PROGRAMMING - HISTORY - INTERVIEWS - REVIEWS - INSIGHTS

Future days are back

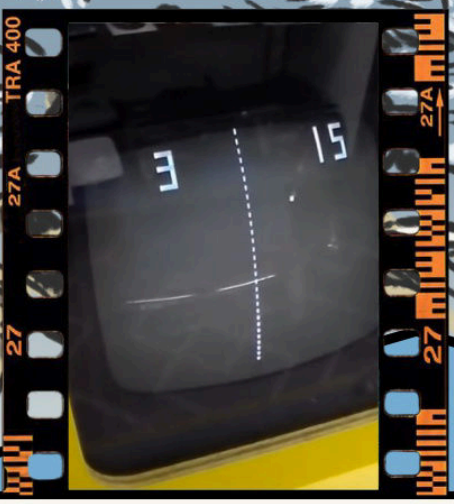
RETROMAGAZINE WORLD

ISSN 3103-697X

ISSUE 27 - WINTER EDITION - WWW.RETROMAGAZINE.NET

DARK KNIGHT

BATMAN RETURNS TO ZX



ATARI PONG

120 PAGES!

Digital scans = Permanent memory

It's just after 11:30 p.m. Tomorrow I have to wake up early for a work commitment that has been scheduled for some time. I'm already in bed, but after a stressful day spent in front of the computer, sleep is slow in coming. In fact, it just doesn't want to come at all. In the dim light of a lamp, the room seems to take on a dark amber color. With an almost automatic gesture, I reach for my tablet on the bedside table and open the app that accesses my home NAS, part of which is dedicated to the nostalgic world of retrocomputing, an archive full of technical articles, manuals, books, and magazines published mainly in the 80s and 90s. I select the 'Sinclair' folder, then 'Mags', and open issue 24 of Sinclair User, the English edition from March 1984. The summary lists various articles for the ZX Spectrum and a presentation/ review of the then-new jewel of the company founded by Sir Clive, the Sinclair QL. After skimming through a few programming articles and the special feature on the QL, Morpheus' welcoming arms finally envelop me.

The next morning, on my way to a business meeting, I think back to the "door of time" that I was able to open the night before, to a world that no longer exists, but which I can access whenever I want. And I ask myself: who should I thank for this fantastic opportunity? Obviously, the publishing house that published the magazine in those years, but also the work of a single enthusiast, an organized group, or the entire retrocomputing community that carried out the digitization. Certainly, I am not the first to discover the importance of digital scanning. But here I would like to emphasize a few key points that highlight the great usefulness of these efforts made by groups of passionate enthusiasts.

Preservation of historical knowledge: digital scanning allows us to preserve knowledge related to the technology of the past, documenting how hardware and software have developed over the decades. After so many years, this cultural and technical heritage is in fact at risk of disappearing.

Universal accessibility: with digitization, anyone, anywhere in the world, can access rare materials without having to physically own a hard copy.

Preservation of technical documentation: manuals and magazines often contain crucial technical information, such as wiring diagrams, hardware specifications, and programming procedures, which enable the restoration and maintenance of old computers and consoles.

[continues and ends on the back cover...]

Publication Note: this editorial and all the contents were first released in Italian in November 2024 - ref. issue 49-IT

TABLE OF CONTENTS

	PAG.
◊ Atari PONG	Pag. 3
◊ Atari 2600 – Tecnologie AI...	Pag. 9
◊ The SPECTRUM	Pag. 13
◊ Il Nintendo 64	Pag. 16
◊ Donkey Kong su PC128 Olivetti Prodest	Pag. 18
◊ Il C64 incontra l'Intelligenza Artificiale	Pag. 23
◊ Zena	Pag. 29
◊ Commdore 64 - BASIC V2	Pag. 32
◊ Inform 6 - Tutorial 4	Pag. 34
◊ To boldly go where no SID has gone before...	Pag. 43
◊ Il peso del Tetris	Pag. 52
◊ Giappone 25^puntata: Gachagacha...pon!	Pag. 58
◊ Fireteam Rogue per Super Nintendo...	Pag. 60
◊ Firenze Vintage Bit 2024	Pag. 64
◊ Quinty (SNes)	Pag. 67
◊ Escape 2042 (Jaguar)	Pag. 68
◊ Kudzu (GB)	Pag. 70
◊ Zephyr's Pass (GBC)	Pag. 72
◊ King's Quest VI (Atari ST)	Pag. 74
◊ Spelunky (C64)	Pag. 76
◊ Dark Knight (ZX)	Pag. 78
◊ Hayato's Journey (MD)	Pag. 80
◊ Super Dassalo Land (GBC)	Pag. 82
◊ Black & White (NEXT)	Pag. 84
◊ Data Man (NES)	Pag. 86
◊ Ruff in Trouble (Atari 8bit)	Pag. 88
◊ Super Mario Bros Mini (GBC)	Pag. 90
◊ Super Mario Eclipse (GameCube)	Pag. 92
◊ Tony Montezuma's Gold (C64/...)	Pag. 94
◊ Jester (NES/SNes)	Pag. 97
◊ Dr. Dangerous (Amiga)	Pag. 98
◊ Parasol Stars (Switch)	Pag. 100
◊ Portal Buster (C64)	Pag. 102
◊ Binary Battle (C64/...)	Pag. 104
◊ Shift (C64/...)	Pag. 106
◊ Metal Slider Glory... (Super Famicom)	Pag. 108
◊ Qyxar (Coleco)	Pag. 110
◊ Pokettohiro! (GBC)	Pag. 111
◊ Spanky's Quest (GB)	Pag. 112
◊ The Curse of Lies (MSX2/...)	Pag. 113
◊ Lunar Lancer (GB)	Pag. 114
◊ Tiburoncy (C64)	Pag. 115
◊ Aliens Neoplasma 2 (ZX/...)	Pag. 116
◊ Twist of Souls (Atari 8bit)	Pag. 117
◊ Chimney Man (MSX)	Pag. 118
◊ Back to the Future II (C64)	Pag. 119

The following people contributed to this issue of RetroMagazine World (in no particular order):

- Carlo N. Del Mar Pirazzini
- Daniele Brahimi
- Francesco Fiorentini
- Paolo Cattaneo
- Roberto Del Mar Pirazzini
- Giampaolo Moraschi
- Leonardo Miliani
- Takahiro Yoshioka
- Barbara "Morgana" Murgida
- Eugenio Rapella
- Javier Gonzalez
- Marta Rossmann
- Gianluca Girelli
- Marco Pistorio
- Ksenia Kamash
- Michele Ugolini
- Evan Granovitz
- Alberto Apostolo
- Ingrid Poggiali
- David La Monaca
- Bimbomillo
- Fredrik Ramsberg

- Cover image: **Giuseppe Mangini**
- Cover layout: **Carlo N. Del Mar Pirazzini**

Atari PONG

by Leonardo Miliani

In issue 19-EN (ref. 41-IT) of this magazine, published about two years ago, we talked about the history of the VCS/2600, an Atari console that marked an era by transforming video games into a real industry. But this console was not the first successful one, nor was it the one that launched Atari into the Olympus of video game manufacturers. Before that, the company founded by Nolan Bushnell had released another successful product that brought the company fame and money. A product so iconic that, in the second half of the 1970s, it was cloned by virtually every electronics manufacturer: we are talking about PONG (fig. 1).

History repeats itself

During his summer vacations while in college, Nolan Bushnell worked at an amusement park, where he began to develop an entrepreneurial mindset by observing how people spent money on the park's electromechanical games. He later attended engineering courses at the University of Utah in Salt Lake City, where he came into contact with one of the first, rudimentary video games: it was at this university that Spacewar, a space-themed video game in which two spaceships battle each other, had been created a few years earlier. The game ran on a PDP-1 and was controlled by rudimentary joysticks. After university, he married and moved to Silicon Valley, where he went to work for Ampex, which produced audio/video recording systems on magnetic tape. There he met electronic engineer Ted Dabney, with whom he became friends. The two later learned that a smaller version of Spacewar had been developed based on a PDP-11, called Galaxy Game. Compared to the original, the computer and all the electronics were contained in a single cabinet and, more importantly in Bushnell's eyes, the game was coin-operated. Bushnell remembered the money he had spent on the coin-operated games in the arcade where he worked as a boy and came up with the idea of creating his own video game. Together with Dabney, they founded Syzygy in 1971 and created their first arcade game, Computer Space. Unlike Spaceware and Galaxy Game, Computer Space was much simpler, being based on TTL logic rather than a computer, which also made it cheaper.



Fig. 1 - The original Atari PONG cabinet (image: Patrick Nafarrete – source: Wikimedia Commons)



Fig. 2 - Alan Alcorn (image: Alex Handy - source: en.wikipedia)

They pitched the game to Nutting Associates, who distributed it without success: the gameplay was difficult for people who were not yet accustomed to video games and more inclined towards the classic entertainment machines of the time. When they proposed creating a simplified version to boost sales, Nutting Associates backed out, refusing to distribute it.

(If you want to learn more about the early years of Bushnell and Atari itself, I recommend rereading the aforementioned article on the Atari VCS/2600 from issue 41 of the magazine).

The development of PONG

Bushnell and Dabney realized that the current distributors of electromechanical games were not yet ready for the electronic revolution and decided to distribute the games they would produce from that moment on themselves. However, they needed further help and decided to hire a former colleague, another engineer they had worked with at Ampex, Alan "Al" Alcorn (fig. 2).. Bushnell and Dabney had in mind to create a driving game, but they thought the task was too challenging for Alcorn, not only because he was completely new to video games, but also because Alcorn was an analog engineer, while the video games Bushnell had in mind required digital knowledge. This is where the Magnavox Odyssey console came into play. In May 1972, before the official presentation, Magnavox toured several US states to gather pre-launch impressions of the console and its games. Bushnell also took part in one of the stops, and his attention was caught by one of the games which, despite its flaws, he thought had potential: Tennis. There were a couple of 'rackets' on either side of the screen that bounced a 'ball' around the screen.

Bushnell decided to use this game as a test bed to familiarize Alcorn with video game development. To motivate the engineer, he told him a little white lie: he said he had a contract with General Electric for a device that had to be delivered within a certain time frame. This device had three fixed elements: there had to be a moving object (a "ball") that bounced around the screen, it had

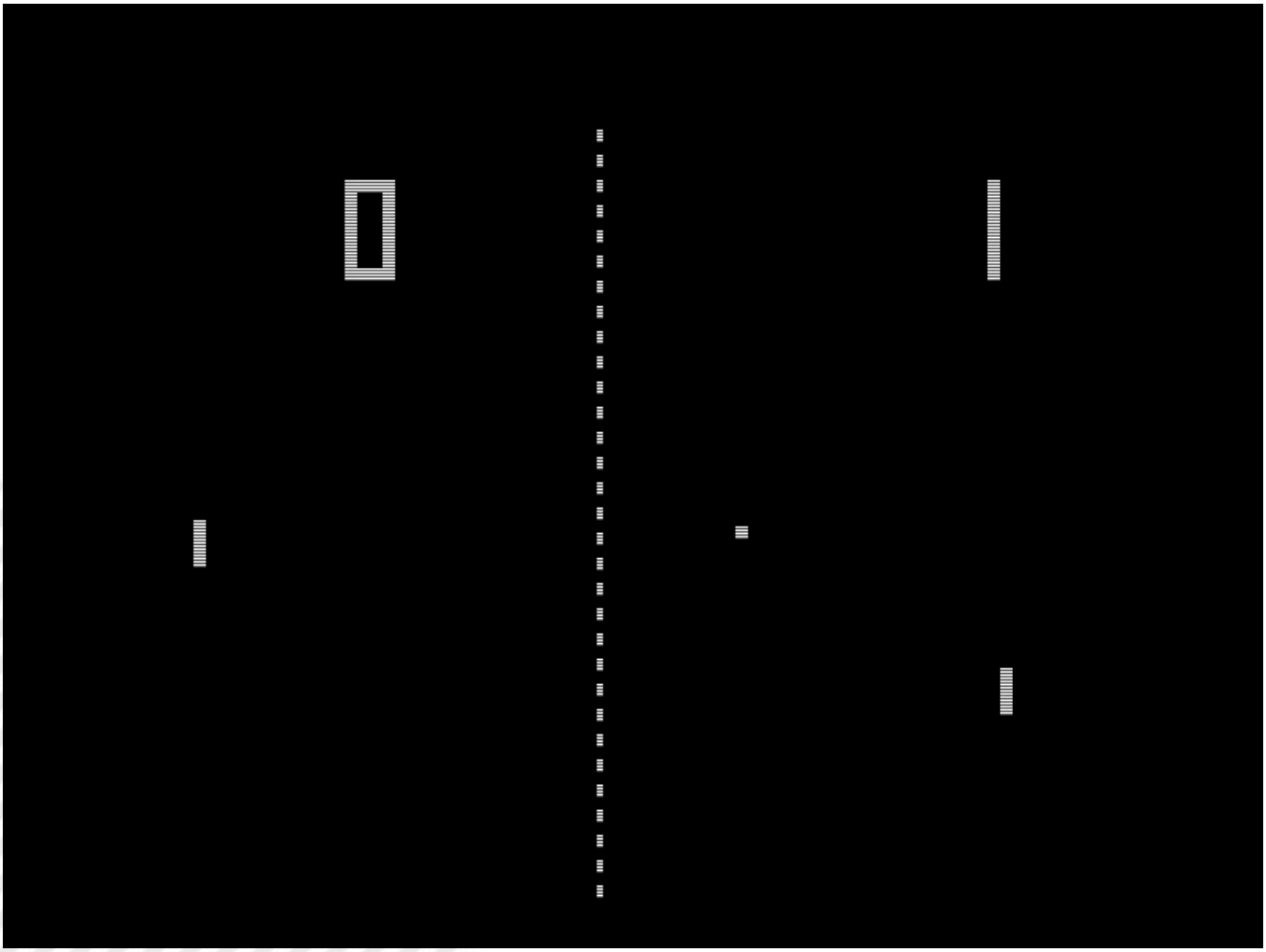


Fig. 3 - The PONG game screen: 2 paddles bounce a ball, a scoreboard at the top keeps track of the 2 players' points (image: Bumm13 - sorce: En.wikipedia)



Fig. 4 - Andy Capp's Tavern, as it appears today (a new bar took its place in 1984), where the PONG prototype was installed in 1973 (image: CoolCaesar - source: Wikimedia Commons)

to have two paddles to control two "rackets," and it had to display a score on the screen. Alcorn began working on the project and, within a few months, created a prototype that met the requirements, adding some features that made his video game much more intriguing than Odyssey's Tennis. For example, the ball bounces at different angles: to achieve this, Alcorn divided the racket segment into eight smaller segments and, depending on which one was hit, the ball changed angle. The speed of the ball is also not always the same: with each bounce, it increases to raise the level of difficulty, returning to its starting speed only when it goes out on either side (i.e., when a player misses the ball on a return). The score is shown with numbers at the top similar to those shown on 7-segment LED displays: the first player to reach 11 points wins (fig. 3).

Curiously, a problem with the paddle-based racket control system is not corrected by Alcorn, but left as an additional difficulty for the player: in the system developed by the engineer, there is a defect whereby the rackets cannot reach the top of the screen, thus preventing the player from hitting the ball when it travels very high. Alcorn later stated that this "flaw/feature," the speed of the ball increasing with each successful hit, and the different angle of the bounce are elements that exist to prevent two skilled players from continuing a game indefinitely.

At the end of development, Bushnell realized that the game had potential. He therefore asked Alcorn to add sound. The problem Alcorn faced was serious because the system was made entirely with digital components, and it was not easy to obtain analog signals without increasing the complexity of the board that had been built up to that point. He eventually solved the problem by using the video signal generation circuit, obtaining sounds using the electronic components already present.

At the end of all this work, what was supposed to be a simple training tool for Alcorn turned out to be a very interesting video game. However, before putting it on the market, they decided to test the game: they mounted the electronic board inside a wooden cabinet, where they

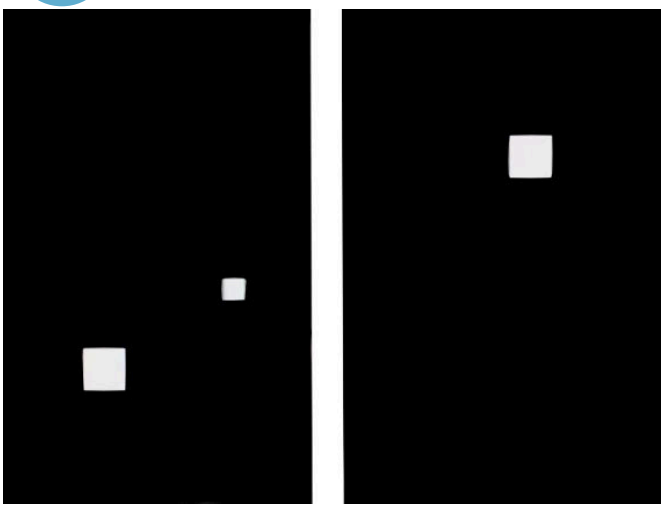


Fig. 5 -The Odyssey Tennis game, which inspired Bushnell to create PONG (image: Schnurrikowski - source: Wikimedia Commons)

also installed a common Toshiba TV bought in an appliance store, fixed two paddles on the front panel, and finally adapted the coin mechanism of a coin-operated washing machine to give the signal to start a game. Bushnell made an agreement with Bill Gaddis, one of Atari's customers who was supplied with pinball machines, who agreed to install the test machine in his bar, Andy Capp's Tavern in Sunnyvale (fig. 4). The game immediately showed its potential: many customers began to frequent the bar not only to eat and drink and then perhaps spend a few minutes playing one of the pinball machines, but also just to play this new video game, called "PONG" because it simulated a ping-pong match between two players, sometimes arriving even before opening time. After a few days, however, the game crashes: when Alcorn goes to see what the problem might be, he finds that the coin box is full of coins and rejects those inserted, no longer starting a new game. Once emptied, the game restarts, much to the delight of Bushnell, who realizes that this is a real bargain! On November 29, 1972, PONG was officially presented to the public and immediately became a success. After finding a financier and solving the assembly problems of the first models, the construction of the cabinets continues unabated, bringing a lot of money into Atari's coffers: just think that every single PONG arcade machine is sold on average for more than double its production cost, thus generating a nice profit. The game was a success and the market expanded: in early 1973, it was also exported abroad. By the end of 1973, 2,500 arcade cabinets had been sold, and the following year this figure reached 8,000: considering the period, the type of product, and the market for which it was intended, these numbers were quite respectable.

