosi, speriamo in una futura emulazione della periferica, altrimenti difficilmente sarà recuperabile.

Alla prossima!!!

di **Federico "Arzak1" Gori**

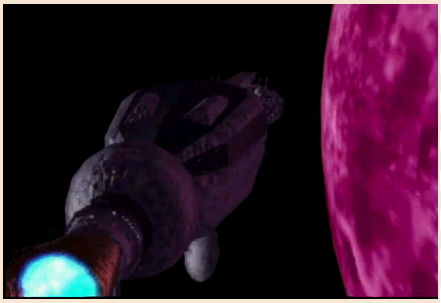
GIUDIZIO FINALE

» Giocabilità 85%

Divertente ed impegnativo, riesce a tenere sempre alta l'attenzione del giocatore grazie alla varietà di missioni disponibili.

» Longevità 80%

La durata dell'esperienza è particolarmente consistente. Non lo finirete tanto presto.



SUBTERRANEA

Anno: 1983
Sviluppatore: Imagic
Piattaforma: Atari 2600
Genere: shoot'em up

Prodotto nel 1983 dalla Imagic per mano di Mark Klein, Subterranea è un veloce e frenetico shoot'em up per Atari 2600, modellato sulla meccanica di combattimento di Defender.

Alla guida del Cave Ranger, un velivolo da guerra progettato per operare in ambienti bui e con poco spazio di manovra, dobbiamo raccogliere quanti più cristalli possibile, inoltrandoci sempre più nelle viscere di un non meglio specificato pianeta.

Detta così sembra facile, peccato che il sottosuolo sia costituito da una rete di tunnel infestati da veloci e letali automi volanti da combattimento, che collegano un complesso di caverne in cui possiamo ottenere le gemme di cristallo.

Una volta raggiunta una caverna, per ricevere l'agognata ricompensa, dovremo combattere con un boss posto a difesa del passaggio al prossimo sistema di tunnel, nonchè legittimo proprietario delle gemme: il mostruoso Hexuplex, che in verità a noi appare piuttosto grazioso con quelle sue zampette estensibili su cui si muove.

Il gioco inizia all'ingresso del mondo infero, dove siamo costretti fin da subito confrontarci con l'Hexuplex, il quale ha giustamente pensato di mettersi a guardia dell'accesso ai sotterranei. Questa colorata creatura, ottimamente animata per gli standard dell'Atari 2600, ha la deprecabile abitudine di scatenarci contro una serie di macchine volanti, gli Aerobot, particolarmente veloci, elusive e determinate nel cercare l'impatto con il nostro velivolo, nel chiaro intento di farci desistere dal sottrargli i suoi cristalli. Per fortuna, dobbiamo affrontare solo un Aerobot per volta e, per nostra ulteriore buona sorte, l'Hexuplex dispone di un numero limitato di Aerobot ed esaurita la scorta, fugge via per aspettarci alla fine del prossimo livello, lasciando sul campo una gemma che, una volta raccolta, incrementa sensibilmente il nostro punteggio. Come avrete sicuramente capito, in questo gioco il boss è sempre e solo uno che non può essere ucciso ma solo costretto a rifugiarsi sempre più giù, nel profondo del pianeta. In effetti non ci battiamo davvero contro di lui, visto che non possiamo colpirlo in alcun modo, la lotta si svolge unicamente tra noi e gli Aerobot. Ad ogni buon conto, raccolto il cristallo viene aperto un passaggio verticale che ci conduce ad una sequenza di tunnel da attraversare prima di giungere alla successiva caverna. Qui dovremo evitare di urtare contro le pareti, pena la perdita di punti per tutta la durata del contatto, contro i nemici volanti che infestano il tunnel ed i teschi che sporgono dal tetto e dal pavimento per non distruggere il Cave Ranger in uso. Il gioco inizia con una dotazione di quattro velivoli ma possiamo guadagnarne altri fino ad un massimo di sette, come vedremo più avanti. Nella parte alta dello schermo viene indicato sulla sinistra il numero di Ranger a nostra disposizione e sulla destra il punteggio raggiunto. Nella parte bassa invece, una serie di quadratini tiene il conto di quanti cristalli abbiamo raccolto, mentre una barra che decresce mostra quanti nemici restano da abbattere prima di passare oltre. L'azione qui si svolge in senso orizzontale, potendo percorrere il tunnel verso destra o si-

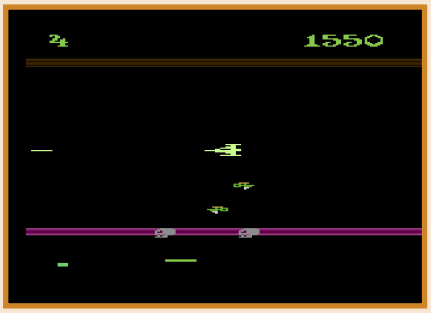
GIUDIZIO FINALE

» Giocabilità 85%

È uno shoot'em up per Atari 2600, cosa vi aspettate? Scansa, spara, fine. L'unica variazione sul tema, è rappresentata dagli Electro-Gate, un po' pochino...

» Longevità 95%

...ma dopo aver fatto 3 o 4 partite, diventa difficile rinunciarci se amate il genere. Un livello di sfida ben calibrato ed un'azione frenetica vi faranno venire sempre voglia di rigiocarlo di tanto in tanto.