The lawsuit with Magnavox

Obviously, all this success did not go unnoticed, either by the general public, in whose eyes Atari began to build its reputation as a major video game manufacturer, which it would soon consolidate, or by someone who felt cheated out of their idea. Yes, we are talking about Magnavox, which saw too many similarities between the tennis game on its Odyssey console (fig. 5) and PONG. In early 1974, it therefore decided to enforce the patents registered by Ralph Baer, the original developer of the console (if you want to learn more about the history of Odyssey, we refer you to the article dedicated to it in issue 34 of our magazine), and sued not only Atari but several other companies that had released PONG clones in the meantime. Bushnell had actually attended the game's presentation, and the signatures of those present at the event incriminated him. Therefore, he did not want to risk a lawsuit that could drain Atari's coffers and decided to settle out of court, paying the (considerable) sum of $1.5 million in several installments to become a Magnavox licensee. The agreement, drawn up in June 1976, also stipulated that Atari would inform Magnavox of all products released on the market for a period of one year: this clause seems to have been one of the reasons why Atari waited until September 1977 to officially present its new VCS console



Fig. 6 - Tele-Games, the first version of Home PONG as marketed in the Sears chain of stores (image: Evan-Amos - source: Wikimedia Commons)



Fig. 7 - The Commodore TV Game 2000K, sold only in Europe (image: José Antonio Ortigueira Amor - source: Wikimedia Commons)

without being forced to reveal details of it to Magnavox.

The home version

Sales were decent, even though competition had affected them. To overcome this, a version of the game was developed on a single LSI (Large Scale Integration) chip, both to reduce production costs and to make it more difficult for competitors to reverse engineer PONG. The development of this chip takes time (and money) and, in the meantime, Bushnell asks his engineers to come up with something new, to squeeze out new ideas to make PONG profitable and bring new money into the company's coffers. Harold Lee, another Atari engineer, begins to develop the idea of using this new LSI chip for a home version of PONG. Lee worked with Alcorn on developing the chip while Bob Brown worked on an initial prototype that could be connected to a television. By the end of 1974, the chip was ready and so was the prototype: Atari

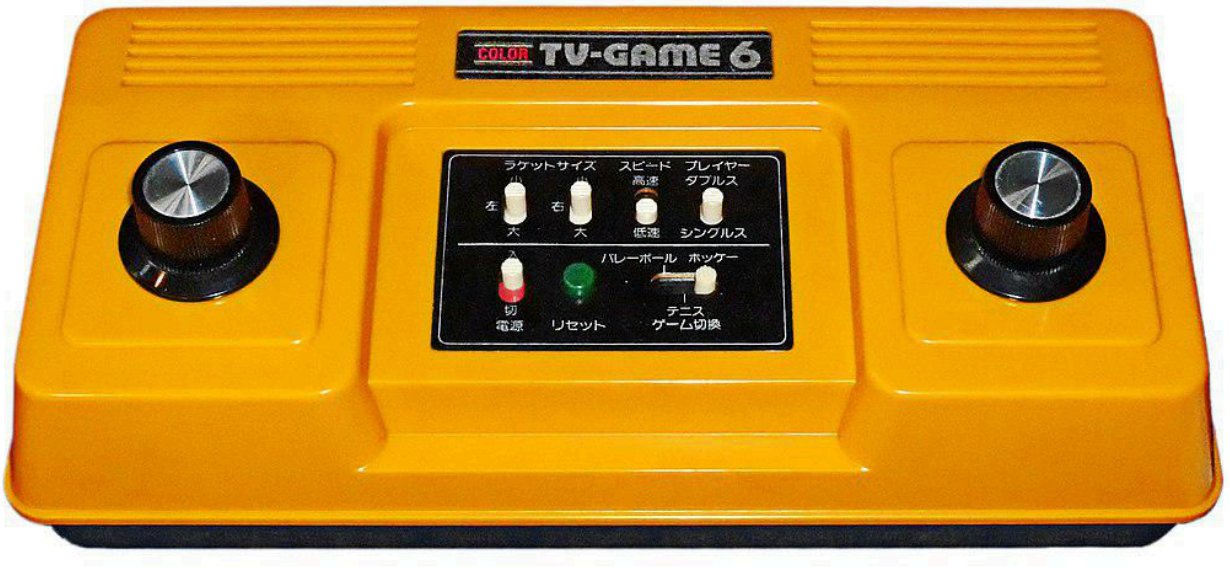


Fig. 8 - The Nintendo Color-TV Game 6 (image: Chapuisat - source: Wikimedia Commons)

then contacted Sears for the exclusive distribution of this home version of the game, which in the meantime had been named "Home PONG." Home PONG went on sale in Sears stores as "Tele-Games" (fig. 6) at a price of $99 between October and November 1975, just in time for the Christmas holidays: the approximately 150,000 units supplied to Sears were all sold. In 1976, Home PONG was also marketed directly by Atari, which sold about 50,000 units.

Clones and variants

The success of the arcade attracted competition: realizing that PONG was very popular and that this new video game market was profitable, clones of the game soon appeared on the market, eating into the market share of the original Atari. Replicating the arcade game was very simple: this was because the game did not contain a computer running a program but consisted of a simple electronic board made of discrete and integrated TTL components normally available on the market. All you had to do was copy the board and put in the same components, and the game was cloned.

As in the case of the arcade, the home version also faced the same problem, despite the fact that it was based on a single chip. The numbers recorded by Home PONG prompted several integrated circuit manufacturers to offer their own chips containing copies of the game. Often, manufacturers not only replicated the game but also offered several variants, some of which were improved. This was the case with MOS Technology, which offered a chip with four variants of the game, used by Commodore for its 2000K (fig. 7) and 3000H (sold only on the European market) consoles. Coleco, which would later release the famous ColecoVision computer, entered the home video game market with the Telstar console (fig. 9), sold in 14 models that were gradually updated with increasingly improved versions of the original PONG, for a total of about 1 million units. Nintendo also entered the world of home video games with a PONG clone called Color TV-Game (fig. 8): counting subsequent versions, the Color TV-Game sold around 3 million units, laying the foundations for the development of the more famous Nintendo Entertainment System.

Atari itself released its arcade game "Breakout" in 1976 as an updated version of PONG: unlike the latter, here a single player must break a wall at the top of the screen by destroying the individual bricks that make it up, hitting them with a ball that bounces off a paddle moving at the bottom of the playing field.



Fig. 9 - The Coleco Telstar
(image: Chris Yarzab - source: Wikimedia Commons)

The legacy

PONG is one of those iconic games: there is no one, in the period between the late 1970s and early 1980s, who did not come into contact with one of the hundreds of clones in circulation at that time, either because they had it at home or saw it at a friend's house. But even those who were not born or raised during that period recognize it, as it has been re-released over the years on many platforms, even in more recent times: for example, the game has been ported to various contemporary consoles and smartphones, proving that PONG, all these years after its debut, is still PONG. In fact, many publications consider PONG to be the product that transformed the video game industry into a lucrative business, and Home PONG to be the video game that led to the commercial success of the home console market. Over the years, numerous magazines have included it in their lists of the most famous titles in the history of video games, paying PONG the tribute it deserves for the coins and challenges

Atari 2600 - AI Technologies and Movements in Play

by Giampaolo Moraschi

An interesting project for our Atari 2600 that implements the use of hands instead of the classic joystick.
Created by Eran Feit, this concept sees every single movement on the screen made with the upper limbs. No joystick required. The project is able to detect body movements to simulate directions: up, down, left, right, and the "fire" actions of the button. You need a little familiarity with Python and a good dose of technical skill. The developer's web page (which can be found at this address:
https://medium.com/@feitgemel/a-revolutionary-way-to-play-atari-2600-using-ai-faa7d7ee7aa6
includes numerous examples and a diagram for connecting the project.

The code is developed in Python:

```
import mediapipe as mp
import cv2
import time

from numpy import hamming
#from activateKeyboard import IKEY, UKEY, OKEY, COMMAKEY, SPACE
#from activateKeyboard import PressKey, ReleaseKey

#
print("start Joystick")

import RPi.GPIO as GPIO
import time

UP_PIN = 6
LEFT_PIN = 23
RIGHT_PIN = 22
DOWN_PIN = 27
BUTTON_PIN = 12

GPIO.setmode(GPIO.BCM)
GPIO.setup(UP_PIN,GPIO.OUT)
GPIO.setup(LEFT_PIN,GPIO.OUT)
GPIO.setup(RIGHT_PIN,GPIO.OUT)
GPIO.setup(DOWN_PIN,GPIO.OUT)
GPIO.setup(BUTTON_PIN,GPIO.OUT)

# starting point - all LOW
# ========================
```

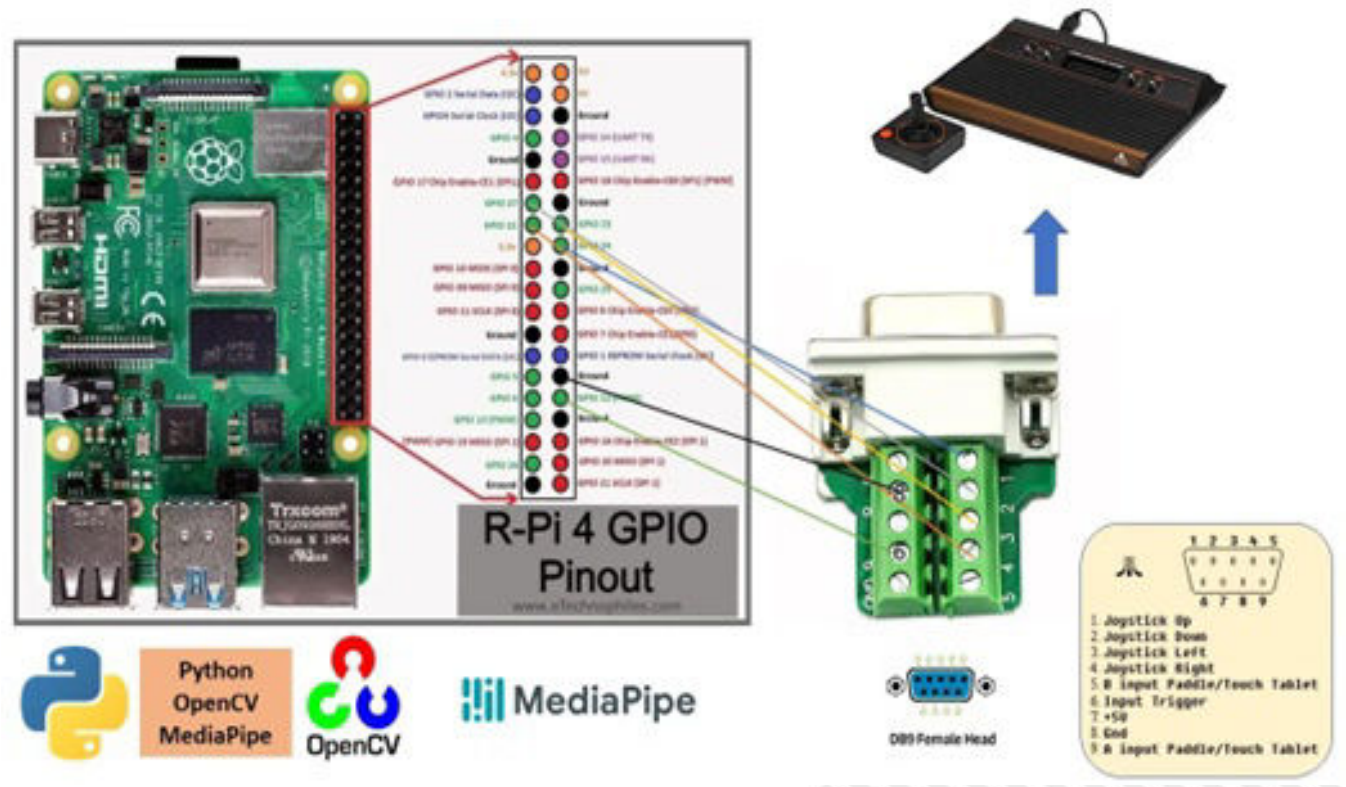


Fig. 1 - Connection diagram

```
GPIO.output(UP_PIN,GPIO.LOW)
GPIO.output(LEFT_PIN,GPIO.LOW)
GPIO.output(RIGHT_PIN,GPIO.LOW)
GPIO.output(DOWN_PIN,GPIO.LOW)
GPIO.output(BUTTON_PIN,GPIO.LOW)

mp_drwaing = mp.solutions.drawing_utils #drawing
uility - Help to render the landmarks
mp_hands = mp.solutions.hands

waitTimePressKey=0.3
difHeight = 100
maxHeight = 150
fireHeight = 80
LastMove=''
fireStatus=False

cap = cv2.VideoCapture(0)
cap.set(3,1280)
cap.set(4,720)

with mp_hands.Hands(min_detection_confidence=0.5,
min_tracking_confidence=0.5,max_num_hands=1)  as
hands:

    while cap.isOpened():

        re, frame = cap.read()

        scale = 75
        w = int(frame.shape[1] * scale / 100)
        h   = int(frame.shape[0] * scale / 100)

        dim = (w,h)

        resized = cv2.resize(frame , dim ,
interpolation = cv2.INTER_AREA)

        # Start the detection :
        # ======================

        # change it to RGB
        image =


cv2.cvtColor(resized,cv2.COLOR_BGR2RGB)

        # flip the image
        image = cv2.flip(image, 1)


        image.flags.writeable = False
        results = hands.process(image)
        image.flags.writeable = True

        centerX = int(w/2)
        centerY = int(h/2)

        gap = 100

        x1 = centerX - gap
        x2 = centerX + gap
        y1 = centerY - gap
        y2 = centerY + gap

          cv2.rectangle(image, (x1, y1), (x2,y2),
(255, 0, 0), 3)

        # center of frame
        #cv2.circle(image, (centerX,centerY) ,
2 , (255,0,0), -1 )

        if results.multi_hand_landmarks:

            for handLMS in
results.multi_hand_landmarks:

#mp_drwaing.draw_landmarks(image,handLMS) # draw
dots - handLMS is the hand index

mp_drwaing.draw_landmarks(image,handLMS,
mp_hands.HAND_CONNECTIONS) # draw connection -
handLMS is the hand index
                for id , lm in
enumerate(handLMS.landmark):
                    h , w , c = image.shape
                     cx , cy = int(lm.x * w) , int
(lm.y * h)
                    #print (id, cx , cy)


                    if id == 4 : # this is the
thumb landmark
                        cv2.circle(image, (cx,cy) ,
```

```
3 , (0,255,0), -1 )
                    thumbTipX = cx
                    thumbTipY = cy

                if id == 8 : # this is the
indexFingerTip landmark
                    cv2.circle(image, (cx,cy) ,
3 , (0,255,0), -1 )
                    indexFingerTipX = cx
                    indexFingerTipY = cy

                if id == 12 : # this is the
middleFingerTip landmark
                    cv2.circle(image, (cx,cy) ,
3 , (0,255,0), -1 )
                    middleFingerTipX = cx
                    middleFingerTipY = cy

                if id == 16 : # this is the
ringFingerTip landmark
                    cv2.circle(image, (cx,cy) ,
3 , (0,255,0), -1 )
                    ringFingerTipX = cx
                    ringFingerTipY = cy

                if id == 20 : # this is the
pinkeyTips landmark
                    cv2.circle(image, (cx,cy) ,
3 , (0,255,0), -1 )
                    pinkeyTipX = cx
                    pinkeyTipY = cy

                if id == 0 : # this is the wrist
landmark
                    cv2.circle(image, (cx,cy) ,
3 , (0,255,0), -1 )
                    wristX = cx
                    wristY = cy

        fireStatus=False
        GPIO.output(BUTTON_PIN,GPIO.HIGH)

        # check for fire

        #print("abs(wristY-
middleFingerTipY)",abs(wristY-middleFingerTipY))
        #print("abs(wristX-
middleFingerTipX)",abs(wristX-middleFingerTipX))
        #print("abs(pinkeyTipX-
thumbTipX)",abs(pinkeyTipX-thumbTipX))

        if abs(wristY-middleFingerTipY) <
fireHeight and abs(wristX-middleFingerTipX) <
fireHeight and abs(pinkeyTipX-thumbTipX) <
fireHeight:
            fireStatus=True
            GPIO.output(UP_PIN,GPIO.HIGH)
            GPIO.output(LEFT_PIN,GPIO.HIGH)
            GPIO.output(RIGHT_PIN,GPIO.HIGH)
            GPIO.output(DOWN_PIN,GPIO.HIGH)
            GPIO.output(BUTTON_PIN,GPIO.LOW)
            #print('FIRE')
            cv2.rectangle(image, (50, 50),
(250,150), (255, 255, 255), cv2.FILLED)
            cv2.putText(image,'Fire',(70,120),
cv2.FONT_HERSHEY_COMPLEX , 2 , (0,102,0), 5 )

        # Up direction
        if (indexFingerTipY < y1 and
fireStatus==False) :
            #print('UP')
            GPIO.output(UP_PIN,GPIO.LOW)
            GPIO.output(LEFT_PIN,GPIO.HIGH)
            GPIO.output(RIGHT_PIN,GPIO.HIGH)
            GPIO.output(DOWN_PIN,GPIO.HIGH)
            GPIO.output(BUTTON_PIN,GPIO.HIGH)
            LastMove="UP"
            cv2.rectangle(image, (50, 50),
(250,150), (255, 255, 255), cv2.FILLED)
            cv2.putText(image,'UP',(70,120),
cv2.FONT_HERSHEY_COMPLEX , 2 , (0,102,0), 5 )

            #print(str(wristY-middleFingerTipY))

        # Stop direction
        if (indexFingerTipX > x1 and
indexFingerTipX < x2 and indexFingerTipY > y1 and
indexFingerTipY < y2 and fireStatus==False)  :
            #print('STOP')
            GPIO.output(UP_PIN,GPIO.HIGH)
            GPIO.output(LEFT_PIN,GPIO.HIGH)
            GPIO.output(RIGHT_PIN,GPIO.HIGH)
            GPIO.output(DOWN_PIN,GPIO.HIGH)
            GPIO.output(BUTTON_PIN,GPIO.HIGH)
            LastMove="STOP"
            cv2.rectangle(image, (50, 50),
```

```
(250,150), (255, 255, 255), cv2.FILLED)
                cv2.putText(image,'STOP',(70,120),
cv2.FONT_HERSHEY_COMPLEX , 2 , (0,102,0), 5 )

                #print(str(wristY-middleFingerTipY))

            # down direction
            if (indexFingerTipY > y2  and
fireStatus==False)  :
                #print('DOWN')
                GPIO.output(UP_PIN,GPIO.HIGH)
                GPIO.output(LEFT_PIN,GPIO.HIGH)
                GPIO.output(RIGHT_PIN,GPIO.HIGH)
                GPIO.output(DOWN_PIN,GPIO.LOW)
                 GPIO.output(BUTTON_PIN,GPIO.HIGH)
                LastMove="DOWN"
                cv2.rectangle(image, (50, 50),
(270,150), (255, 255, 255), cv2.FILLED)
                 cv2.putText(image,'DOWN',(70,120),
cv2.FONT_HERSHEY_COMPLEX , 2 , (0,102,0), 5 )
              #print(str(middleFingerTipY-wristY ))

            # left direction
            if (indexFingerTipX < x1  and
fireStatus==False)  :
                #print('LEFT')
                GPIO.output(UP_PIN,GPIO.HIGH)
                GPIO.output(LEFT_PIN,GPIO.LOW)
                GPIO.output(RIGHT_PIN,GPIO.HIGH)
                GPIO.output(DOWN_PIN,GPIO.HIGH)
                 GPIO.output(BUTTON_PIN,GPIO.HIGH)
                LastMove="LEFT"
                cv2.rectangle(image, (50, 50),
(270,150), (255, 255, 255), cv2.FILLED)
                cv2.putText(image,'LEFT',(70,120),
cv2.FONT_HERSHEY_COMPLEX , 2 , (0,102,0), 5 )

            # right direction
            if (indexFingerTipX > x2 and
fireStatus==False)  :
                #print('RIGHT')
                GPIO.output(UP_PIN,GPIO.HIGH)
                GPIO.output(LEFT_PIN,GPIO.HIGH)
                GPIO.output(RIGHT_PIN,GPIO.LOW)
                GPIO.output(DOWN_PIN,GPIO.HIGH)
                 GPIO.output(BUTTON_PIN,GPIO.HIGH)
                LastMove="RIGHT"
                cv2.rectangle(image, (50, 50),
(270,150), (255, 255, 255), cv2.FILLED)
                cv2.putText(image,'RIGHT',(70,120),
cv2.FONT_HERSHEY_COMPLEX , 2 , (0,102,0), 5 )

        image = cv2.cvtColor(image,cv2.COLOR_RGB2BGR)
         cv2.imshow('image',image)
         cv2.moveWindow('image',40,0)

         if cv2.waitKey(10) & 0xFF == ord('q'):
             break


cap.release()
cv2.destroyAllWindows()
```

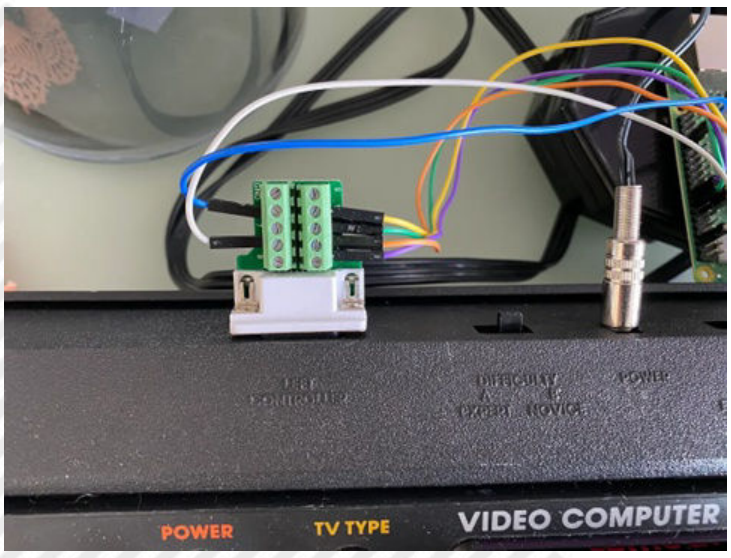


Fig. 2 - The finished project

In order to be developed, it also requires several components, and the author has included what is needed to try to implement it on the page.

We will try to follow its future development more closely in upcoming issues and on our Facebook page, and perhaps test it as soon as possible for a review.

THE SPECTRUM

How to win the heart of Sir Clive's product lovers

by Giampaolo Moraschi

Let me start by saying that I will be buying it. Many purists will be horrified by this choice, but in 2024, I would say it's time to stop competing over what is truly retro and what is not. I find it a useful and versatile product.
That said, here is my purely personal first impression of The Spectrum.
Plaion and Retro Games LTD officially announced it at the end of August 2024, causing a stir across the internet and beyond.
A modern recreation of the beautiful 1982 ZX Spectrum. The model was then 'hands-on' at Gamescon 2024, and we know that the European launch will take place on November 22 and that it is available for pre-order for â,¬99 and change.

For me, the ZX Spectrum has always had an incredible appeal in terms of aesthetics and some of the games it contains. It is considered one of the symbols of the tech revolution of the 1980s, and I am particularly happy about its 'rebirth'.
I was very impressed by the launch of a full version. No mini version or future maxi version. Here we have a product with 48 pre-installed titles, four USB ports, 720p HDMI outputs, and a perfect replica of the famous "rubber" keyboard, just like the 1982 product.
This is a great move for those who want to tinker with the Basic version of this machine and not just use it as a home console.
The final package will not include a joystick, but it will be possible to connect any type of USB controller or pendrive to the ports and use it without any problems.

The titles included are as follows:
- Alien Girl (Skirmish Edition)
- Ant Attack
- Army Moves
- Auf Wiedersehen Monty
- Avalon
- Bobby Bearing
- Cosmic Payback
- Devwill Too ZX
- Exolon
- Fairlight
- Firelord
- Football Manager 2
- Freddy Hardest
- The Great Escape
- Head Over Heels
- Highway Encounter
- The Hobbit



Fig. 1 - THE SPECTRUM in its future packaging and manual



Fig. 2 - THE SPECTRUM: the keyboard is reproduced complete with "rubber" keys

- Horace Goes Skiing
- Jack the Nipper
- Knot in 3D
- The Lords of Midnight
- Manic Miner
- Match Day II
- Movie
- Nodes of Yesod
- Penetrator
- Phantis (Game Over II)
- Pheenix
- Pyracurse
- Quazatron
- Robin of the Wood
- Saboteur! Remastered
- Shovel Adventure
- Skool Daze
- Snake Escape
- Spellbound
- Starquake
- Starstrike II
- El Stompo

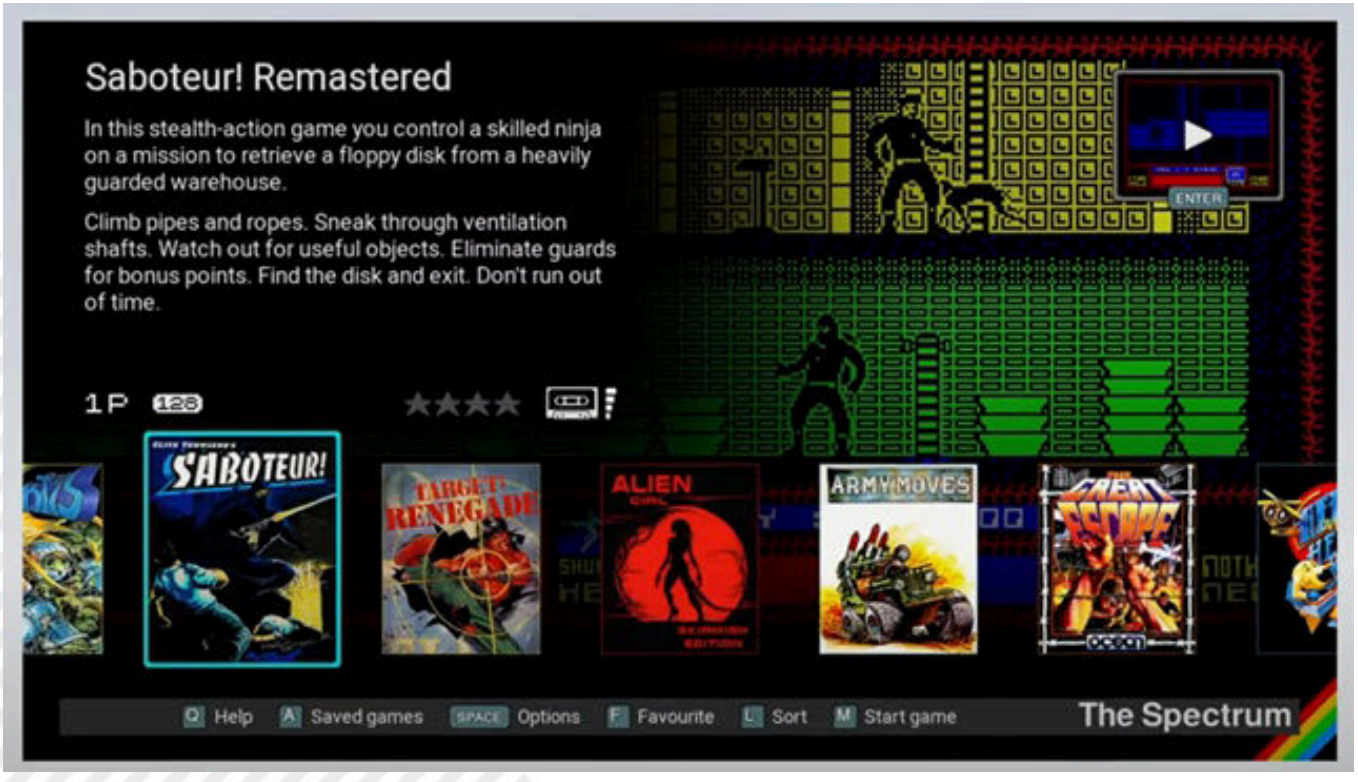


Fig. 3 - The introduction carousel with 48 pre-installed titles



Fig. 4 - New titles can be added via USB

- Stonkers
- Target: Renegade
- TCQ
- Technician Ted – The Megamix
- Tenebra
- Trashman
- The Way of the Exploding Fist
- Wheelie
- Where Time Stood Still

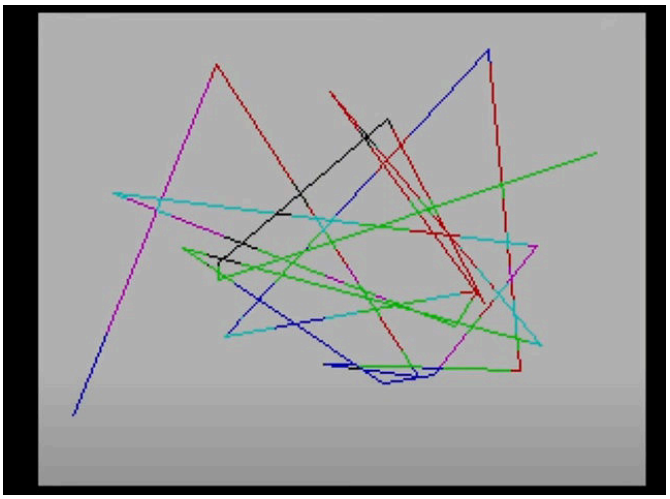


Fig. 5 - BASIC will also be included

A nice list with some gems that I will gladly play again. All titles will be presented with an official license.
All ZX Spectrum models from 16k to 128k will be emulated, you will be able to map the game controls with a special interface, and of course, you will be able to load additional old and new titles via USB flash drive.
This will allow you to play some small masterpieces such as Seraphima, Alien: Neoplasma, or Mighty Final Fight.

I personally can't wait to get my hands on it and test it for a more in-depth review.
These productions, such as the reissues of the 2600 or the future 7800+, are very welcome. The market is expanding and there is room for everyone.

Thanks to Retro Games LTD for the photos and press kit.

The Nintendo 64

by Bimbomillo



The Nintendo 64 (abbreviated to N64) is a 64-bit video game console produced by Nintendo from 1996 to 2002 and successor to the Super Nintendo Entertainment System (SNES).

In Italy, the console was released in 1997 by the Giochi Preziosi Group under the GIG brand.

When the Super NES era was finally coming to an end, the big software houses began to think about the next generation of consoles, and while many companies opted for the transition to 32-bit, Nintendo chose instead to move directly to 64-bit. Thus, after several years of development, the Ultra 64 was born, later changed to Nintendo 64, as Konami was already using this term for its Ultra Games division.

Nintendo's decision to use cartridges for this console as well, justified by the desire to prevent piracy, was criticized by those who expected a futuristic product on par with its rival PlayStation. The PlayStation could play music CDs in addition to games, offering multifaceted entertainment that particularly appealed to consumers.

In addition, the low production cost of CD-ROMs allowed Sony to resell games that were no longer top sellers at reduced prices, something that Nintendo could not do due to the high manufacturing cost of cartridges.

The Nintendo 64 marked a real revolution in the field of video games, offering the first true three-dimensional gaming experience with high-quality titles that are still considered milestones in their genre. It was a worldwide success, selling over 30 million units.

However, in its final years of sales, it was overtaken by its rival PlayStation, which set a record by selling over 70 million units, but with results significantly higher than those of the Sega Saturn, whose sales reached just over 9 million units.

The development of Nintendo 64 applications ended on April 30, 2002.

The machine looks like a black rectangular box (but special editions in different colors were also released over the years) with rounded edges and a slot on the top for inserting game cartridges and four connectors for gamepads on the front, without the need to purchase external peripherals.

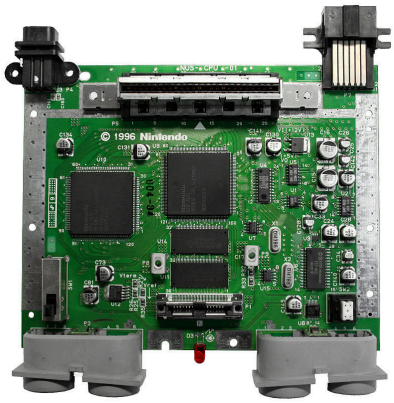
The console works with cartridges that are inserted directly into the machine. This system allows less data to be stored than CD-ROMs, but it allows for minimal loading times and the ability to save without the aid of memory cards, featuring a cartridge recognition system via an electronic circuit called Checking Integrated Circuit flanked by a Serial Peripheral Interface bus.

It has four gamepad outputs, so you can play multiplayer titles for four players without having to buy additional accessories such as Sony's Multitap.

There is a slot under the gamepad that allows you to insert additional peripherals. There is a slot at the front of the console for inserting a new RAM memory bank. Unlike its predecessor (the SNES), the Nintendo 64 cannot output an RGB video signal, and, in the European version, not even S-Video. This means that a European N64 has composite video as its best possible video output signal. Underneath the console is a slot used to connect the console to the Nintendo 64DD, a hardware expansion released only in Japan.

During the Nintendo 64's life cycle, several official

add-ons were produced for the console:
Controller Pak: a 256 IO memory that was inserted under the pad, divided into 123 "pages." Nintendo later realized the small size of this memory card and decided to create other versions ranging in size from 1 to 4 MB.
Expansion Pak: this is a 4 MB RDRAM (Rambus DRAM) bank that sits alongside the original 4 MB bank mounted on-board, allowing for 8 MB of RAM, enabling high game definitions and many other improvements. Although it is supported by several games, some of them can only work if the Expansion Pak is installed on the console, while others require the Expansion Pak to activate certain extra game features.
Rumble Pak: this is an accessory that, when inserted into the gamepad, vibrates during gameplay in response to events generated in the game environment. This accessory is now considered a standard feature available for all latest-generation consoles. It is essentially a motor powered by two AA batteries.
Transfer Pak: This accessory, when inserted into the controller, allows game data to be transferred between the Nintendo 64 and the Game Boy, Nintendo's portable console. In reality, this accessory had only one use, which was to transfer images captured with the Game Boy Camera to the Nintendo 64, but it was later used in PokÃ©mon Stadium games to transfer data from the Game Boy to the Nintendo 64.



The Nintendo 64 motherboard with the CPU, RCP (Reality Co-Processor) and RDRAM



Nintendo 64 Game Paks open and closed

Technical specifications

The Nintendo 64 motherboard with 1st CPU, I'RCP (Reality Co-Processor) and 1st RDRAM Central Processing Unit:
- NEC VR4300-64 bit with 4.6 million transistors @93.75 MHz
- 24 kB first-level cache
- MIPS / R4300i RISC architecture
- Total CPU math capacity: 93.0 million operations per second
- 0.35 um (micrometers) manufacturing process
Graphics Processing Unit:
- Silicon Graphics-RPC Reality 64-bit coprocessor @62.5 MHz 8-bit integer vector unit
- Peak graphics rendering of 150,000 polygons per second with all processing, 600,000 monochrome polygons per second.