nistra (ha una struttura ad anello). Ripulito l'ambiente dalle minacce volanti che lo popolano, comparirà da qualche parte un Electro-Gate, un cancello dotato di sensori ad impulsi luminosi che dobbiamo raggiungere ed attraversare per poter accedere al prossimo tunnel. I primi Electro-Gate sono facili da superare, ma i successivi sono più complessi e capaci di generare sequenze di impulsi più articolate; osservarne l'ordine di attivazione e prendere il ritmo per sgusciare tra un raggio e l'altro è di vitale importanza per superarli con successo. Se si viene colpiti da un impulso o se si toccano le cellule ai lati dell'Electro-Gate, il velivolo viene riportato fuori dal cancello e dovremo tentare di nuovo di attraversarlo. Fallendo per tre volte l'obiettivo, si provoca la chiusura dell'Electro-Gate e la ricomparsa dei nemici nel tunnel ma, cattiveria delle cattiverie, la loro distruzione stavolta non ci fa accumulare nessun punto! Come ulteriore nota sadica, una volta attivato il cancello, il punteggio inizia a decrescere, arrestandosi solo dopo che lo si è attraversato, così da spingervi a passarlo nel minor tempo possibile per non perdere troppi punti, ma in questo modo il rischio di fallire i tre tentativi a disposizione aumenta e ricominciare il livello daccapo (senza guadagnare nulla) diventa molto più probabile...
Superato un numero di tunnel che varia a seconda del livello di gioco, si arriva ad un'altra caverna a schermata fissa, ossia potete muovervi solo entro i limiti dello schermo, dove l'Hexuplex tenterà nuovamente di porre termine alla vostra scorreria. Man mano che si procede, gli Aerobot inviati contro di noi diventano sempre più numerosi e veloci e già dalla terza caverna dovremo impegnarci non poco per scacciare l'Hexuplex e sottrargli la gemma. Lo stesso vale per i nemici incontrati nei tunnel: diventano sempre più rapidi ed in alcuni livelli possono anche spararci contro, rendendo i nostri tentativi di incrementare il bottino di cristalli un compito sempre più arduo man mano che si avanza.
Il gioco non può essere vinto, si procede a ritmo sempre più serrato fino a quando l'ultimo Cave Ranger della nostra flotta viene distrutto. Alla fine, vedremo l'Hexuplex esultare per essere riuscito finalmente a levarci di mezzo, riconquistando così i suoi amati cristalli.

Punteggio

Abbattere un Aerobot: +100 punti.
Abbattere un nemico volante: tra +25 fino a +99 punti, a seconda del livello raggiunto nel gioco.
Raccogliere un cristallo: +1000 punti.
Impatto con le pareti del tunnel: decrementa il punteggio di una unità alla volta per tutta la durata del contatto.
Inoltre vengono sottratti punti nel lasso di tempo che intercorre tra l'abbattimento dell'ultimo nemico in un tunnel fino all'attraversamento dell'Electro-Gate, ad una velocità che cresce in base ai progressi nel gioco.

Superare un tunnel senza perdere un velivolo fa guadagnare un Cave Ranger extra, fino ad un massimo di 7.

Il contatto con gli Aerobot, le creature volanti o i teschi nei tunnel, causa la perdita di un Ranger.

Varianti

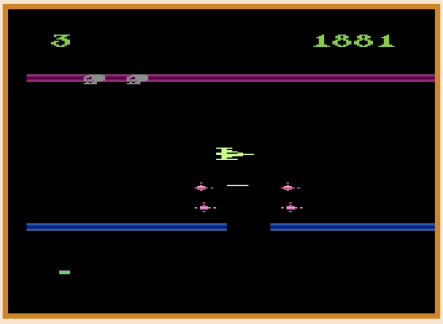
1 1: 1 giocatore, livello standard
1 2: 1 giocatore, livello avanzato
2 1: 2 giocatori, livello standard
2 2: 2 giocatori, livello avanzato

di **Giorgio Balestrieri**

Conclusioni

Subterranea è un gioco realizzato con maestria e che, nonostante la poca varietà dovuta alle limitatissime risorse della macchina su cui gira, è in grado di regalare ore di sfida e divertimento senza mai annoiare. I primi livelli sono relativamente facili da affrontare, ma presto le cose diventano decisamente complicate. Uno shoot'em up veloce e frenetico che vi costringe ad impegnarvi parecchio, punendo senza pietà ogni vostra minima distrazione. Se avete un Atari 2600 nella vostra collezione ed amate gli "spara e fuggi", questa è senz'altro una cartuccia che merita di essere recuperata e giocata.

Ah, dimenticavo: il gioco contiene anche un piccolo easter egg, che a distanza di tanti anni può ormai essere svelato. Vinta la sfida con il boss alla terza caverna, invece di raccogliere il cristallo che compare dopo l'uscita di scena della creatura, sparategli una 30ina di volte e lo vedrete trasformarsi nella coppia di lettere MK , iniziali di Mark Klein, il programmatore del gioco.



WONDERBOY

Publisher: Sega
Anno: 1987
Piattaforma: arcade
Genere: Platform-Rpg

Quando nel 1986 la Sega lanciò nelle sale giochi **Wonderboy** tutti noi rimanemmo colpiti da questo platform originale e dinamico. La possibilità di prendere uno skateboard e di muoversi in un mondo che ci veniva sempre incontro senza poter rimanere mai fermi erano caratteristiche che non si erano mai viste, per non parlare del protagonista Tom Tom, un biondo cavernicolo che non poteva non essere amato a prima vista.

Forse proprio per l'impossibilità di replicare queste dinamiche che la Sega, quando presentò il suo seguito, pensò ad un titolo completamente diverso, a metà tra un gioco di ruolo e un action game, un mix che ancora non si era mai visto nelle sale giochi. E così, soltanto un anno dopo, fa la sua comparsa **Wonderboy in Monster Land** che ha ancora un biondo personaggio come protagonista che questa volta si presenta con il nome di Shien.

Il nostro eroe, che all'inizio si trova letteralmente in mutande, si muove in un mondo fantasy popolato da strane e ibride creature. I livelli da affrontare sono ben undici e lo scopo del gioco è quello di eliminare un terribile dragone sputafuoco che minaccia tutto il regno.

Gli elementi tipici del platform ci sono tutti, quindi dobbiamo sempre muoverci e saltare su piattaforme, raccogliere monete ed evitare trappole di ogni tipo, ma accanto a questo gameplay classico, gli sviluppatori hanno aggiunto diverse novità.

Innanzitutto sono presenti i dialoghi tra i vari personaggi che permettono di spiegare e di dare un senso a tutta la storia.

Seconda importante novità è l'introduzione degli shop che si incontrano lungo il cammino all'interno dei quali è possibile spendere le monete raccolte per comprare pozioni curative, armi magiche nonché potenziare il player con scudi, spade, scarpe e armature varie.