Processing:
- Z-buffering
- Anti-aliasing
- Texture mapping

- Linear filter (bilinear and trilinear)
- Mip-mapping
- Gouraud shading
- Pixel fill rate of 30 megasamples per second with Z-buffer
- 16.7 million colors (32,768 on screen)
- Resolutions: 320 x 240/640x480 pixels

- GPU performance: 600 MegaFlops

Memory:
- 4 MB RDRAM
- System Bandwidth @562.5 MB/second
- Latency of 640 nanoseconds

Sound:
- 16-bit stereo
- 100 linear PCM channels (maximum 16-24 high quality). Each channel occupies an entire CPU cycle
- Sampling frequency at 48.0 kHz
- Supported formats: MIDI, MP3, ADPCM, and Tracker

Storage media:
- 4 to 64 MB electronic cartridges

References:

- *https://en.wikipedia.org/wiki/Nintendo_64*
- *https://it.wikipedia.org/wiki/Nintendo_64*

All trademarks represented belong to their respective owners.

Donkey Kong on PC128 Olivetti Prodest
Paolo Cattaneo's new retro project

by Paolo Cattaneo and Carlo Nithaiah Del Mar Pirazzini

Paolo Cattaneo is a talented developer. Captivated by the potential of ugBASIC, he pulled a really interesting port out of his hat that shows how "it can be done..." is the real formula behind the new wave of retrogaming.
But let's hear from him and delve into his world.

The Olivetti Prodest PC 128

Owning an Olivetti Prodest PC128 in 1986 or thereabouts was like being part of an Indian reservation.
Through some obscure and bizarre commercial agreement, Olivetti got its hands on the French Thomson MO6 (which was a huge flop in its home country) and resold it as its own product after slapping a nice ad hoc Italian logo on it and convincing the director of Commodore Italia to switch sides.
For some equally obscure reason, the attempt succeeded in persuading many more parents than it did across the Alps, and the machine ended up on the desks of many teenagers across the peninsula, including yours truly, who naively believed they could experience the same joys as their friends who owned Commodore 64s or similar machines.
What prompted Olivetti to market a machine so outdated in its specifications even for the year it was released has never been understood, but among a wide range of competing microcomputers boasting endless libraries of gamesâ€"many of which were conversions of the latest arcade titlesâ€"the PC 128 offered a decidedly more limited software library in terms of both quality and quantity.
The few big names on the machine for us Zzap! readers were Arkanoid and Green Beret, and it only takes two seconds on YouTube to see how they performed: the developers did their best in the short time they had, but unfortunately, the machine was what it was.

We are talking about a 1MHz Motorola 6809E with no auxiliary chips, not even for audio, managed by a poor 1-bit beeper in sync with the processor; As if that weren't enough, the latter has a fatal lack of register-to-register operations: you have to go through the RAM to compare registers A and B (both 8-bit), which takes at least 10 CPU cycles, when the optimal would be to do it in 1 or 2 cycles (like all immediate operations such as CMPA, SUBB, etc.). Even swapping registers A and B costs a whopping 8 cycles.
In short, moving things around on the screen on this machine means dealing with disarming slowness if you don't adequately limit the number of pixels that need to be redrawn every frame.
And so, returning to the Indian reservation, budding home gaming adventurers like me soon realized that their machine could never match even a Spectrum in terms of speed and action: the Spectrum certainly had outdated graphics, but it ran Chase H.Q. and Metro Cross with a fluidity that a PC128 could never dream of.
Anyone who owned one will undoubtedly have experienced at least once the frustration I'm talking about, that inability to invite your friend with the C64 to your house because he would simply laugh in your face.



Fig. 1 - Olivetti Prodest PC 128 S in all its glory

Many PC 128 buyers, once they realized this, simply decided to change machines. In my case, that wasn't possible.

My father had specifically chosen the PC 128 so that I could learn to program. He would never have tolerated the idea of buying me a "console disguised as a computer" that "would only take time away from my studies," and since the company he worked for had started offering internal BASIC courses, he was the one who taught me the basics of the language. All things considered, I have to thank him for his foresight: I don't know if choosing another computer would have prevented me from becoming the full-time programmer I am today, but that Indian reservation certainly allowed me to quickly discover the beauty of this art.

The resident SIMIV BASIC 128, developed by Microsoft and very versatile, also played a big part in all this, practically a forerunner of what would later become GWBasic. In short: once I accepted the fact that it wasn't the right machine for gaming, a programmer had a lot to enjoy, thanks in part to the more than 100KB available for his listing!

Of course, my teenage years on the PC 128 never led me to produce anything truly remarkable, partly because programming certain types of video games required knowledge of topics that were rarely dealt with in such a technical way in our country. But even when I stopped using it a couple of years later to switch to an IBM 8088, I decided that I would never part with my first computer: sooner or later, I would make a "proper" game on it, wanting to reclaim that part of me that had hoped for a machine comparable to others, capable of providing great gaming satisfaction.

The modern era and the advent of ugBasic

The story picks up about ten years ago, when the first online communities dedicated to retro development on Thomson and therefore also on the French twin of the PC 128 began to emerge.

Finally, it was an opportunity to access a wealth of documentation from that era, which had always been confined to France, including assembler manuals and



Fig. 2 - Asmshooter - a demo for a shooter written entirely in ASM 6809

technical diagrams with descriptions of each memory bank. In a way, it was like starting from scratch, but in a more professional way: I devoted myself to ASM 6809, and with a little effort, I managed to create a few small demos that finally ran smoothly.

I also realize that developing an entire game in ASM would be very time-consuming and methodical: a small error discovered too late could mean having to reconsider all the work, and errors in ASM are also quite difficult to find as the program gets bigger and bigger. Furthermore, the community of participants and users of various homebrews on this machine is not particularly large, so at this time I continue to develop only small demos, some for the challenges of retro programming Facebook groups.

Then, in 2021, came what for me was the decisive turning point: ugBasic, the framework created by Marco Spedaletti, was finally able to compile for the PC 128. This is a turning point because it allows programmers to focus solely on business logic without being discouraged by the length and difficulty of the assembler: for example, it is no longer necessary to write the drawing sequences that put the value in the accumulator and then in the desired video memory cell. The listing is written in high-level form, and the framework greatly simplifies the loading and drawing of assets, but the performance is that of machine language. At that point, without even the barriers posed by the

```
UPDATESHOTS
          * Check for start
          LDA MSTSHT
          CMPA #01
          BNE UPDSHTS
          JSR STARTSHOT
UPDSHTS
          * Update *
          LDX #$00
UPDATESHOT
          LDA SHOTS,X
          CMPA #$02
          LBLO UPDNXTSHT   .active
          LDD SHOTS+6,X    .get pos
          STD SHOTS+8,X    .upd pos
          LDA SHOTS+1,X    .frame
          CMPA #$00        .advance frame
          BEQ ADVSHTPOS
          LDA #$00
          STA SHOTS+1,X
          LDU #PLRSHT2     .load sprite
          BRA UPDSHTDONE
ADVSHTPOS
          * Advance position
          LDA #$01
          STA SHOTS+1,X    .advance frame
          LDU #PLRSHT1     .load sprite
          LDD SHOTS+6,X
          ADDD #$0001
          STD SHOTS+6,X
          LDA SHOTS+10,X   .offset x
          * Check right screen border
          CMPA #$26
          BGE SHTBORDER
          INC SHOTS+10,X   .increment
          BRA UPDSHTDONE
SHTBORDER
          LDA #$01
          STA SHOTS,X
          BRA UPDNXTSHT
UPDSHTDONE
```

Fig. 3 - ASMCODE: A portion of code for writing the asmshooter demo

programming method, there were really the means to achieve something substantial. So I thought back to my desire to create a convincing conversion of a famous game on this machine, and I identified what the candidate selection criteria should be. For performance reasons, the game had to be single-screen with a maximum of 800 pixels redrawable per frame; for experience reasons, the game had to have multiple levels; the background soundtrack during gameplay had to be absent or at least not play an important role; finally, the game had to be one that had not been previously converted.

The choice was made one evening when, intrigued by the mechanics of Mario running down slopes in Donkey Kong, I tried to replicate the same thing while tinkering with ugBasic, and succeeded sooner than I thought (which, knowing me, is definitely strange XD). Within a couple of hours, I had the entire first level ready, and it looked pretty convincing, so I decided that there was obviously no more suitable candidate, provided that I presented it only as a technical demo so as not to cause any legal implications (but I'll come back to that later).

Donkey Kong

So far, I have described the PC 128 as a slow machine, but it should be noted that its color palette remains, in my opinion, the most beautiful ever seen on an 8-bit computer, with a palette of 4096 shades to choose from to compose the range of 16 that will appear on the screen.

This technical excellence has made it possible to reproduce graphics that are fairly faithful to those of the coin-op, provided, of course, that compromises are made with the original technical specifications. All the sprites from the bar game have been more or less resized. For example, Kong in the arcade is up to 43px in length, while here he is always in a 40x32 frame. Mario is in a 16x16 frame but is actually 1 pixel shorter than the original, and all the levels have had to be compacted. Mario is smaller than he

```
' tapis roulant livello 3
IF (curLvl == 02 AND curPage == 01) THEN: reposPlayer[]:
    IF (yOnMap == 161) THEN: mvPlrH[-01]: ENDIF
    IF (yOnMap == 090) THEN: IF (plrX < 80) THEN: mvPlrH[01]: ELSE: mvPlrH[-01]: ENDIF: ENDIF
    IF (yOnMap == 062) THEN: mvPlrH[-01]: ENDIF
ENDIF

plrFrm = plrSFrm
IF (KEY PRESSED(#LEFT))  THEN: ftL = 13: ftR = 05: mvPlrH[-01]: plrSFrm = 04: plrJFrm = 07: ENDIF
IF (KEY PRESSED(#RIGHT)) THEN: ftL = 05: ftR = 13: mvPlrH[01]:  plrSFrm = 00: plrJFrm = 03: ENDIF
IF (timeHammer == 00) THEN
IF (KEY PRESSED(#SPACE)) THEN: isJumping = 01: jumpCount = -04: plrFrm = plrJFrm:
    isJX = 0: IF (KEY PRESSED(#LEFT) OR KEY PRESSED(#RIGHT)) THEN: isJX = 1: ENDIF
    jmpnt = 01: GOSUB progressJump: ENDIF: ENDIF
```

Fig. 4 - A bit of ugCODE, a snippet of ugBasic code, serves to show the difference in writing compared to ASM (high level vs. low level)

should be to give the eye more breathing space in the first level 1, where all 6 sloping planes have been brought closer together so as not to sacrifice any of them. The arcade has a resolution of 224x256x256 colors, which I had to translate into the 160x200x16 color mode of the PC 128. This mode has three advantages: 1) there is no color clash, 2) it uses two interchangeable video pages that avoid flickering, and 3) all screen coordinates can fit into one byte each, which speeds up calculations. The two video pages have an annoying peculiarity: not one but two positions must be tracked for each update, and this inevitably leads to using more variables than would be needed to erase and redraw.

The machine's memory is used by dividing it into a maximum of 26Kb for the code and almost three 16Kb banks for the graphics (only one of which is for the splash screen).

Each level of the game has its own particular characteristics: the first requires Mario to travel down slopes, and the barrels must do the same; the second has a moving platform that must transport Mario; the third has treadmills that must slow Mario down or speed him up; the fourth has platforms that disappear when Mario passes over them. To manage the levels, I used a matrix of values for each one: the game screen is 160x200, so a logical division into 10x20 tiles of 16x10 pixels would have been a good compromise compared to a 20x20 with tiles only 8 pixels wide, which would certainly have been more detailed but much more expensive in terms of memory.

This tile represents the way I divided the levels into boxes for this conversion.

Each value in the matrix therefore specifies the Y coordinate where Mario must place his feet (so Mario will be drawn at that coordinate, minus 16 pixels higher). So the matrix of the first level has values offset by one unit. Let's suppose that Mario has his feet on the Y coordinate 192. The row at that point in the matrix will be something like 192, 191, 190, 189, 188, etc., and we will see him climb from left to right. Similarly, the flat levels have rows with all the same values (192, 192, 192, etc.).

The stairs are managed with arrays: matrices would have

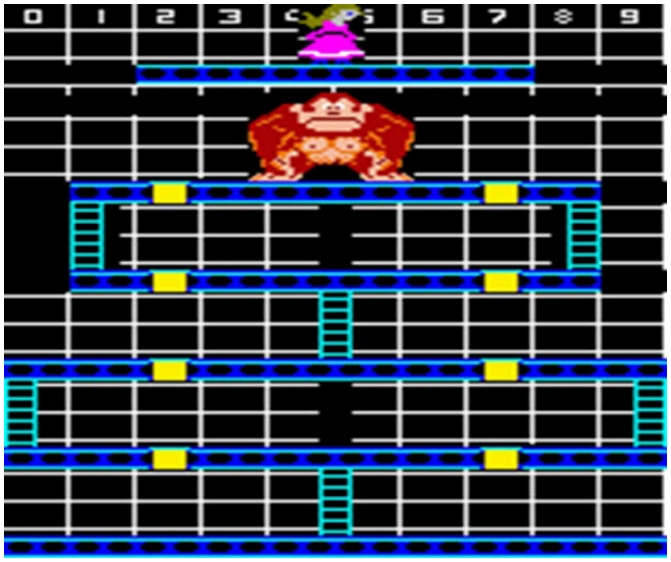


Fig. 5 -This tile represents how I divided the levels into boxes for this conversion

been fine too, but they are slower to access. With a view to optimizing everything, I opted for a maximum of 6 stairs, so two arrays of 6 values indicating the X and Y coordinates of the start and end of the stairs. The barrels must check for the presence of stairs above or below them every time they move, so the control cycles must be reduced to a functional minimum. Mario's detection costs less because it is only done when the up or down directional button is pressed, and once he has moved to the "on the stairs" state, he remains there until he moves and detects the end coordinate of a staircase, thus returning to the previous state.

For the mobile platform on the second level, I did the opposite: it checks that Mario is on top of it, and when this happens, it orders him to be pushed up once the necessary status checks have been made. The bouncing spring is not 16x16 but 8x16 pixels in order to optimize game speed.

The treadmills, on the other hand, work by invoking the directional walking routine without the player actually pressing any keys to move Mario: this is obviously only done when Mario is detected with his feet resting on the rollers, but this is only done on one of the two video pages, so that the other can be used to give the player a neutral frame where they can move Mario at will, which otherwise would see him remain stationary (all of Mario's horizontal movements are in fact 4 pixels).

The disappearing platforms are managed with a status



Fig. 6 - The title screen reminds us to "buy NINTENDO games"... Let's not make them angry!

that is activated when Mario walks on them. Once the status is activated, if Mario's coordinates are no longer within the limits of that platform, it disappears from the level (I draw the black block and the relative matrix is modified).

The splash screen was created by adapting the original Japanese flyer from 1981, brought to a resolution of 160x200x16 and reworked in pixels and colors.

As for the audio, there is not much to say: at the moment, ugBasic does not support sound commands, but it does support inline ASM code, so I used that solution. I first "played" the various tunes directly in BASIC 128 on an emulator, noted the values, and transferred them to the ASM code to set and play the beeper. The jump and bonus jingle sounds are played one step per frame with a score contained in an array.

I tried to include all 4 arcade levels as a top priority, at the expense of some shortcomings or inaccuracies that Marco and I are trying to fix. We will try to give the fire enemies behavior more faithful to that of the arcade, and to insert the bonuses and skits that separate the various levels. If everything goes as it should, we also hope to include the possibility of playing with Luigi and Popeye sprites.

Distribution and Nintendo legal guidelines

I took a look at the (long) list of fan games (like mine) that Nintendo has tried to block, often successfully, and I got the impression that, unlike most other game companies, Nintendo takes these things very, very seriously.

And at the risk of sounding boring, I'll tell you: I can understand them very well. It's not so much a question of profit, it's that their characters still bear the Nintendo name: therefore, if they are represented in an inappropriate way, Nintendo is represented in an inappropriate way, and Nintendo is the only one who can decide if and when the way is appropriate. I think that's how they see it, and I respect their view. When I consider the odyssey that the Kyoto giant had to endure with this game, which started out as a remedy for the failure of Radar Scope and ended up taking them to court against Universal, risking closure, I understand why they protect their products with samurai-like rigor.

With the passing of Mr. Iwata (sigh!), Nintendo has never updated its homebrew guidelines, so I am unable to distribute the binaries or source code to play this demo. I hope that something will soon change in Kyoto and that one day you too will be able to try it out for yourselves. In the meantime, of course, I invite you to play Donkey Kong in the thousand other (legal) ways possible.

And I thank you at Retro Magazine World for the wonderful opportunity to share this story! :)

Update

After promising it, he finally did it! Paolo managed to add the option to choose Luigi and Popeye in the game.

The title is really beautiful!

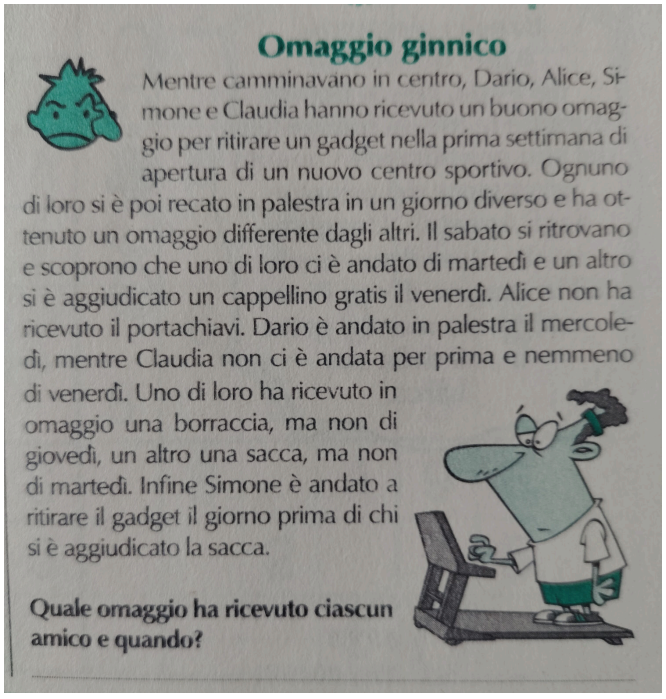
The C64 meets Artificial Intelligence

by Eugenio Rapella

Dario, Alice, Simone, and Claudia received a free voucher to pick up a gadget during the first week of a new sports center's opening. Each of them went to the gym on a different day and received a different gift from the others. On Saturday, they meet up and discover that one of them went on Tuesday and another got a free cap on Friday. Alice did not receive the key ring. Dario went to the gym on Wednesday, while Claudia did not go on the first day or on Friday. One of them received a free water bottle, but not on Thursday; another received a bag, but not on Tuesday. Finally, Simone went to collect his gift the day before the person who received the bag.

What gift did each friend receive and when?

In Italian magazines, questions like this (in fig. 1, the original text from issue 987 of "Settimana SUDOKU") are often labeled as "logic questions." To be honest, I find them rather boring and certainly don't waste my time trying to solve them.



Omaggio ginnico

Mentre camminavano in centro, Dario, Alice, Simone e Claudia hanno ricevuto un buono omaggio per ritirare un gadget nella prima settimana di apertura di un nuovo centro sportivo. Ognuno di loro si è poi recato in palestra in un giorno diverso e ha ottenuto un omaggio differente dagli altri. Il sabato si ritrovano e scoprono che uno di loro ci è andato di martedì e un altro si è aggiudicato un cappellino gratis il venerdì. Alice non ha ricevuto il portachiavi. Dario è andato in palestra il mercoledì, mentre Claudia non ci è andata per prima e nemmeno di venerdì. Uno di loro ha ricevuto in omaggio una borraccia, ma non di giovedì, un altro una sacca, ma non di martedì. Infine Simone è andato a ritirare il gadget il giorno prima di chi si è aggiudicato la sacca.

Quale omaggio ha ricevuto ciascun amico e quando?

Can we do anything with our Commodore? The C64's Basic doesn't seem to be the most suitable language for tackling our question (what about Prolog?).

I had fun finding a way to describe the situation to the C64; once I had done that, I let it get on with the job and... it didn't disappoint!

```
100 dim p1(24),p2(24):print chr$(5)
110 for k=1 to 24:read p1(k):p2(k)=p1(k):next
120 data 1234,1243,1324,1342,1423,1432
130 data 2134,2143,2314,2341,2413,2431
140 data 3124,3142,3214,3241,3412,3421
150 data 4123,4132,4213,4231,4312,4321
160 dim n$(4),day$(4),gad$(4),day(4),gad(4)
170
n$(1)="dario":n$(2)="alice":n$(3)="simone":n$(4)="cla
udia"
180
day$(1)="Tue.":day$(2)="Wed.":day$(3)="Thu.":day$(4)
="Fri."
190 gad$(1)="cap":gad$(2)="keyring"
200 gad$(3)="water bottle":gad$(4)="bag"
210 for h=1 to 24:for k=1 to 24
220 v1=int(p1(h)/10):day(1)=p1(h)-10*v1
230 v2=int(v1/10):day(2)=v1-10*v2
240 day(4)=int(v2/10):day(3)=v2-10*day(4)
250 v1=int(p2(k)/10):gad(1)=p2(k)-10*v1
260 v2=int(v1/10):gad(2)=v1-10*v2
270 gad(4)=int(v2/10):gad(3)=v2-10*gad(4)
280 for a=1 to 4
290 if gad(a)=1 then x=a
300 next
310 if day(x)<>4 then 510
320 if gad(2)=2 then 510
330 if day(1)<>2 then 510
340 if day(4)=1 or day(4)=4 then 510
350 for a=1 to 4
360 if gad(a)=3 then x=a
370 next
380 if day(x)=3 then 510
390 for a=1 to 4
400 if gad(a)=4 then x=a
410 next
420 if day(x)=1 then 510
```

```
430 for a=1 to 4
440 if gad(a)=4 then x=a
450 next
460 if day(x)-1<>day(3) then 510
470 print
480 for w=1 to 4
490 print n$(w),day$(day(w)),gad$(gad(w))
500 next
510 next:next
```

To begin with, let's load the names of the four friends into the vector N$(..); the names of the days of the week (only those we need) into DAY$(..) and, finally, the names of the gadgets into the vector GAD$(..).
The vectors are sized to 160 and the assignments are made in instructions 170 to 200.
So:
DARIO > 1; ALICE > 2; SIMONE > 3; CLAUDIA > 4.

TUESDAY > 1; WEDNESDAY > 2; THURSDAY > 3; FRIDAY > 4.

CAP > 1; KEY RING > 2;
WATER BOTTLE > 3; BAG > 4.

Each character goes to the gym on a different day and receives a different gift. Before considering other limitations, this means that a "possible solution" will be something like:

```
NAME:      1     2     3     4
++++++++++++++++++++++++++
DAY:       3     1     2     4
GADGET:    4     1     3     2
```

In this example, DARIO (1) goes to the gym on THURSDAY (3) and receives the BAG (4); ALICE (2) goes on TUESDAY (1) and receives the CAP (1), etc.
The first row, the one with the names, is fixed: 1,2,3,4.
In the second row, there will be a permutation of the numbers from 1 to 4, so that everyone goes to the gym on a different day. In the third row, there will also be a permutation of the numbers from 1 to 4 so that the friends receive different gifts.
The basic idea is to match each of the permutations of days (there are 24 permutations of four elements) with each of the permutations of gifts.
We will then have to find a way to describe the limitations indicated in the text in order to "filter" the "candidate solutions" (which, at the beginning, are 24*24=576).
In the end, only those that meet the requirements will be printed, and if the problem is well posed, ... exactly one will remain: the solution.
The 24 permutations of the numbers 1,2,3,4 are loaded into the vectors P1(..) and P2(..) at the beginning of the program (instructions 100-150). We could have asked the C64 to generate them itself, but there are so few that it is easier and faster to provide them directly.
For convenience, I did not use P1(0) and P2(0); at the end of the FOR-NEXT of instruction 110, we will have P1(1)=1234, P1(2)=1243, P1(3)=1324, ..., P1(24)=4321, and the same for P2(..).

The actual program begins at instruction 210 with the two nested FOR loops that close with the two NEXT loops of the last instruction (510).
For each H from 1 to 24, the permutation of position H is taken from the vector P1(..) and the four digits that compose it are assigned to the vector DAY(..) which has been sized to 160. This assignment takes place in the three instructions 220-240; if, for example, P1(H)=2341, we will have V1=INT(P1(H)/10)=INT(234.1)=234.
DAY(1)=P1(H)-10*V1=2341-10*234=1.
V2=INT(V1/10)=INT(234/10)=23; DAY(2)=234-10*23=4;
DAY(4)=INT(V2/10)=INT(23/10)=2
and DAY(3)=V2-10*DAY(4)=23-10*2=3.
Ultimately, if P1(H)=2341, we obtain DAY(1)=1, DAY(2)=4, DAY(3)=3, DAY(4)=2: the digits of the permutation are transferred to the DAY(..) vector.
The purpose of the DAY(..) vector is to associate the day of the week with the character: if DAY(2)=3, it means that the day associated with ALICE (2) is THURSDAY (3).
The same applies to the permutation extracted from the P2(..) vector: if P2(K)=1243, we will have GAD(1)=3, GAD(2)=4, GAD(3)=2, and GAD(4)=1. With the vector GAD(..), we associate an object with the character: if GAD(4)=1, it means that we associate the HAT (1) with CLAUDIA (4).

If and when they are reached, instructions 470-500 print the names of friends (N$(..)) associated with the names of days (DAY$(..)) corresponding to DAY(..) and the names

of objects (GAD$(..)) corresponding to GAD(..).
If you delete all instructions from 280 to 460 and run the program, the C64 diligently lists the names of friends matched with the permutations of days, and each of these is matched with all possible permutations of gadgets. There are 24*24=576 lists: one of these lists is the solution to our problem.
How do we find it? Here's the fun part: the messages contained in the text are transformed into a series of IFs: as soon as a request is not met, it immediately jumps to 510 to examine a new candidate. We will only reach 470 if all requests are met!

Let's follow the text:
[1] SOMEONE WON THE CAP (1) ON FRIDAY (4):

```
280 for a=1 to 4
290 if gad(a)=1 then x=a
300 next
310 if day(x)<>4 then 510
```

The 280-300 cycle allows us to determine the number X of the person who has the cap. At 310 we check: if the day associated with X is not Friday, our condition is not satisfied and we can go directly to 510: the "candidate" cannot be the solution.

++

[2] ALICE (2) DID NOT RECEIVE THE KEYRING (2):

```
320 if gad(2)=2 then 510
```

In other words: if the gadget associated with Alice is the key ring, the one under consideration cannot be the solution; let's move on to another candidate.

++

[3] DARIO (1) WENT TO THE GYM ON WEDNESDAY (2):

```
330 if day(1)<>2 then 510
```

If the day associated with Dario is not Wednesday ... we're not there yet, so let's move on.

++

[4] CLAUDIA (4) DOES NOT GO TO THE GYM ON THE FIRST DAY (1) OR ON FRIDAY (4):

```
340 if day(4)=1 or day(4)=4 then 510
```

If she went on Tuesday or Friday, this cannot be the solution.

++

[5] WHO RECEIVED THE WATER BOTTLE (3) DID NOT RECEIVE IT ON THURSDAY (3):

```
350 for a=1 to 4
360 if gad(a)=3 then x=a
370 next
380 if day(x)=3 then 510
```

At the end of cycle 350-370, X contains the number of the person who received the water bottle. If the day corresponding to X is Thursday, the request is not fulfilled and we must move on.

++

[6] THOSE WHO RECEIVED THE BAG (4) DID NOT RECEIVE IT ON TUESDAY (1):

Proceeding as in the previous step:

```
390 for a=1 to 4
400 if gad(a)=4 then x=a
410 next
420 if day(x)=1 then 510
```

++

[7] SIMONE (3) WENT TO PICK UP THE GADGET THE DAY BEFORE THE PERSON WHO WON THE BAG (4):

```
430 for a=1 to 4
440 if gad(a)=4 then x=a
450 next
```

460 if day(x)-1<>day(3) then 510

At the end of cycle 430-450, X contains the number of the person who got the bag. DAY(X)-1 is the day before the one used by X; if this day is not the one corresponding to Simone, nothing happens, we move on...

++++++++++++++++++++++++++++++++++++

Given the RUN, the C64 takes about a minute to provide a solution and another minute to complete the analysis without adding other solutions. The solution proposed by Commodore is unique and coincides with the one reported in the magazine (see fig. 2).

I took advantage of the program to conduct a small experiment: what happens if one (or more) conditions are ignored? Logically, if a condition is eliminated, at least one alternative solution should be obtained in addition to the original solution.
If we eliminate condition [1], the C64 produces a total of three solutions; if we eliminate [3], the solutions become six. By deleting [5], the C64 reports two solutions; by eliminating [7], the solutions are five.
The curious thing is that if we eliminate [2], the C64 provides only the original solution: this means that the information "Alice did not receive the key ring" is not essential information, it is... redundant.
The same happens if we eliminate condition [4] or [6], but also if we eliminate [4] and [6] at the same time! There is only one solution (the usual one) even if [2] and [6] are ignored at the same time, while if both [2] and [4] are deleted, or [2]+[4]+[6], two solutions are obtained (the more masochistic among you can try other combinations... I'll give it a miss).

I submitted the original text of the question to ChatGPT and received a very detailed response that concluded with

Final solution:
- Tuesday: Simone â€“ Water bottle.
- Wednesday: Dario â€“ Key ring.
- Thursday: Claudia â€“ Bag.
- Friday: Alice â€“ Cap.
This solution meets all the conditions of the problem

I pointed out to ChatGPT that its solution did not meet the last condition ("Simone went to pick up the gadget the day before the person who won the bag"); the Artificial Intelligence replied:

I apologize for the error. Let's carefully review the problem, making sure to comply with all the conditions and find a correct solution.

This went on for quite a while: ChatGPT proposed an incorrect solution; I pointed out the error and ChatGPT proceeded with a new proposal with variations, but without progress.
In the end, I pointed out to ChatGPT the solution found by Commodore and invited it to verify that it did indeed meet all the requirements.
Here is ChatGPT's final response:
"Thank you for your patience and for guiding me to the correct solution."
You probably didn't expect our beloved Commodore to give the AI a hand...

Addendum

This part doesn't say much that is new; I am including it for the sake of completeness, and you can certainly skip it.
The days of the week mentioned in the text are Tuesday to Friday (Saturday is excluded as it is the day when the kids all get together). There are four days, just as there are four friends and four gadgets assigned: this is probably the correct interpretation and is the one I used in the previous discussion.
Logically, "Monday" could also be included: the fact that

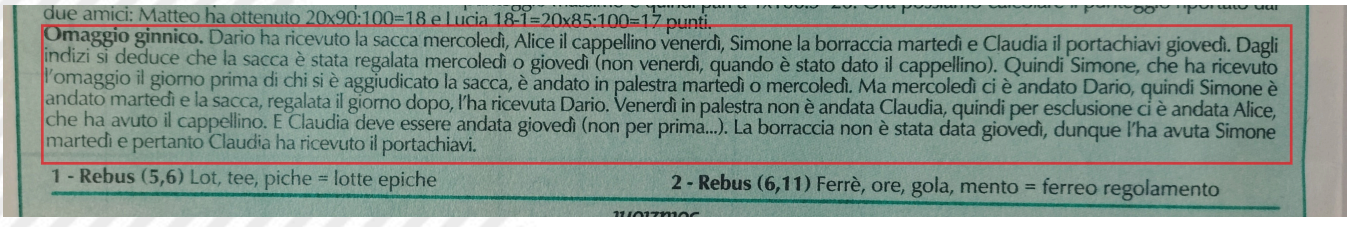

due amici: Matteo ha ottenuto 20x90:100=18 e Lucia 18-1=20x85:100=17 punti.
Omaggio ginnico. Dario ha ricevuto la sacca mercoledì, Alice il cappellino venerdì, Simone la borraccia martedì e Claudia il portachiavi giovedì. Dagli indizi si deduce che la sacca è stata regalata mercoledì o giovedì (non venerdì, quando è stato dato il cappellino). Quindi Simone, che ha ricevuto l'omaggio il giorno prima di chi si è aggiudicato la sacca, è andato in palestra martedì o mercoledì. Ma mercoledì ci è andato Dario, quindi Simone è andato martedì e la sacca, regalata il giorno dopo, l'ha ricevuta Dario. Venerdì in palestra non è andata Claudia, quindi per esclusione ci è andata Alice, che ha avuto il cappellino. E Claudia deve essere andata giovedì (non per prima...). La borraccia non è stata data giovedì, dunque l'ha avuta Simone martedì e pertanto Claudia ha ricevuto il portachiavi.
1 - Rebus (5,6) Lot, tee, piche = lotte epiche
2 - Rebus (6,11) Ferrè, ore, gola, mento = ferreo regolamento

Fig. 1 - The solution as it appeared in the magazine (in Italian...)

it is never mentioned does not mean that someone could not have gone to the gym on Monday and that one of the other days of the week remains uncovered.
To be on the safe side, I tried to vary the program in this sense by changing the numbering of the days: Monday>1; Tuesday>2; Wednesday>3; Thursday>4: Friday>5.

Here is the result:

```
100 dim p1(24),p$(120):print chr$(5):kx=1
110 for k=1 to 24:read p1(k):next
120 for k=1 to 24: m$=str$(p1(k))
130 for ww=1 to 5
140 n$=left$(m$,ww)+"5"+right$(m$,5-ww)
150 p$(kx)=n$:kx=kx+1:next ww
160 next k
170 data 1234,1243,1324,1342,1423,1432
180 data 2134,2143,2314,2341,2413,2431
190 data 3124,3142,3214,3241,3412,3421
200 data 4123,4132,4213,4231,4312,4321
210 dim n$(4),day$(5),gad$(4),day(5),gad(4)
220
n$(1)="dario":n$(2)="alice":n$(3)="simone":n$(4)="clau
dia"
230     day$(1)="Mon.":day$(2)="Tue.":day$(3)="Wed."
240 day$(4)="Thu.":day$(5)="Fri."
250 gad$(1)="cap":gad$(2)="keyring"
260 gad$(3)="water bottle":gad$(4)="bag"
270 for h=1 to 120:for k=1 to 24
280 for tt=1 to 5
290 day(tt)=val(mid$(p$(h),tt+1,1)): next tt
300 v1=int(p1(k)/10):gad(1)=p1(k)-10*v1
310 v2=int(v1/10):gad(2)=v1-10*v2
320 gad(4)=int(v2/10):gad(3)=v2-10*gad(4)
330 for a=1 to 4
340 if gad(a)=1 then x=a
350 next
360 if day(x)<>5 then 610
370 if gad(2)=2 then 610
380 if day(1)<>3 then 610
390 if day(4)=1 or day(4)=5 then 610
400 xa=0
410 for a = 1 to 4
420 if day(a)=1 then xa=1
430 next
440 if xa=0 and day(4)=2 then 610
450 for a=1 to 4
460 if gad(a)=3 then x=a
470 next
480 if day(x)=4 then 610
490 for a=1 to 4
500 if gad(a)=4 then x=a
510 next
520 if day(x)=2 then 610
530 for a=1 to 4
540 if gad(a)=4 then x=a
550 next
560 if day(x)-1<>day(3) then 610
570 print
580 for w=1 to 4
590 print n$(w),day$(day(w)),gad$(gad(w))
600 next
610 next:next
```

The permutations of the five days become 5!=120, and I obtained them by inserting the number 5 into each of the 24 permutations of the numbers 1-2-3-4, in the five possible ways (instructions 120-160). For example, the first permutation 1234 generates the first five elements of P$(..): 51234, 15234, 12534, 12354, 12345 (the STR$(NUMBER) function transforms the NUMBER into a string by adding a "space" at the beginning).
The vectors DAY$(..) and DAY(..) are sized to value 5 (DAY$(0) and DAY(0) are not used).