Terzo elemento è la possibilità di raccogliere e inserire nel nostro inventario diversi item che aumentano la nostra forza danno, indeboliscono l'avversario, ci rendono per alcuni secondi invulnerabili o ci danno tempo aggiuntivo (il tempo è una vera e propria spada di Damocle lungo tutto il cammino) per terminare il nostro livello.
Come se ciò non bastasse gli sviluppatori della Sega si sono divertiti a nascondere nei vari livelli oggetti e passaggi segreti che alla fine risultano indispensabili per il successo finale. I nemici si presentano nelle forme più varie. Dovremmo vedercela contro funghi animati, pipistrelli, granchi e serpenti, mentre i boss che incontriamo nel nostro cammino hanno le sembianze di creature ibride, ognuna con proprie caratteristiche che dobbiamo ben memorizzare se vogliamo continuare nella nostra avventura.
Se poi per magia dovessimo riuscire a raggiungere il livello finale, ecco ancora l'ennesima sorpresa, perché da un mondo tipicamente fantasy saremo catapultati in uno futuristico e tecnologico e lo stesso dragone finale, dopo alcuni colpi, si trasformerà in un robot meccanico. Se riusciamo poi a sconfiggerlo e liberare la Wonder Land, saremo addirittura catapultati su un disco volante pronto per atterrare in nuovi mondi da liberare.

Wonderboy in Monster Land è senza dubbio un titolo affascinante ma anche ricco di difficoltà e forse proprio per questo nelle sale giochi non ha avuto un grande impatto poiché richiedeva l'inserimento continuo di monete.
Nelle console casalinghe invece il successo è stato totale, in particolare la conversione per il Master System ha fatto gridare al miracolo grafico oltre a presentare anche un livello e un boss aggiuntivo rispetto all'arcade. Così la Sega presentò due anni dopo, sempre per la stessa console, **Wonderboy III the Dragon's trap**, una nuova avventura che riparte proprio da dove terminava la precedente.
Ma parlando di conversioni non possiamo non citare quella per il c64, praticamente fedele all'originale e secondo alcuni il miglior titolo per il biscottone di casa commodore.

Con l'avvento del 16 bit la Sega ha pubblicato per il Megadrive altri due titoli che chiudono praticamente la saga.
Nel 1991 esce infatti **Wonderboy in Monster World** che per certi versi ricalca le orme del titolo arcade ma con la possibilità di salvare le partite direttamente nella memoria, senza usare lunghe password.
Infine nel 1994, ma soltanto per il mercato giapponese, esce **Monster World IV**. Questa volta la protagonista è Asha, una ragazza che si muove in mondi con atmosfere tipicamente orientali.

Querino Ialongo

GIUDIZIO FINALE

» Giocabilità 90%

Sarà per gli sprites simpatici e coloratissimi, per l'AI dei personaggi o per la possibilità di combinare elementi tipici di un platform e di un rpg, fatto sta che la giocabilità di questo titolo non ha subito per niente il passare del tempo.

» Longevità 90%

Quando uscì nelle sale giochi i tempi non erano ancora maturi per un rpg, non a caso questo gioco ebbe più successo per le console casalighe, perciò Wonderboy in Monster Land è ancora oggi un titolo tutto da giocare e scoprire.

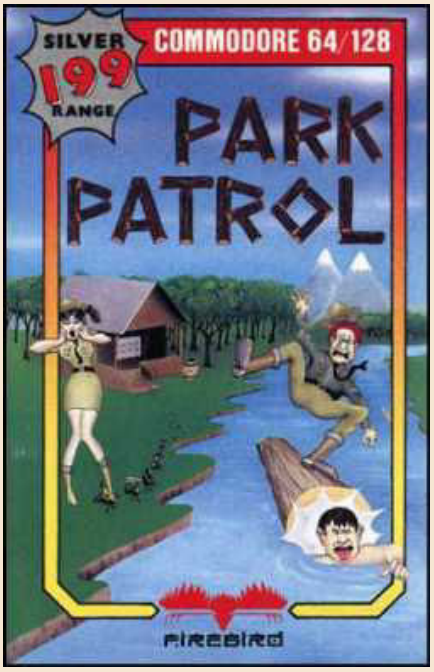
PARK PATROL

Anno: 1984
Sviluppatore: Activision
Piattaforma: C64
Genere: Azione

Primo mio articolo del 2020 e con questo voglio rinnovare il mio augurio per un sereno e proficuo anno a tutti voi che ci leggete ormai da qualche anno e che siete sempre più motivati, appassionati ed entusiasti grazie al retrogaming e retrocomputing che ultimamente stanno spopolando sui social network e nelle fiere, con macchine intramontabili e persone che nonostante sia passato un secolo, realizzano tuttora giochi per retro computer!

Certo, non come all'epoca, ma comunque un grande passo per delle macchine quasi abbandonate ed in piena epoca 64 bit. Nel il periodo natalizio, girando in rete ho visto un giochino singolare; Nemmeno il tempo di vederlo fino alla fine che ho deciso di presentarvelo sul primo numero dell'anno. Park patrol. In questo gioco impersoniamo un boy scout intento a ripulire il campo da bottiglie e lattine, oltre a salvare compagni che non sanno nuotare facendo attenzione alle creature della foresta e del fiume, tra cui le bisce d'acqua. Il gioco si svolge sia a terra che in acqua a bordo di un gommone e bisognerà prendere confidenza con esso per non fare errori quali finire a tutta velocità contro un legno ed altri ostacoli. Ciò che ha mi colpto particolarmente di questo gioco e' il senso civico ed altruistico che in parecchie città ormai manca. Fosse per me lo nominerei come gioco educativo. La musica e la grafica erano quelle dell'epoca dei primi giochi del Commodore 64 come anche la giocabilità, semplice e pura. Inizialmente si potrebbero perdere tutte le vite facilmente ma basteranno pochi minuti per prenderci gusto e mantenere pulita la nostra madre terra e pilotare il difficile gommone (oltre che salirci!). Per tenere lontani i nemici avrete una scorta limitata di colpi rappresentata da oggetti improvvisati; ma tanto i nemici ricompariranno lasciando presupporre che ce l'abbiano tutti con vo,i tranne i rifiuti che raccoglierete. Park patrol uscì direttamente in cassettina economica e se non vado errato erano pochi i negozi a possederlo; sembrava più un gioco da edicola e per questo probabilmente fu sottovalutato da molti. I soli cinque livelli lo facevano destinare ad un rapido archivio, stile fuoco di paglia, ma a mio modesto parere e' un gioco breve ma intenso, oltre che educativo. Immagino che alcuni bambini e ragazzi vedendolo all'epoca abbiano voluto almeno provare a fare il boy scout o il campeggio e spero abbiano dimostrato anche un bel senso civico. A livello di presentazione (io l'ho provato in versione disco) è quasi inesistente, tant'è vero che pensavo si fosse bloccato come tanti giochi all'epoca ...e i tempi di caricamento come ben sapete, su disco sono nettamente ridotti. L'unica cosa che potrebbe farvi salire i cinque minuti è quando cercherete di saltare sul gommone a riva, perché bisogna avere precisione millimetrica; non come quando ci buttavamo in mare o sul canotto gonfiabile da piccoli! Senza essere troppo ripetitivo, vi consiglio di giocare a questo gioco o almeno provarlo per poi magari condividere impressioni, giudizi e critiche costruttive dato che sono pochi ad averlo provato e merita sicuramente tutte le mia attenzione e spero anche la vostra. Con questo, vi auguro di nuovo un sereno anno dal doppio venti e che sia un anno ricco di novità Videoludiche!