At 270, the two FOR loops of the main program open, and the FOR loop relating to the permutations of the days goes from 1 to 120. The permutation is loaded into DAY(..) by extracting the individual characters (with MID$(..)) and then converting them into digits (with VAL(..)).
The rest of the program does not differ greatly from the one already presented; naturally, the instructions with DAY(..)=N have been modified with the new numbering of the days.

The only substantial change is in the condition "Claudia did not go to the gym first." In the first program, this condition coincided with "Claudia did not go on Tuesday." Now, in addition to "Claudia went on Monday," another situation must be excluded: "if no one went to the gym on Monday, Claudia goes on Tuesday."

This is reflected in the instructions:

400 xa=0
410 for a = 1 to 4
420 if day(a)=1 then xa=1
430 next
440 if xa=0 and day(4)=2 then 610

In the situation under consideration, variable XA, initially zero, is set to "one" if someone went to the gym on Monday. If this does not happen and Claudia goes on Tuesday (inst. 440), then Claudia would be the first to go to the gym and the situation analyzed could not be the solution (go to NEXT in inst. 610).

In the first program, the group of instructions within the two nested FOR-NEXT loops was executed 24*24=576 times; now it is executed 24*120=2880 times, so the processing time is, of course, longer: about ten minutes to find the first solution and another six to complete the processing. Even in this version of the problem, the C64 finds the solution and proves that it is unique. Doesn't it deserve a standing ovation?

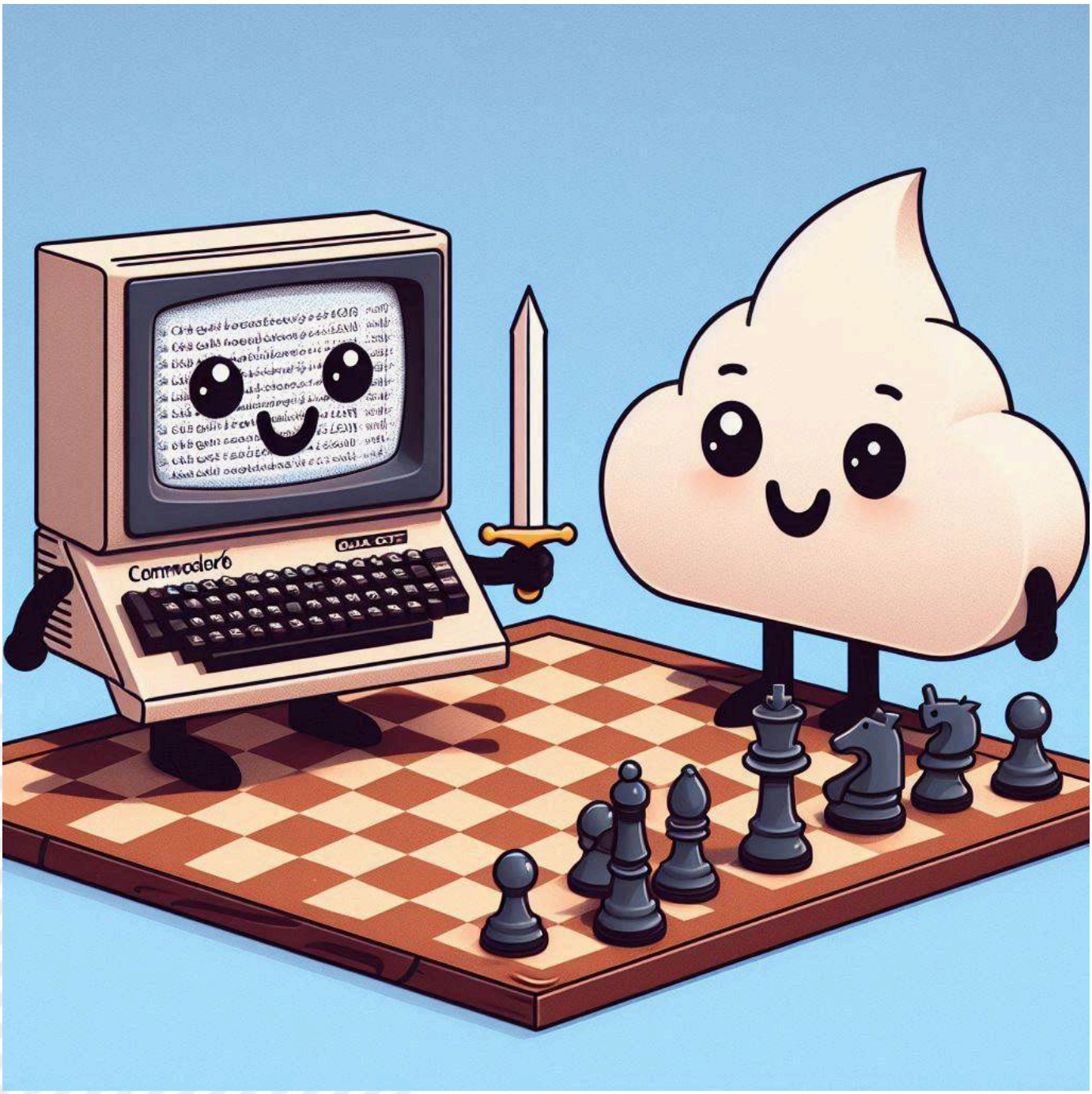


Fig. 2 - The image of the challenge imagined by Bing's AI

Zena

by Javier Gonzalez

The VIC1001 is a Japanese version of the VIC20 which was released several months earlier in 1980. It's basically an NTSC VIC20.

The game Zena for the VIC1001 was a type-in published in a Japanese magazine called IO, issue October 1983. The magazine can be found (for the moment) in the internet archive, it has cool photos of weird japanese micros and stuff.

Instead of the usual BASIC listing, this is a binary listing. The article makes reference to the machine code monitor cartridge VIC1213. There are some instructions too on how to execute the code. Fortunately it's easy to find the d64 online.

The article has some explanations about the game's inner workings. I tried to copy the text from the pdf and paste into Google Translator with mediocre results.



The program starts on a PAL VIC20 too, the game is playable. It runs on a +3K memory expanded machine. Apparently it is inspired by a game called Xevious. It can be found at the VIC20 forum on the internet (Denial forum).

Keyboard

I had trouble playing the game on the emulator, because the game uses the colon and semicolon keys to move the ship, and the shift key to shoot. That doesn't work out on my keyboard because the colon and semicolon characters are shifted keystrokes.

I decided to patch the program to change the keys. First I checked the use of keyboard matrix registers. I used VICE monitor's hunt command to try and find the sequence LDA $9111 in the RAM memory addresses. This yielded 0 results. Then I searched just the address, and the command listed 1 unrelated result.

On BASIC I always liked the trick WAIT 653,1 to wait for the user to press the SHIFT key. The fact the game uses the SHIFT key made me suspect the author had used the KERNAL keyboard routines. I hunted for 653, there were 1 or 2 results, one was spot on.

With the keyboard input routine located, it is easy to change the colon and semicolon keypresses to O, P. As a bonus I found it was easy to patch an autofire. Just override a test of the SHIFT key being released before the next shoot. Autofire is then achieved by pressing SHIFT/LOCK.

```
...
.C:0e09  AD 8D 02  LDA $028D  <- 653, SHIFT address
.C:0e0c  F0 0E     BEQ $0E1C
.C:0e0e  A5 05     LDA $05    <- $05 memory address is being used to test SHIFT key release
.C:0e10  D0 0C     BNE $0E1E  <- for autofire change this instruction to BIT
...
.C:0e1c  85 05           STA $05
...
.C:0e1f  A5 C5     LDA $C5    <- last keypress code
.C:0e21  C9 2D     CMP #$2D   <- colon keypress code
.C:0e23  F0 0B     BEQ $0E30
.C:0e25  C9 16     CMP #$16   <- semicolon keypress code
.C:0e27  D0 0C     BNE $0E35
```

. . .

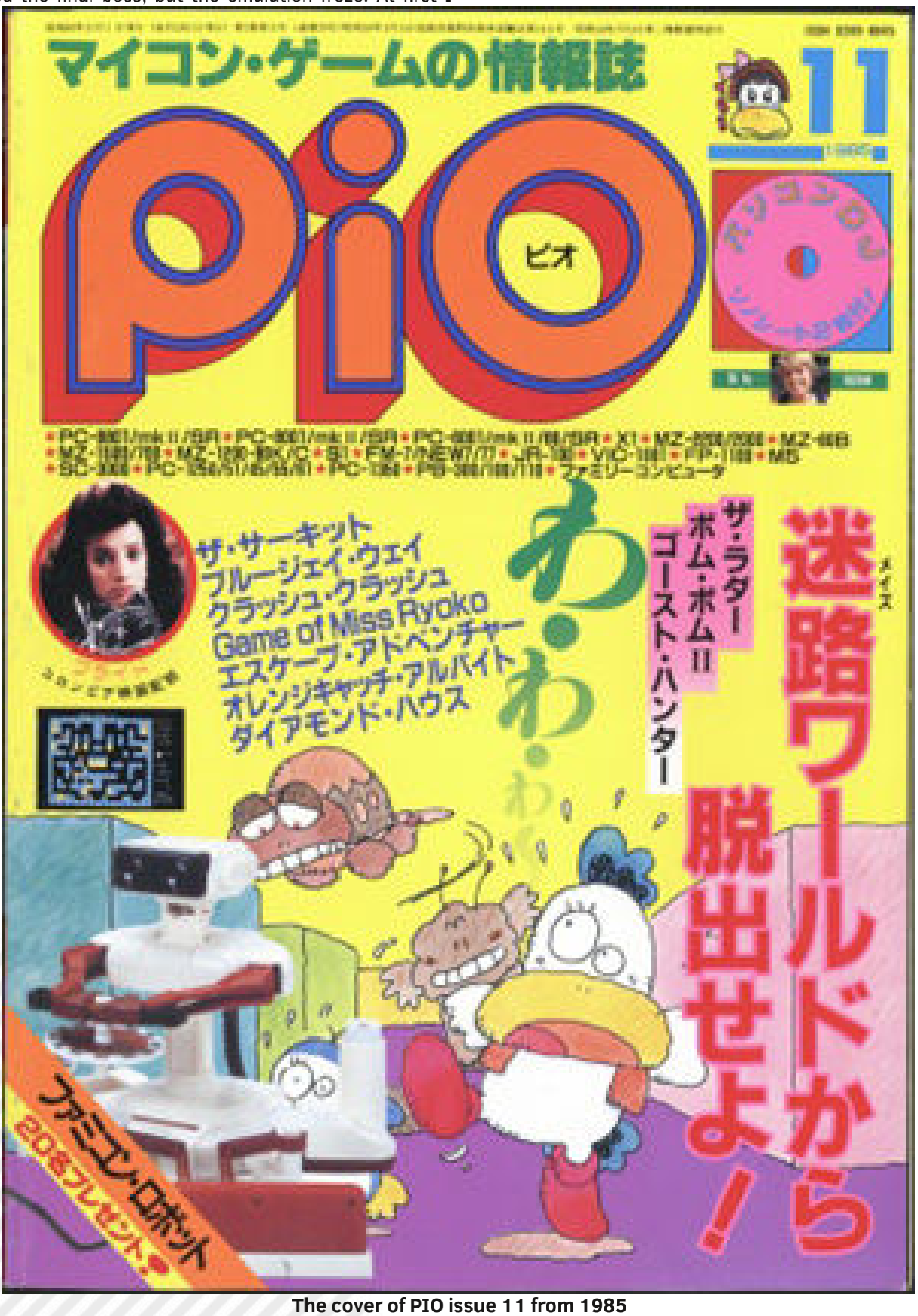
You can find keypress codes running this small BASIC program:

10 PRINT PEEK(197) : GOTO10

Emulation is not the real thing

Now I was playing the game like a real ace. I effortlessly reached the final boss, but the emulation froze. At first I suspected the three bytes I had changed had screwed up something. After a few times, I opened the VICE monitor and tracked an infinite loop which happened to be unimpressive.

```
.C:12e3  20 45 13  JSR $1345
```



The cover of PIO issue 11 from 1985

```
C:12e6  9D A7 03   STA $03A7,X
.C:12e9  D0 05     BNE $12F0
.C:12eb  BD A2 03  LDA $03A2,X
.C:12ee  F0 F3     BEQ $12E3
```

At the address $1345 I found something to blame. It was akin to a random number generator. Looks like the author used some mirror register related to the paddles.

```
.C:199f  A5 FB     LDA $FB
.C:19a1  6D 18 90  ADC $9018
.C:19a4  85 FB     STA $FB
.C:19a6  60        RTS
```

Thanks to my previous attempts to translate the article, I was reassured it was about random numbers and not some alternate game control with paddles.
$1994 random number

On the emulator PEEK($9018) yields always 0. I changed the address for $9007, the raster line. Now the game has a quite higher number of enemies on screen.

I decided to check $9018 on my NTSC VIC20 machine. On the real hardware this address works indeed as a random number generator. I don't know the actual reason.

The bug

There seems to be a bug in the original code. Occasionally a symbol is displayed in the leftmost column. It could be related to the kernel managing the cursor. But it doesn't seem to interfere with the gameplay and it may be difficult to track, so I'm out.

Number of lives

In the code explanation the author quotes a variable in the address:
3C6 galstar number

Galstar happens to be the player, this seems to be the number of lives. We can confirm this opening the monitor and adding a watch command w store 3c6. Then run the game, immediately the monitor pops up and shows the address of the code doing the variable initialization.

```
.C:0b48  A9 04    LDA #$04  <- this is the
initial number of lives
.C:0b4a  8D C6 03  STA $03C6
```

If you keep playing you eventually lose a life and then you find the code which decreases the number of lives.

```
.C:0dd8  CE C6 03  DEC $03C6 <- this decreases
the number of lives when hit
```

This is a binary decimal variable that can be increased up to 99.

Loader

I made a loader to run the game on my +37k expanded VIC from my SD2IEC. It doesn't start immediately to give a chance to enter the POKEs. Once run, press RETURN to load the binary, then press RETURN or move down cursor to enter or skip the POKEs.

```
5 poke 55,0:poke56,30:clr:poke648,30:sys58648
10 print"{clr}{7xcrsrd}
load"chr$(34)"zena2"chr$(34)","right$(str$
(peek(186)),1)",1"
15 print"{4xcrsrd}poke3618,52:poke3622,13:poke3600,36
20 print"{2xcrsrd}poke6562,7"
25 print"{2xcrsrd}sys2816{15xcrsru}"
```

The line with 3 POKES patches the two keys and then the autofire. The following line patches the random generator thing. I don't recommend patching the lives because the game becomes too easy and boring.

Resources

Zena - download of the game:
- https://drive.google.com/file/d/1AJb8lDOebOGjooisaADCcWc9GafCrJJ9/view?usp=sharing

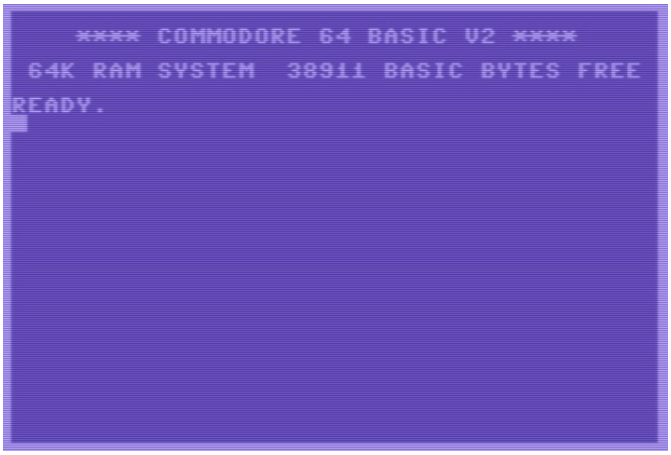
PIO magazine issue 11 from 1985:
- https://archive.org/details/pi-o-1985-11/page/136/mode/2up

Commdore 64 - BASIC V2

by Francesco Fiorentini

Some time ago, in a conversation with David La Monaca, we expressed our desire to create a series of articles that would shed light on the hundreds of Basic dialects of the Commodore 64. The task seems daunting, given the considerable amount of work that awaits us and, for some of them, the almost total lack of information. However, we decided to give it a try and...

...we couldn't help but start our review with the language that came with the machine. The Commodore 64 had version 2.0 of Microsoft Basic installed in its ROM.



BASIC, which stands for Beginner's All-purpose Symbolic Instruction Code, is probably one of the simplest third-generation programming languages. Many of the 8-bit machines of the 1980s were equipped with a version of this language because of its simplicity and ease of learning.

Basic V2, also known as Commodore Basic, was purchased by Jack Tramiel from Microsoft. Yes, that's right, Bill Gates' company. At the time, however, the balance of power between the two companies was clearly in Tramiel's favor. Commodore was a solid company, while Microsoft was still a small company constantly seeking funding. Emblematic of this is the story of the purchase of the Microsoft Basic licenses. Bill Gates wanted to sell his Basic for a royalty of $3 for each installation, but Jack Tramiel, confident in his advantageous position, rejected the offer with the famous phrase "I'm already married" and instead offered to purchase the package for a one-time payment of only $25,000 for a perpetual license. The exact figure is unknown, with various versions circulating with figures ranging from $10,000 to $30,000. What is certain, however, is that Jack Tramiel managed to get the better of Bill Gates for a decidedly paltry sum, given the incredible number of Commodore 64s sold!

As mentioned, Basic V2 is loaded into the Commodore 64's ROM and is immediately available when the machine is turned on. The interpreter is loaded and made available to the user, who can then start using Basic commands immediately or within a program. Note that the user has 38911 bytes of RAM available for Basic, which corresponds to the contiguous space defined between $0800-$9FFF (2048-40959).

Basic V2 commands

Basic V2 consists of 70 commands:

ABS, AND, ASC, ATN, CHR$, CLOSE, CLR, CMD, CONT, COS, DATA, DEF, DIM, END, EXP, FN, FOR, FRE, GET, GET#, GOSUB, GOTO, IF, INPUT, INPUT#, INT, LEFT$, LEN, LET, LIST, LOAD, LOG, MID$, NEW, NEXT, NOT, ON, OPEN, OR, PEEK, POKE, POS, PRINT, PRINT#, READ, REM, RESTORE, RETURN, RIGHT$, RND, RUN, SAVE, SGN, SIN, SPC, SQR, STATUS/ST, STEP, STOP, STR$, SYS, TAB, TAN, THEN, TIME/TI, TIME$/TI$, TO, USR, VAL, VERIFY, WAIT.

At first glance, 71 commands may seem like a lot, but upon closer inspection, we realize that Basic V2 is lacking a number of commands, especially those related to graphics and sound management. However, some of these shortcomings could be circumvented by reading or writing to memory using the PEEK and POKE instructions: since some hardware registers of the CPU, VIC-II, and SID were mapped directly to memory, it was possible to use these instructions to modify those registers and alter the behavior of the computer.

Unfortunately, the shortcomings of the Commodore 64's

BASIC were not limited to the lack of certain commands, but were also evident in terms of speed. Obviously, the comparison with machine language was merciless, but even when comparing Basic V2 with other Basic dialects of the same period, such as Amstrad CPC's Locomotive Basic, Microsoft's product came out badly. Add to this the limitation of two characters for variable names, and the crime is complete.

Variable names were limited to two characters in length, but no error was reported to the programmer if more characters were used.

Let's take an example:

```
10 PI=10
20 PIPPO=15
30 PINOCCHIO=18
40 PRINT PI
```

We would expect the result of this program to be 10. The variable PI, which is printed (PRINT) on line 40, is initialized with the value 10 on line 10. Instead, the result of the program is 18, precisely because of the aforementioned 2-character limit for variable names.

Be careful, however, because two variables with the same name but different data types remain distinct:

```
10 PI=10
20 PI$="123"
30 PRINT PI
40 PRINT PI$
```

The result of this program will correctly be 10 and "123". Certainly, the cases reported in the examples cannot be considered real shortcomings, but they give an idea of the level of attention required to write a program with the C64's Basic.

IF...THEN

Nowadays, we are automatically accustomed to associating ELSE with the IF...THEN construct, but unfortunately this is not the case in Basic V2, as this dialect lacks the ELSE operand. However, the lack of ELSE does not undermine the effectiveness of the code, but it does require the programmer to insert a series of additional checks to compensate for this shortcoming. Especially when porting programs from other Basic dialects, we must pay attention to the checks to be put in place if the source dialect implements else.

Tokens

For obvious reasons of optimization and space, Basic V2 commands and functions are not stored as the strings that appear on the screen. Instead, each of them is replaced by a unique code represented by an integer value, a byte, called a token.

During code execution, the token acts as an index to the jump table or vector table containing the memory addresses of the assembly routines that implement the BASIC commands. Conversely, at the time of a LIST, the token is used to read the full name of the basic command to be displayed on the screen from that same table.

Basic Extensions

It is immediately clear that, if we want to create extensions to the Basic V2 commands, we would have to interact with the tokens and the corresponding vector table in some way. It is also evident that the limitations mentioned so far have created fertile ground for a decisive proliferation of C64 basic extensions.

During the long life of Commodore's little gem, hundreds of programmers have developed dozens of these extensions, and in this overview we will try to list as many as possible, attempting to shed light on this interesting phenomenon that is almost exclusive to the Commodore 64!

Where to find it

The ROM containing BASIC V2 is generally supplied with Commodore 64 emulators such as Vice, Frodo, CCS64, Hoxs64, or Power64, to name just a few of the best known.

References

"C64 - Reference Guide for the Programmer" (1983)
"Commodore 64 User's Guide" (1983)
https://pickledlight.blogspot.com/p/commodore-64-guides.html

INFORM 6 - Tutorial 4

by Fredrik Ramsberg - adapted for RMW by Gianluca Girelli

PunyInform Coding 2: Objects and Actions

PunyInform is a tool for writing text adventures for 8-bit computers and newer computers. If you have already installed PunyInform and followed the PunyInform Coding 1 tutorial, it's time to learn more about objects and actions. This tutorial examines the different parts of a simple game in which there are two puzzles to solve, you can earn points, and you can win the game. My goal is to explain everything that was not covered in the previous tutorial.

The game starts with the player outside a house, where he thinks Uncle Billy's will might be. He must go through the locked front door and then through another locked door to get to the room where the will is located. That's the whole game. You can download the game code and the compiled game from this folder.

Declaring constants

```
1  !% -~S
2  !% $OMIT_UNUSED_ROUTINES=1
3  !% $ZCODE_LESS_DICT_DATA=1
4
5  Constant Story      "Uncle Billy's Will";
6  Constant Headline   "^An interactive paper hunt.^";
7
8  Constant STATUSLINE_SCORE; Statusline score;
9
10 Constant DEBUG;
11 Constant OPTIONAL_SIMPLE_DOORS;
12 Constant OPTIONAL_NO_DARKNESS;
13 Constant INITIAL_LOCATION_VALUE = Lawn;
14
15 ! Scoring
16 Constant OPTIONAL_SCORED;
17 Constant OPTIONAL_FULL_SCORE;
18 Constant MAX_SCORE 200;
19 Constant OBJECT_SCORE 20;
20 Constant ROOM_SCORE 15;
21
22 Include "globals.h";
23
24 Include "puny.h";
25
```

Game code part 1

A PunyInform game starts with a set of constants to indicate the name of the game, which optional parts of the library you want to use, and more. You typically need to declare all constants before including "globals.h".

In line 8, we declare that we want to show the current score (and number of moves) in the status line. The other option is to show the time in the status line, such as "3:00 PM." Be sure to always declare what kind of status line you want.

On line 11, we say that we want to use the Simple Door mechanism available in PunyInform. This simplifies the creation of doors.

For this game, we decided that we don't need the library to worry about light and darkness: we will assume that there is always light present in all locations. So, we define OPTIONAL_NO_DARKNESS on line 12, to exclude the parts of the library that handle light and darkness. This makes the library slightly smaller, and the light attribute will not be defined, as it is not needed.

Lines 15-20 deal with scoring. OPTIONAL_SCORED means that we want to use the scoring attribute as a simple way to give the player points when they collect certain items or visit certain places. OPTIONAL_FULL_SCORE means that we want to support the verb "full score," giving the player a summary of their points. MAX_SCORE sets the maximum score. OBJECT_SCORE sets the number of points the player gets for each collected object that has the scoring attribute, and ROOM_SCORE sets the number of points for each visited room that has the scoring attribute. If you don't declare OPTIONAL_SCORED, these two constants have no effect.

Then we include "globals.h" and "puny.h" as we always do in a PunyInform game. If we need to declare some global variables of our own or write entry point routines, we do so between the inclusion of "globals.h" and "puny.h". For this game, we don't need to write anything there.

Grammar, verbs, and actions

```
Verb 'knock'
    * 'on' noun -> KnockOn;

Extend 'look'
    * 'under' noun -> LookUnder;

[ KnockOnSub;
    "Knock knock!";
];

[ LookUnderSub;
    "There's probably not much to see there.";
];
```

Game code part 2

PunyInform comes with a standard set of verbs that most games need, such as take, drop, open, lock, put, insert, etc. Each verb has a grammar, one or more lines explaining how it can be used by the player to form a command that leads to an action. Different lines can lead to different actions, such as "put coat on shelf" which triggers the ##PutOn action, but "put on coat" triggers the ##Wear action (actions are (optionally) preceded by ## to make them stand out in the text and code).

In this game, we would like the player to be able to knock on a door, and PunyInform does not have a verb or action for this. So we declare the verb "knock." Each grammar line starts with *, adds any prepositions the player must type, if any, and "noun" to say that the player must specify an object. You can also write alternative prepositions, like this: * 'on'/'at'/'in' noun -> KnockOn. And you can specify two nouns like this: * 'on' noun 'with' noun -> KnockOn. The last grammar line must end with ;. Writing KnockOn after -> in a grammar line means that there is now an action called ##KnockOn, and we will need to create a routine called KnockOnSub to handle this action, otherwise we will get a compilation error. Also note that dictionary words (words we want to compare with the words typed by the player) must always be written with single quotes in Inform.

We would also like the player to be able to look under objects. This is a little different, because PunyInform already supports the verb 'look'. We just want to add a line to the grammar of the verb 'look'. We do this with Extend. For this grammar line, we specify another new action called ##LookUnder. Of course, we will also need to create a routine called LookUnderSub to handle this action.

Now we come to action routines. An action routine defines what an action does by default. Each action falls into one of three groups:

Group 1: Meta-actions that do something to control the game itself, such as ##Save (which saves the current state of the game to a file) and ##Score (which prints the player's score).

Group 2: Actions that change something in the game world or print some important information about it, such as ##Take (which moves an object into the player's inventory) and ##Inv, which lists the player's possessions.

Group 3: Actions that by default do nothing except print a message that essentially refuses to do it or says it has no effect, such as ##Swim, ##Push, and ##Yell.

You can change what actions in group 2 and group 3 do for certain objects by adding before routines to those objects or to the room where the action takes place. You can also choose to let any other objects present react using a react_before routine. If any of these routines return true, it means that the default behavior of the action has been overridden and the action routine will not even be called.

For group 2 actions, you can also use an after routine in the affected object or room, or a react_after routine in any other object present to do something special and/or print a custom message after the action has occurred. If the routine returns true, the action routine will not print anything.

Most actions you add to a game tend to be group 3 actions. Both actions specified here are group 3 actions. This means that the action routine should only print a default message and nothing else.

The outdoor locations

```
Object Lawn "Lawn"
    with
        description "You are on the lawn. A path leads north to the house.",
        n_to FrontPorch,
        s_to "But you just got here!";

Object FrontPorch "Front Porch"
    with
        description "It's a lavish porch, certainly larger than your entire
            flat.",
        n_to FrontDoor,
        s_to Lawn;
```

Game code part 3

Let's start defining the game objects. First, there is the Lawn location. Nothing new there. Then there is the FrontPorch location. The only new concept here is that an exit can lead to a door object instead of a location. We will discuss this in the next section.

Using parse_name

```
53  Object -> DoorMat "door mat"
54      with
55          parse_name [ w1 w2;
56              w1 = NextWord();
57              w2 = NextWord();
58              if(w1 == 'door' && w2 == 'mat') return 2;
59              if(w1 == 'doormat' or 'mat') return 1;
60          ],
61          before [;
62              Take, Remove:
63                  "That would be incredibly rude.";
64              LookUnder, Pull:
65                  if(parent(MetalKey) == 0) {
66                      move MetalKey to location;
67                      score = score + 10;
68                      "You discover a key!";
69                  }
70          ],
71      has supporter enterable;
72
```

Game code part 4

Now let's move on to the doormat, and this is where things get interesting. It might seem like we could have just added the name "door" "doormat" to the properties, but that would get us into trouble. You see, there is a front door at this location (declared further down) and this doormat. If the player types "open door" or "examine door," we don't want the parser to ask, "Which one do you mean, the front door or the doormat?" So, let's give the DoorMat object a parse_name routine. This is a routine that can read words from the player's input using NextWord() and return how many words in a line match this object. This specific routine returns 2 (meaning it found two words) if the player typed "doormat" or, otherwise, returns 1 if the player typed "doormat" or just "doormat." If the player has typed anything else, the routine returns false (which is equal to 0), meaning it found nothing. Routines in properties, such as this one, return false/0 by default, while named routines (such as the action routines in the last section) return true/1 by default.

In line 55, the routine begins with [. After that, we can name up to 15 local variables that we want to use in this routine. These are also used as parameters, i.e., they receive assigned values if someone calls the routine with one or more arguments. We choose to create two local variables named w1 and w2. The ; marks the end of the list of local variables. A local variable is a variable that is created when the routine is called and destroyed when the routine returns. It only exists within the routine and cannot be accessed in any way from any other routine. Different routines can have local variables with the same name without risk of confusion.

In lines 56 and 57, we assign values to these two variables. The assignment is performed with a single =. On the right side, we call the NextWord function and use its return value. The parentheses mean "call the routine with the name I just gave." If you want to send arguments to the routine, you put them in parentheses with commas between them.

In line 58, we see ==, which is used to check if one value is equal to another value. We also see the operator &&, which means AND, so the if statement checks if the first word is "door" AND the second word is "mat." In addition, Inform has the operators ||, which means OR, and ~~, which means NOT.

In line 59, we see another trick in Inform: the keyword or. It can be used when comparing a value with multiple other values and generates shorter and faster code than performing comparisons one by one. Line 59 is functionally equivalent to:

```
if(w1 == 'doormat' || w1 == 'mat') return 1;
```

Hide the key under the doormat

There is a key hidden under the doormat. Since the key cannot be seen, we don't actually put it in the location yet, but the player should be able to find it. So, we want the DoorMat object to be special in two ways:

If the player looks under the doormat or moves it aside, they should find the key.

To avoid exposing the key with one of the standard actions that many players try, such as "take everything," we want to prevent the player from picking up the doormat.

We achieve this with the before routine. First, we react to

the ##Take and ##Remove actions. Actually, only ##Take would have been enough here, but ##Take and ##Remove are the only two actions you ever need to consider if you want to prevent an object from coming into the player's possession, so I usually write both (##Remove is used when an object is removed from inside a container or on a stand).

Printing a response and returning true

The syntax on lines 63 and 68, where we simply type a string enclosed in double quotes, is a shortcut for the print_ret command, which means "print this, then print a newline character and return true." So these four pieces of code are actually equivalent:

```
print "No!";
new_line;
rtrue;
```

```
print "No!^";
rtrue;
```

```
print_ret "No!";
```

```
"No!";
```

Four ways to do the same thing

Looking under the mat

So let's take a look at the second part of the before routine. If the action is ##LookUnder or ##Pull, we check if MetalKey has parent 0, which means it hasn't been placed in the game yet. If true, we move the key to the current location (location is a global variable that contains the object ID of the current location), increase the score by 10 points, print a message, and return true. If the condition is not met, the routine will return false and the action will respond with its default message. Note that whenever we want an if statement to execute more than one command, we must enclose these commands in { } as we do on lines 65-69.

Finally, this object has two attributes: supporter and enterable. The first means that there can be objects on this object. The second means that the player can also be on this object, using a command such as "sit on mat" or "enter mat" to get on it.

The key

```
73  Object MetalKey "key"
74      with
75          name 'metal' 'key',
76          description "It's a shiny metal key.",
77      has scored;
78
```

Game code part 5

Now we define the key. Since there is no -> after Object, the key will not be placed in the last defined position, but will rather be suspended in limbo, not inside or on anything, when the game starts. The only new thing here is the scoring attribute. This means that the player will get a certain number of points when they pick it up for the first time. The number of points is the value of OBJECT_SCORE, given in line 19.

The entrance door

```
79  Object FrontDoor "front door"
80      with
81          name 'front' 'door',
82          description "Looks like a very sturdy oak door. Fine craftmanship.",
83          when_open [;
84              if(location == FrontPorch)
85                  "The front door is open, beckoning you inside.";
86              else
87                  "The front door is open, showing you a glimpse of the porch
88                      and lawn outside.";
89          ],
90          when_closed "The sturdy front door is closed.",
91          before [;
92              KnockOn:
93                  "You knock and wait in silence. There is no reply.";
94          ],
95          found_in FrontPorch Entrance,
96          door_dir (n_to) (s_to),
97          with_key MetalKey,
98      has static door openable lockable locked;
99
```

Game code part 6

Now it's time to define our first door. We will use PunyInform's Simple Door mechanism, as it simplifies the code and most of the time meets our needs. A door in PunyInform is any object that has the door attribute. A door is a passage between locations. It typically exists in two locations and provides a means to go from the first location to the second or from the second location to the first.

when_open and when_closed play the same role as initial but only for doors and containers. That is, the object gets its own paragraph in the room description. If the object is open, when_open is used, otherwise when_closed is used. In this case, we want the when_open text to be different depending on whether we are seeing the door from outside or inside the house, so we use a routine. For when_closed, a static string is sufficient.

Knock knock, who's there? Now comes the reason why we created the verb "knock" and the action ##KnockOn earlier: we want to give an appropriate response if the player tries to knock on the front door, as it seems like a

sensible thing to do. We do this with a before routine. If the action is ##KnockOn, we print a message and return true, thus preventing the default message from being printed.

Movable objects

At any given time, each object can only be in one location, but PunyInform has a mechanism for moving objects whenever the player moves, so an object can appear to exist in multiple locations. This is sometimes referred to as movable objects and is controlled with the found_in property. The value of found_in should be a routine that returns true if the object should be in the current location, or a list of locations where it should be (note that a list of values in Inform is also referred to as an array in the documentation).

If we want to use the Simple Door mechanism for a door, that door must have a found_in property, and its value must be an array with exactly two values. In this case, we place the door in FrontPorch and Entrance.

Making the door work

A door must have a door_dir property to indicate which direction a door is facing. For a normal door, this should be a single value or a routine, and you usually have to write a routine that returns different values depending on the current position. If you use the Simple Door mechanism and the door has an array with two values in found_in, you can simply provide an array with two values here. If the player is in the first position in the found_in array, the first value is used, otherwise the second value is used. We use parentheses around the values to avoid a warning from the compiler.

For a normal door, we also need to provide a door_to property, which indicates where the door leads to. This should also be a single value or a routine. For a Simple Door, we can just skip it. If the player is in the first position in the found_in array, the door leads to the second position in the array, and vice versa.