di Daniele Brahimi



GIUDIZIO FINALE

» Giocabilità 70%
Una volta imparati i comandi sara' divertente!

» Longevità 70%
Pochi livelli, breve ma intenso! E non facile.

Giappone 8^ puntata: il salvadanaio di Targa!

di Michele Ugolini

Cari lettori, oggi parlerò dei treni giapponesi.
E' un argomento titanico. La domanda però sorge spontanea. Come mai in una rivista di Retrogame compare improvvisamente un articolo inerente al mondo nipponico su rotaia? Semplice, giacchè costeggerò questo profondo ed impetuoso mondo su rotaia, vi parlerò solo e soltanto di un loro simpatico e geniale giocattolo che hanno creato alcuni anni fa:

http://www.targa-japan.com/dengin/dengintop.html

Il Giappone, nel mondo, raggiunge numerosi primati riguardo svariati aspetti tecnologici.
Il discorso ferroviario (metropolitana, treni di infinite tipologie, super tecnologici Shinkansen, prototipi Maglev, filobus, monorotaia, la particolare Yurikamome, etc..) è sicuramente un mondo infinitamente vasto dentro il quale possiamo raccogliere pregi e meritocrazia realmente utili all'evoluzione dello scibile umano, al fine di risolvere piccoli e grandi problemi che attanagliano le rotaie dell'intero globo.
Il Governo giapponese coinvolge puntualmente, anzi, maniacalmente tutta la popolazione, attraverso una documentata analisi "in diretta" della situazione ferroviaria di tutte le loro isole, città, distretti e quartieri, sia dal punto di vista della puntualità del trasporto sincronizzando l'educazione e il rispetto a tale settore, sia, soprattutto, armonizzando l'approccio individuale alla popolazione stessa.

Adesso guardate questi due video su Youtube:

www.youtube.com/watch?v=OvkVMcNOhjg

www.youtube.com/watch?v=4V6Q5l2S7Co

Adesso invece visionate qualcuno dei link di questa raccolta di jingle del canale Youtube "tk's midi":

www.youtube.com/watch?v=Lgv6h7xb5gE

Avete ascoltato il bravissimo ragazzo, tutte quelle divertenti musichette e quell'adorabile nonché talentuoso bimbo che ogni tanto chiacchiera?
Cari amici che avete viaggiato in Giappone, anche a voi si sono inumiditi gli occhi?
Eh si, anche a me arrivano le lacrime quando ascolto quei jingle. Futuri viaggiatori, quelle infinite musichette vi stanno aspettando lungo una monumentale quantità di stazioni giapponesi. Sono lì, numerosissime, pronte per voi, ognuna profusa per aiutarvi, con una maniacale precisione al secondo, sebbene momentaneamente vi possano sembrare buffi jingle casuali dal gusto orientale!

Confusione? Perfetto! E' giunto il momento di spiegare tutto.

Vi avevo già abbondantemente avvisati che il Giappone è una terra intricata e l'unico modo per costeggiarla consiste nel permanere in un adrenalinico stato di costante apprendimento, non sto esagerando: provare per credere!

Il mondo giapponese su rotaia permea una realtà poliedrica coinvolgendo la popolazione sotto numerosi aspetti psicologici e sociali, che vanno dal puro divertimento dei bambini, fino alle sfere dell'alto business.

Affettivamente parlando per noi italiani, il nostro carissimo motorino e la nostra carissima automobile, rappresentano quasi una parte della famiglia. Altrettanto in Giappone, il mondo su rotaia, rappresenta un elemento indissolubilmente legato alle emozioni nonché al DNA di ciascun nipponico che spesso, in città come Tokyo, Kyoto, Osaka, Nagoya, non si sogna minimamente di acquisire la patente per possedere una inutile ed imparcheggiabile autovettura.
Pertanto il treno, in Giappone, scandisce la vita, le proprie emozioni, lo sviluppo della propria sfera psicosociale come la scuola, gli amori, il lavoro, i propri hobby, i propri amici, in maniera realmente viscerale ed a volte i giapponesi decidono in drammatica intimità che proprio tale mezzo di trasporto, ormai elevato ad "entità superiore", possa portargli via tutto. Purtroppo questa situazione è abbastanza frequente da scorgere nei tabelloni lampeggianti che indicano non ben precisati "problemi tecnici ad un treno", immediatamente rimpiazzato dal treno successivo. Questo avviene soprattutto nelle megalopoli, siate forti.

Dal momento che il mondo nipponico su rotaia pervade pressoché tutte le sfere sociali possiamo elencare numerose aree di relazione all'interno della collettività.
Esiste un area ludica che coinvolge scolaresche, gite, giocattoli, numerosi musei con treni storici o super tecnologici come il mitico prototipo JR-Maglev che ha stabilito i 603Km/h.
Esiste un area di alto business alla quale accedono unicamente i frettolosi e schivi businessmen che vedrete sfrecciare tra la folla per

gettarsi negli Shinkansen più rapidi del Giappone, i Nozomi N700 (300Km/h), Shinkansen ovviamente non usufruibili da noi turisti muniti di pass ferroviario turistico "JRP".
Esiste un vasta, puntuale ed accurata area mediatica/ informativa che allerta costantemente la popolazione al fine di evitare gli orari di punta delle rispettive linee ferroviarie e soprattutto i periodi dell'anno dove reperire un biglietto vanta probabilità simili alla lotteria.
Esiste un area femminile nel discorso ferroviario, cioè esistono designate carrozze, spesso di colore rosa, usufruibili unicamente da donne, questo per evitare problemi di sovraffollamento nel caso una donna sia in gravidanza oppure voglia truccarsi in tranquillità (atteggiamento abbastanza accettato dalla popolazione nipponica solo in questo ultimo ventennio) e soprattutto per evitare che qualche degenerato possa palpeggiare il fondoschiena di una malcapitata!
Esiste addirittura un area sociale prettamente dedicata ai maniaci notturni dei treni che, letteralmente, inseguono i treni super tecnologici che corrono spesso durante la notte: sto parlando di particolari Shinkansen utilizzati per raccogliere dati inerenti all'efficienza delle rotaie

Figura 1

percorse/esaminate, tramite i sofisticati strumenti a bordo. Il loro nome è: JR Shinkansen "Doctor Yellow".

Pertanto in questo articolo di retrogaming vi parlo di un gioco veramente carino e soprattutto educativo per i bambini giapponesi che ben presto dovranno affrontare il dedalo infernale, accennato sopra, delle infinite rotaie nipponiche.
Un dedalo maniacalmente organizzato, tanto che la media annua nazionale di ritardo su rotaia per ciascun treno equivale a 30 miseri secondi. Si esatto. Trenta secondi.
Potete immaginare quando nel Novembre del 2017, il capotreno, nella corsa programmata per le 9:44:40 (ora locale), ha fatto partire anticipatamente il treno alle 9:44:20? Non licenziato in tronco dalla Tsukuba Express (che opera da Tokyo a Tsukuba) ma tanti inchini del capotreno per chiedere scusa a tutti i passeggeri. "Inchini e scuse" ovviamente fotografati e pubblicati sui quotidiani della città di Tsukuba. E' andata peggio ad un suo collega capotreno, sempre in quello stesso periodo: licenziato per essere salito senza biglietto valido. Si, in Giappone i treni rappresentano un tema di massima serietà.

Figura 2

Quindi, come possiamo educare i bambini per un approccio funzionale a questo complicato e vasto mondo dei treni giapponesi?
Semplice, bisogna creare qualche gioco stimolante ed interattivo!
Allora, prendiamo un soldino, un salvadanaio, una musichetta accattivante, delle luci colorate, poi ci aggiungiamo altre musichette accattivanti, altre luci colorate, mettiamo tutto nel frullatore e serviamo la bevanda. Così nacque il treno/salvadanaio.

Svariati anni fa, questo gioco era diventato una vera e propria moda:

a prescindere dai collezionisti come me, veniva spesso regalato dai genitori ai loro bambini per depositare monete al suo interno.
Questo salvadanaio è la perfetta replica, in scala ridotta, della locomotiva di alcuni treni giapponesi.

In questo articolo parlerò unicamente del treno/salvadanaio più reperibile nei negozi di retrogaming, sebbene a prezzi ad oggi poco allettanti. (cfr. figura 1)
Il treno/salvadanaio più venduto era rappresentato dalla riproduzione della locomotrice di una linea "JR" di Tokyo: la Yamanote.
Facilmente riconoscibile poiché abbraccia quasi tutta la metropoli, attraverso un monumentale quantitativo di km di rotaie a "loop" attorno a Tokyo. Questo giocattolo è stato creato con ottima plastica robusta, con diversi pezzi assemblati e numerosi colori e serigrafie, una riproduzione realmente maniacale. Misure discretamente voluminose: un cubo di 15cm per lato.
Il tema della linea Yamanote accorpa tre colori: bianco, grigio scuro, verde acido.
L'aspetto delizioso di questo giocattolo è che sul tetto c'è una fessura per introdurre il soldino. Nel lato del cubo che dovrebbe congiungere le altre carrozze, invece, c'è una minuscola porta apribile che permette il prelievo dei soldi. E' inutile dirlo, tanto maniacalmente è stato costruito all'esterno, altrettanto maniacalmente sono stati riprodotti i sedili dei passeggeri e i vari elementi interni dell'abitacolo.

Ricordate tutte quelle musiche dei video? E' giunto il momento di parlare dell'aspetto più divertente del salvadanaio.
Una volta alimentato con 3 batterie AAA, introducendo il soldino dall'apposita fessura sul tetto, si accenderanno i fari e verrà emesso l'audio di un jingle, a caso, tra quelli registrati nella scheda di memoria.

Ovviamente esistono vari tipi di questi salvadanaio, (cfr. figura 2) vi allego dei link:

https://www.youtube.com/watch?v=HbSjC1ELX78

https://www.youtube.com/watch?v=YTi2Gsl4DfE

Questi numerosi jingle che possiamo ascoltare sono frutto di una rigida selezione tra altrettanto numerosi compositori che, con ferocia, annualmente, tentano di vendere la propria composizione alle numerose ditte ferroviarie giapponesi.
Ognuno di questi jingle, difatti, deve soddisfare rigidissimi requisiti, poiché ciascun jingle sarà emesso dalla sola ditta che ha acquistato tali diritti di copyright.
Non solo, ogni azione del treno (partenza, arrivo...) di ciascuna ditta, potrà emettere unicamente il jingle designato per tale azione.
Un jingle per segnalare che il treno sta arrivando.
Un jingle per segnalare che le porte si stanno per aprire e uno che si stanno per chiudere.
Un jingle per segnalare che il treno sta lasciando la stazione.
Quindi ogni specifico repertorio di questi jingle può essere emesso unicamente dalla ditta che l'ha acquistato e soprattutto ad ogni azione del treno verrà univocamente emesso il jingle designato per quella sola azione.
Perché sto insistendo in questa precisazione?
Il motivo è semplice, in una qualsiasi stazione di ciascuna città, sono frequentemente presenti più di una ditta di treni e/o metropolitana. Oltre alla classica linea nazionale JR esistono le ditte ferroviarie: Tobu, Seibu, Keisei, Keio, Tokyu, Keikyu, Tokyo metro, Sagami, Meitetsu, Kintetsu, Nankai, Keihan, Hankyu, Hanshin, NishiNippo, etc..
Ognuna deve essere ben riconoscibile dai passeggeri in procinto di salire e/o che sono a bordo.
Quindi, attenzione allo scioglilingua che state per leggere.

Un passeggero che da lontano sta correndo in direzione del proprio treno, da prendere al volo, udendo e riconoscendo da lontano il jingle pertinente all'azione che la ditta proprietaria di quel treno sta per intraprendere, potrà decidere di:
1) adattare la propria corsa per riuscire a prendere in tempo il treno che emette il jingle di imminente "arrivo in stazione", oppure,
2) rinuncerà a correre, udendo il jingle che quel treno "sta per lasciare la stazione", così il ritardatario giungerà in stazione passeggiando tranquillamente per poi salire sul treno successivo.

Un servizio veramente notevole!

Una volta saliti a bordo, dagli altoparlanti di ciascuna carrozza, saranno emessi sia gli svariati jingle pertinenti alle azioni che sta per intraprendere il treno, sia le informazioni in giapponese, inglese, coreano, cinese, delle varie stazioni già percorse, le stazioni che percorrerà, la stazione della prossima fermata, insieme a numerosi avvisi di non lasciare bagagli incustoditi, informazioni turistiche, informazioni meteo, informazioni sulle coincidenze di altri treni in attesa alla prossima stazione e infinite altre informazioni.

Un servizio altrettanto notevole!

Il video di quel bimbo che chiacchiera in maniera così fluente altro non è che una precisa sequenza di informazioni emesse nelle carrozze durante un tratto percorso dal treno, con i rispettivi jingle delle azioni che sta per intraprendere il treno stesso. Quelle informazioni sono già entrate nella testa di quel bimbo come le preghiere per noi italiani: dopo averle ascoltate così tante

volte, vengono memorizzate per tutta la vita.
Questi jingle devono possedere ferrei requisiti: devono essere orecchiabili e ben memorizzabili, non devono possedere suoni striduli o acuti poiché non devono infastidire il silenziosissimo equipaggio delle carrozze. Se, come frequentemente avviene, un passeggero sta riposando nel treno, allora il jingle che comunica l'arrivo alla stazione deve possedere requisiti di "sveglia", ma senza allarmare gli altri passeggeri, etc..

Un jingle comunicherà che il treno sta per fermarsi, uno che si aprono le porte, uno che si chiudono, uno che il treno riparte etc. Sono tutte musichette stile "carillon", con una spiccata tonalità orientale, adorabili ed indimenticabili, le ascolteremo tutte, tantissime volte, durante tutto il nostro viaggio, emesse in precisa sequenza, in base al numero di stazioni che dovremo percorrere.

Oltretutto nelle stazioni più sperdute delle campagne, anche quando da una eternità non si vede l'ombra di alcun treno, vengono emesse da numerosi altoparlanti della stazione stessa, numerose varianti di fischi di volatili.
Anche questa specie di "jingle" ha uno scopo: probabilmente sono poco graditi agli onnipresenti corvi ed hanno la funzione di dissuasori acustici contro questi volatili.

Un servizio ancor più notevole!

In sintesi, dietro queste musichette, apparentemente casuali, del costoso giocattolo (circa 50€) prodotto dalla ditta "Targa", dimora uno scopo sociale profondo e preciso: educare i bimbi a memorizzare ciò che a breve lubrificherà l'apparato meccanico del loro "orologio sociale", che sarà inevitabilmente gettato in pasto alla frizzante macchina nipponica, rodata, ultra precisa, con treni super puntuali che arrivano e partono a ritmi serrati.

Ora che ci penso, sto proprio vivendo un attimo di grande lusso nel recensire questo gioco del passato, comodamente, davanti al Pc. Ogni tanto inserisco un soldino per ascoltare con tanta serenità tutti questi jingle, pensando che dall'altra parte del mondo qualche businessman sta ascoltando il medesimo jingle e, magari, sta correndo col fiatone sotto una fredda pioggia, dentro una stazione così troppo affollata, con la propria pesante ventiquattrore, spiegazzando la povera cravatta, per non perdere quel treno, ancora così tanto lontano!

Cari lettori, ci aggiorniamo alla prossima recensione! Un abbraccio.

Elenco dei negozi fisici che trattano RetroGaming in Italia

a cura di Starfox Mulder

Salve a tutti lettori, da questo mese (pronti ad aggiornare la lista nei prossimi) elencherò tutti i negozi "reali" che vendono materiale retroludico.

Per reali intendo che NO, niente venditori online. Lo scopo è quello di poter andare in loco a visitare il negozio e spulciare tra le rarità messe in vendita, per gli acquisti online ci sono i siti internet.

Quella che vedrete è una prima infornata, in attesa che negozianti da tutta Italia (o lettori informati) mi comunichino di altre location.

- NORD ITALIA -

Emilia Romagna

Nome: Bip Games
Indirizzo: Via Giuseppe di Vittorio, 16
Città: San Lazzaro di Savena (BO)
Numero di telefono: 347/3968714

Nome: Brain Fusion
Indirizzo: Piazzale Melozzo degli Ambrogi, 1
Città: Forlì (FC)
Numero di telefono: 0543/33221

Nome: Fumetteria Micro Talpa Studio
Indirizzo: Via G. Garibaldi, 36B
Città: Minerbio (BO)
Numero di telefono: 393/3716121

Nome: Jolly Roger Bay Videogames
Indirizzo: Via Monchio, 6
Città: Carpi (MO)
Numero di telefono: 059/650846

Nome: Time Out Videogames
Indirizzo: Via di Corticella, 119/a
Città: Bologna (BO)
Numero di telefono: 051/6313325

Nome: Virtual Game
Indirizzo: Corso Giuseppe Garibaldi, 39/1
Città: Lugo (RA)
Numero di telefono: 0545/25022

Nome: World Games Videogiochi
Indirizzo: Via Emilia Est, 71
Città: Modena (MO)
Numero di telefono: 059/238968

Friuli Venezia Giulia

Nome: Arcana Comics And Games
Indirizzo: Via Francesco Mantica, 23
Città: Udine (UD)
Numero di telefono: 0432/503725

Nome: Brain Records
Indirizzo: Via Giulia, 64
Città: Trieste (TS)
Numero di telefono: 040/351280

Liguria

Nome: Arcadia Games
Indirizzo: Via Felice Cavallotti, 85
Città: La Spezia (SP)
Numero di telefono: 0187/1864355

Lombardia

Nome: DB Games
Indirizzo: Via Nino dall'Oro, 19
Città: Lodi (LO)
Numero di telefono: 331/803/ 9891

Nome: Gamepeople Gallarate
Indirizzo: Via Torino, 23
Città: Gallarate (VA)
Numero di telefono: 0331/781478

Nome: Old Game Videogiochi e Retrogame
Indirizzo: Via Concordia, 9
Città: Assago (MI)
Numero di telefono: 02/4881859

Nome: Play Store Italia s.r.l.
Indirizzo: Via Enrico Fermi, 56
Città: Curno (BG)
Numero di telefono: 035/612806

Nome: Second Life
Indirizzo: Corso Giuseppe Garibaldi, 89
Città: Seveso (MB)
Numero di telefono: 0362/1544435

Piemonte

Nome: Caos A.D.
Indirizzo: Corso Nizza, 64
Città: Cuneo (CN)
Numero di telefono: 0171/605194

Nome: Gameland
Indirizzo: Via Tornio, 65
Città: Trofarello (TO)
Numero di telefono: 011/19479669

Nome: Il Mercatuccio
Indirizzo: Str. di S. Mauro, 197/A
Città: Torino (TO)
Numero di telefono: 346/3892231

Nome: Videogames Generation
Indirizzo: Via Fratelli Vercelli, 76
Città: Carmagnola (TO)
Numero di telefono: 011 086 0334

Veneto

Nome: Arena Games
Indirizzo: Via Padova, 26
Città: Selvazzano Dentro (PD)
Numero di telefono: 049/8176825
Nome: Mioshi di Alessandro Ramorino
Indirizzo: Piazzale Vittime delle Foibe, 17
Città: Belluno (BL)
Numero di telefono: 0437/942510

Nome: Mondo Virtuale
Indirizzo: Via Garibaldi, 42
Città: Valdagno (VI)
Numero di telefono: 0445/401622

Nome: Power Games
Indirizzo: Via Gracco Spaziani, 33
Città: Verona (VR)
Numero di telefono: 045/569848

Nome: Retrogaming Point
Indirizzo: Via G. dai Libri, 1B
Città: Verona (VR)
Numero di telefono: 371/359 4593

- CENTRO ITALIA -

Lazio

Nome: DVD Planet
Indirizzo: Via Messala Corvino, 47
Città: Roma (RM)
Numero di telefono: 06.45422855

Nome: Il Nascondiglio dell'Androide
Indirizzo: Via Giuseppe Bagnera, 46
Città: Roma (RM)
Numero di telefono: 331/7825543

Nome: L'Arcobaleno
Indirizzo: Via del Portico d'Ottavia, 2,3
Città: Roma (RM)
Numero di telefono: 06/68192825

Nome: Player One Videogame Roma
Indirizzo: Via Lago Tana, 10/B
Città: Roma (RM)
Numero di telefono: 06/86212382

Nome: Retro Games Club
Indirizzo: Via Messala Corvino, 47
Città: Roma (RM)
Numero di telefono: 392/0466987

Nome: Video Boy Club
Indirizzo: Via Casilina, 1336
Città: Roma (RM)
Numero di telefono: 06/78343837

Marche

Nome: Game Evolution SAS
Indirizzo: Viale Papa Giovanni XXIII, 7
Città: Jesi (AN)
Numero di telefono: 0731/214670

Nome: Games Time Fano
Indirizzo: Via Roma, 124C
Città: Fano (PU)
Numero di telefono: 0721/1540212

Nome: VideoJays
Indirizzo: Via Passeri, 54
Città: Pesaro (PU)
Numero di telefono: 328/2217697

Toscana

Nome: I Giochi di Alice
Indirizzo: Via Giovanni da Montorsoli, 61
Città: Firenze (FI)
Numero di telefono: 055/7130202

Nome: Kuma Retroshop di Laura Bassetti
Indirizzo: Via A. Gramsci, 287
Città: Sesto Fiorentino (FI)
Numero di telefono: 055/3851476

Nome: Retrogame snc di Borgognoni Francesco
Indirizzo: Via Claudio Monteverdi, 8
Città: Firenze (FI)
Numero di telefono: 055/351069

Umbria

Nome: Retrogaming & Toys
Indirizzo: Via della Rinascita 17/19
Città: Terni (TR)
Numero di telefono: 0744/460695

- SUD ITALIA -

Campania

Nome: Future Games
Indirizzo: Via S. Carlo, 8
Città: Caserta (CE)
Numero di telefono: 0823 216516

Nome: Gamebusters
Indirizzo: Via Vittorio Emanuele III, 95
Città: Aversa (CE)
Numero di telefono: 081/19649062

Nome: Max & Jo' Assistenza & Accessori: Sony Microsoft Nintendo Apple
Indirizzo: Via Santa Maria in Portico, 43
Città: Napoli (NA)

Numero di telefono: 081/0323608

Nome: Top Games
Indirizzo: Via Sant'Anna dei Lombardi, 15
Città: Napoli (NA)
Numero di telefono: 081/4202131
Secondo punto vendita, Indirizzo:
Via Salvador Dali', 116, Napoli,
Tel. 081/7267877

Puglia

Nome: Nerd's Landing
Indirizzo: Via Roma, 50
Città: Acquaviva delle Fonti (BA)
Numero di telefono: 391 758 6232

Sardegna

Nome: Altrove
Indirizzo: Via Carlo Alberto, 111
Città: Alghero (SS)
Numero di telefono: 079/983584

Sicilia

Nome: 16 Games
Indirizzo: Via Antonino Pecoraro, 7
Città: Palermo (PA)
Numero di telefono: 091/512125

Nome: Al Teknodromo snc
Indirizzo: Via Generale Arimondi Giuseppe, 29
Città: Palermo (PA)
Numero di telefono: 091/9778097

Nome: Bit-World
Indirizzo: Via Cataldo Parisio, 41
Città: Palermo (PA)
Numero di telefono: 091 560 1491

Nome: Console & Mania
Indirizzo: Via A. De Gasperi, 83
Città: Palermo (PA)
Numero di telefono: 091 512340

Nome: Il Chioschetto del Videogioco
Indirizzo: Via Duca della Verdura, 13
Città: Palermo (PA)
Numero di telefono: 389/1272091

Nome: Insert Coin Videogames
Indirizzo: Via Generale Arimondi Giuseppe, 62
Città: Palermo (PA)
Numero di telefono: 091/7794282

Nome: PlayEvolution
Indirizzo: Via Sammartino, 38
Città: Palermo (PA)
Numero di telefono: 091/332788

Nome: Play House Console & Videogames
Indirizzo: Via Alfredo Casella, 8
Città: Palermo (PA)
Numero di telefono: 091/6826259

Nome: Playnow
Indirizzo: Via Sardegna, 24-26
Città: Palermo (PA)
Numero di telefono: 091/8544014

Nome: Showgame
Indirizzo: Via I. Pizzetti, 46
Città: Palermo (PA)
Numero di telefono: 091/6819328

Retroeventi d'inverno

L'inverno e il freddo sono davvero arrivati, ma per fortuna noi amanti del retrocomputing e del retrogaming sappiamo riscaldarci con poco.
Ci bastano infatti qualche evento e mercatino per accendere i nostri cuori di retroappassionati.
Ecco perciò due importanti appuntamenti che vi vogliamo segnalare in questo freddo mese di febbraio.

Se siete dei veri amanti del videogioco allora conoscerete sicuramente il Vigamus, il museo del videogioco di Roma che ha come scopo quello di preservare, ricercare e divulgare il videogioco come arte e come massima espressione di comunicazione digitale.
Tra le tante attività che il Vigamus organizza, quest'anno c'è anche il **Retro Game Festival** che si terrà nelle sale del museo l'8 e il 9 febbraio.
Questo retroevento, che avrà ingresso gratuito, è stato pensato dagli organizzatori come un fine settimana all'insegna del divertimento e della cultura.
Sono infatti previste conferenze con esperti del settore, tornei con titoli esclusivamente retro, visite guidate all'interno del museo e tante attività per i bambini, per avvicinarli al mondo del retrogaming e del retrocomputing.
Insomma l'evento sembra garantire un'occasione unica per tutta la famiglia per passare del tempo piacevole ed istruttivo all'insegna di un amico che davvero non tradisce mai...il videogioco.

Il secondo importante evento è la quarta edizione del **Fermo Forum Comic and Games** che si terrà nella città di Fermo il 15 e 16 febbraio.
L'evento si presenta come un vero e proprio contenitore di tante attività interessanti. Innanzitutto ci sarà un'area mercato con tanti stand in cui gli espositori potranno mostrare e vendere i loro prodotti.
Presente poi anche una bella area dedicata al modellismo di terra e navale, nonchè una dedicata ai mitici mattoncini Lego in cui, oltre ad ammirare le opere esposte, ci si potrà sfidare tramite corse Lego Racer.
Non mancherà nemmeno la consueta gara cosplay e la presenza di tanti ospiti del mondo del disegno e del fumetto.
Ma ancora una volta il cuore dell'evento sarà l'area dei videogames moderni e retro.
L'area moderna sarà curata da AK Informatica by Pc e Computer che allestiranno diverse postazioni con i migliori titoli del momento per un'esperienza di gioco a 360 gradi.
l'area retro sarà gestita invece da Bitstore che allestirà consolle Nintendo come il NES, Nintendo 64, Gamecube e Wii, mentre per i giochi si passerà dai classici Supermario Bros e Donkey Kong, fino ad arrivare ai nuovi titoli come Splatoon, Supermario Kart e Luigi Mansion.
Prevista infine anche un'area Xbox e Playstation con tornei a Tekken, Fifa e Pro Evolution Soccer.

di Querino Ialongo

EVENTI FEBBRAIO 2020

RETRO GAME FESTIVAL
8-9 FEBBRAIO ROMA

FERMO FORUM
COMIC AND GAMES
15-16 FEBBRAIO FERMO

Punto! Due punti! ...ma sì, fai vedere che abbondiamo... Abbondandis in abbondandum.

(Totò nel film Totò, Peppino e la... malafemmina)

Siamo veramente fortunati! Abbiamo un'abbondanza di materiale a disposizione che nemmeno immaginiamo.

Di quale materiale sto parlando? Dei manuali, delle guide, dei libri e delle riviste che tanto tempo fa agognavamo e che attualmente invece siamo abituati a trovare con una semplice ricerca con Google.

Ripenso alla mia infanzia, quando per scovare un libro del Commodore 64 o dell'Amiga (ho posseduto queste due macchine) dovevo andare a Firenze a spulciare la libreria Marzocco. Ricordo che il reparto dedicato ai computer si trovava in una stanzetta al primo piano, nascosto alla vista dei comuni lettori, dopo il reparto esoterismo, quasi che anche la ricerca di un libro di informatica fosse un'attività esoterica...

Se ripenso a quel periodo, quando non potevo permettermi tutti i libri che desideravo e guardo invece all'abbondanza che adesso ho a portata di mano, mi sento immensamente fortunato. Fortunato perchè la mia passione può essere alimentata e la mia curiosità soddisfatta. Centinaia di libri contenenti concetti ed informazioni che non aspettano altro di essere letti per dissetare la mia sete di conoscenza.

Cosa desiderare ancora?
Beh, forse qualcosa da desiderare ancora ci sarebbe... Il tempo... Il materiale a disposizione è decisamente tanto, forse troppo, mentre il tempo da dedicargli è purtroppo limitato.

Come dice un vecchio adagio: chi_ha pane non_ha denti... Quando avevamo tempo in abbondanza, pagavamo lo scotto nel reperire le informazioni, adesso che potremmo facilmente accedere a tutto quello che prima potevamo soltanto sognare, la vita ci ricorda che dobbiamo fare altro.

Ma forse un angolo di tempo possiamo sempre ritagliarcelo e vivere questo momento di abbondanza come una seconda possibilità. Personalmente lo sto facendo ed è anche la ragione per la quale è nata e sopravvive la nostra rivista.

E voi? State approfittando di questo momento?

Francesco Fiorentini

RetroMagazine
Anno 4 - Numero 20

Direttore Responsabile
Francesco Fiorentini

Vice Direttore
Marco Pistorio

Gennaio 2020