For an object that is lockable, the with_key property is used to say which key fits the lock.

Then come the attributes. We use static to say that this object is fixed in one place and cannot be picked up by the player. We use door to say that it is a door. We need to add openable to say that it can be opened and closed and lockable to say that it can be locked and unlocked. Finally, we add locked to say that the door is locked from the start. And that's it, we have a complete door!

Adding an accessible container

```
100  Object Entrance "Entrance"
101      with
102          description "This area feels warm and welcoming, with its wooden panels
103              and many potted plants. A small sign next to the doorway to the
104              east tells you there's an office in that direction. Below the sign
105              are two boxes on the wall, one blue and one green.",
106          s_to FrontDoor,
107          e_to OfficeDoor,
108      has scored;
109
110  Object -> GlassBox "huge glass box"
111      with
112          name 'huge' 'glass' 'box' 'boxes//p',
113          inside_description "The box seems surprisingly roomy now that you're in
114              it. Of course the glass sides help - you'd feel a lot more
115              constrained if the sides weren't transparent.",
116          before [;
117              Take, Remove, Turn, Push, Pull:
118                  "It weighs a ton at least!";
119          ],
120      has static container openable enterable transparent;
121
```

Game code part 7

Here is the definition of Entrance, the room inside FrontDoor. You can also see that one exit leads to FrontDoor and another leads to another door called OfficeDoor. This location has the scoring attribute, which means that the player gets a certain number of points when they arrive here for the first time. The number of points is the value of ROOM_SCORE, given in line 20.

There are several interesting objects in this location. Let's start with the glass box. As you will soon see, there are several boxes in this location. It makes sense to allow the player to refer to all of them at once, as in "take the boxes." To support this, we add the word "boxes" to the name property, except we add the plural flag to indicate that it is the plural form of the name, like this: "boxes//p".

In addition, we are making this box accessible and want to add a description for when the player is inside the box. We do this with inside_description. This can also be used for accessible supporters.

If the player tries to move the box, we want to tell them that it is very heavy, so we add a before routine that captures the common actions used to move things.

Let's proceed with the attributes. This is a static object, which means that any attempt to pick up or move the

object gets a response like "It is fixed in place." (only we chose to ignore all or most of them in the before routine). It is a container, which means that other objects can be inside it. openable means that it can be opened and closed. Enterable means that the player can enter the object. Transparent means that it is transparent, which means that we can see the contents even when it is closed. Transparent also means that if something emits light outside, there is also light inside and vice versa, but this does not matter in this particular game, as we have defined OPTIONAL_NO_DARKNESS in line 12.

The necklace

```
122 Object -> -> Necklace "necklace"
123     with
124         name 'necklace' 'string',
125         description "It consists of a piece of string with an emerald, a ruby
126             and a sapphire on it.",
127         before [ loc;
128             Attack:
129                 if (ObjectIsUntouchable(noun))
130                     rtrue;
131                 loc = parent(self);
132                 remove self;
133                 move Emerald to loc;
134                 move Ruby to loc;
135                 move Sapphire to loc;
136                 score = score + 20;
137                 "You break the necklace, leaving you with three gems.";
138         ],
139     has transparent;
140
```

Game code part 8

Now let's add a necklace. It has two arrows (->) after Object, which means that it is located in the last object defined with a single arrow, in this case the glass box.

The player should be able to break the necklace to free the gems. We do this with a before routine. We will need a local variable and we will call it loc (short for "location of the necklace"). We capture the ##Attack action, which is what we get if the player types "break necklace," "destroy necklace," or "wreck necklace." First, we use the PunyInform ObjectIsUntouchable routine to check if the necklace is untouchable by the player. If so, ObjectIsUntouchable prints a message explaining to the player why they cannot reach the necklace, and we return true to tell PunyInform not to do or print anything else. Note that here we use noun, which is always set to the primary object of the current action. When we are in a before routine (at least one that does not belong to a room), noun has the same value as self, so we can use them interchangeably. Since we have decided that the player can reach the necklace, we execute a sequence of commands:

We set loc to the current parent of the necklace. This could be the glass box where we found it, but it could also be the player, or it could be the location of the entrance, etc. Then we remove the necklace, i.e., we remove it from the object tree and put it in the void where the player will never see it.

Next, we move the three gems that are part of the necklace to the position where the necklace was.

We increase the score.

We print a message and return true, indicating that PunyInform should not print the default message for ##Attack.

The necklace has a single attribute: transparent. This is to indicate that if there are objects inside this object, they can be seen by the player. Placing objects inside other objects means different things depending on the type of object of the parent object:

For a container (a bag, a box, a car, etc.), it means that the objects inside it are actually inside the container.

For a support (a bed, a table, a carpet, etc.), it means that the objects inside it are on the support.

For an animated object (a person, animal, robot, etc.), it means that the objects inside it are held by the animated object.

For all other objects, it means that the objects are part of the object, such as buttons are part of a cash register, etc. The last case is the one we are dealing with here. Look at the next piece of code to see the gems that are part of the necklace.

The gems

```
141 Object -> -> -> Emerald "emerald"
142     with
143         name 'emerald' 'gem' 'gems//p',
144         article "an";
145
146 Object -> -> -> Ruby "ruby"
147     with
148         name 'ruby' 'gem' 'gems//p';
149
150 Object -> -> -> Sapphire "sapphire"
151     with
152         name 'sapphire' 'gem' 'gems//p';
153
```

Game code part 9

Now let's add the three gems. The three arrows indicate that they are in (i.e., part of) the necklace we just defined. All three have the plural word "gems" added to their name property, so the player can "take gems." The emerald needs the indefinite article "a," so we give it the article property and the desired article as a string.

Putting the gems in the necklace in this way means that wherever the necklace goes, the player can examine the gems, but if they try to take one, they receive a message that it is part of the necklace.

The office door lock

```
Object -> BlueBox "blue box"
    with
        name 'blue' 'box' 'boxes//p',
        description "It's a small metal box, firmly attached to the wall next to
            the office door.",
        describe [; rtrue; ],
    has static container open;

Object -> GreenBox "green box"
    with
        name 'green' 'box' 'boxes//p',
        description "It's a small metal box, firmly attached to the wall next to
            the office door.",
        describe [; rtrue; ],
    has static container open;
```

Game code part 10

In addition to the glass box with the necklace, there is also a blue box and a green box attached to the wall next to the office door. These are actually part of the office door lock mechanism. If you put a sapphire in the blue box and an emerald in the green box, the door unlocks. The code to make this happen, however, is not in the boxes but in the door itself.

The code for defining the boxes holds few surprises. The only new thing here is the describe property. This is used when you might want to print a paragraph of text for an object in the room description. If you do this, you print a new line, whatever text you want, another new line, and then return true. If you decide you don't want to print a separate paragraph for this object, return false. But there is a trick you can do here: print nothing but return true. This will prevent "look" from printing anything about the object. The boxes are mentioned in the room's description property, so we don't want the game to show them again.

The office door

```
Object OfficeDoor "office door"
    with
        name 'office' 'door',
        react_after [;
            Insert, Take, Remove:
                if(Sapphire in BlueBox && children(BlueBox) == 1 &&
                        Emerald in GreenBox && children(GreenBox) == 1 &&
                        self has locked) {
                    give self ~locked;
                    score = score + 70;
                    "A loud click is heard from the office door.";
                }
        ],
        describe [; rtrue; ],
        found_in Entrance Office,
        door_dir (e_to) (w_to),
    has static door openable lockable locked;
```

The player can place various objects in the blue and green boxes. Whenever the player has placed something in something else ("put emerald in green box" triggers an ##Insert action) or picked something up, we want to check if the conditions for unlocking the door are met. When we want an object (in this case, the office door) to react to an action that takes place nearby but in which this object may not be directly involved, we can use react_before and react_after. They work very similarly to before and after. We can use noun and second to check which primary and secondary names are used in the action. In this case, however, we don't need them. Let's check if the following conditions are met:

The sapphire is in the blue box.
There is exactly one object in the blue box.
The emerald is in the green box.
There is exactly one object in the green box.
The door is locked.

If all these conditions are met, we:

Unlock the door

Give the player some points
Print a message and return true, which means that PunyInform should not print anything
The door is unlocked by setting the locked attribute to 0 for the OfficeDoor object. An attribute is a bit value, it can be 0 or 1. Here's what you can do with an attribute:

You can check if the locked attribute is 1 for object X with if(X has locked)
You can check if locked is 0 with if(X does not have locked)
You can set locked to 1 with give X locked
You can set locked to 0 with give X ~locked

The rest of this object's definition is very similar to the entrance door. One difference is that, although it has the lockable attribute, it does not have the with_key property. This means that the player can tell that the door is locked, but there is no key that can unlock it using the "unlock" verb.

The final location

```
Object Office "Office"
    with
        name 'desk' 'chair',
        description "It's a neat and tidy little office. There's a desk and a
            chair near the window.",
        w_to OfficeDoor,
    has scored;

Object -> Will "Will"
    with
        name 'will' 'piece' 'of' 'paper',
        initial "There's a piece of paper on the desk.",
        description [;
            deadflag = GS_WIN;
            score = score + 50;
            "It's uncle Billy's Will, and it says he leaves everything to you!
                You are now rich!";
        ];
```

Game code part 12

We have reached the final location, the office. We describe a desk and a chair, but they are not really that important, so we would prefer not to implement them as objects. Instead, we put these words in the name property of the location, so that the player receives a message saying that they do not need to refer to these objects if they try to interact with them.

The will has the initial property, which means that this string will be printed as a separate paragraph when the player views this room, as long as the object has not been moved. If the player picks up the will and drops it again, it will simply be listed along with all the other normal objects in the location.

And then there's the description, of course. If the player tries to examine or read the will, the description property is consulted. If it's a routine, it's executed. So this is a good point for us to end the game. Here's what we do:

We set deadflag to GS_WIN (which is 2, but GS_WIN is easier to remember) ("GS" stands for Game State, by the way, since this variable controls whether the game continues or ends). If you set it to GS_DEAD instead, the game also ends, but with a message saying that the player has lost.

We increase the score.

We print the description of the will. (Just by writing a string in double quotes like this, we also print a new line, which is good, and return true, which doesn't matter: the description property should always print a description, and its return value is simply ignored.)

After the action is complete, PunyInform notices that deadflag is now GS_WIN and ends the game.

The Initialise routine

```
[Initialise;
    print "^^Ever since your dear uncle Billy got sick, his spoilt kids have
        been fighting over the inheritance. Billy told you in confidence that
        he's been working on a will to fix that problem, but you don't know
        if he managed to finish it before he passed away last month. You've
        worked hard to find it, but this far you've failed. This morning you
        got news from Billy's lawyer that Billy owned another house that noone
        knew about. You realize you're clutching at straws here, but you have
        to check it out. The cab pulls away and you're on your own.^^";
];
```

Game code part 13

Finally, it's time to write the mandatory Initialise routine, which is called when the game starts. We can use this routine to move things around, start timers, etc., but for this game all we need is to print an introduction to the game.

The finished game

After compiling the game into a Z-code file, you can play it with your favorite Z-code interpreter. The screenshot in the next page is from Windows Frotz.

Links

The source code seen in this tutorial, as well as the compiled game (demogame_2.z3) can be downloaded from this folder:

https://drive.google.com/drive/folders/1SIqRNqi99LunTq9jg2-4sWzVGSVhhsg6?usp=sharing

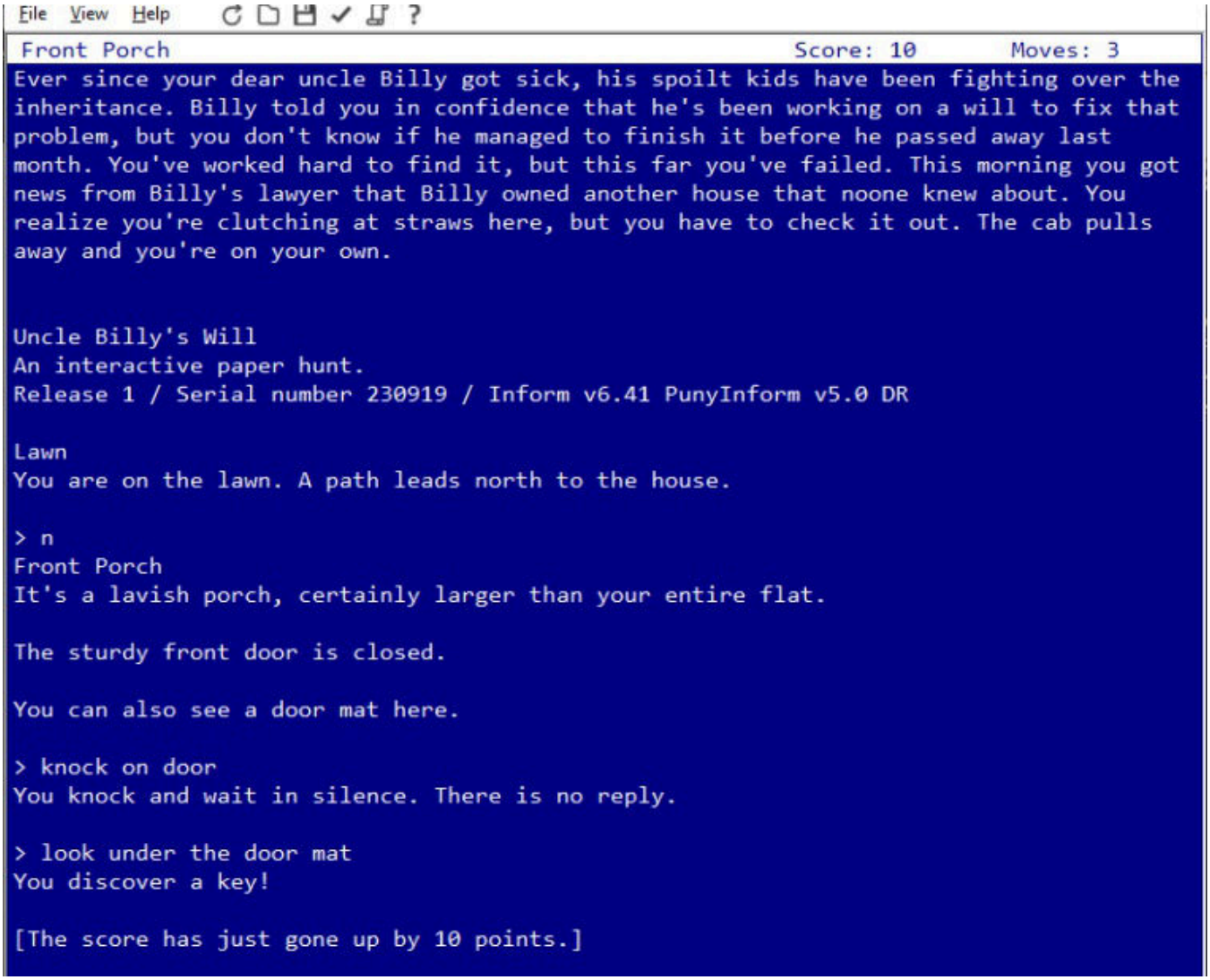


Fig. 15 - The game in action

To boldly go where no SID has gone before!

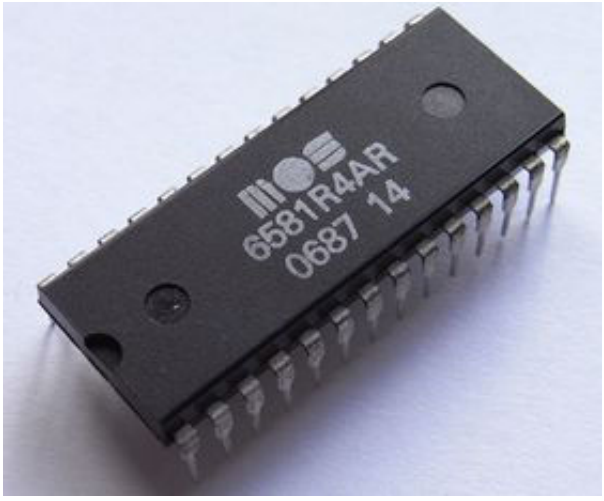
A STUDY OF AUDIO SAMPLING METHODS FOR HIGH-QUALITY MUSIC PLAYBACK ON THE C-64

by Marco Pistorio

Here we are with an article that some of you have been waiting for for quite some time, roughly since I started posting on Facebook and my YouTube channel (https://www.youtube.com/@mpsoftita-retroprogramming8233) clips of audio tracks sampled and then played back via the C64.
Some people called it a "miracle", others said it was edited, others said it was easy with an emulator (but I made videos showing the results obtained on a real machine), and finally, others said it was not possible to get that audio performance from the good old SID.

An important preamble: I am not the "hero" here, nor is the player, but rather the good old SID, the chip dedicated to the audio section that our beloved Commodore 64 was equipped with.
This integrated circuit was built with such foresight that even today we still find it difficult to fully understand how it can reproduce such a wide variety of sounds with such a high level of quality, despite the fact that it was designed way back in 1981.
First and foremost, praise goes to the person who created it, engineer Robert Yannes.
I had a chat about this with my friend FlavioWeb Pasqualin, whom I would like to take this opportunity to greet.
He agreed with me about the foresight of those who created this electronic gem over forty years ago.

At this point, there are a few things that need to be clarified.
First of all, sampling does NOT take place in real time on the C64, but rather involves excerpts of songs sampled from a PC.
More specifically, I used the software "Audacity" for both sampling and exporting these samples in .raw format.
And this is where a second big problem arises.
A lot of RAM is needed to store the song samples.
Using a sampling frequency of 14.4 KHz, each second of audio requires 14,400 samples, which in terms of RAM occupied is about 14 Kb(!).
In other words, with 3 or 4 seconds of song, I use up all the RAM available on the Commodore 64.



MOS Technology SID

My goal was to be able to play at least 1 minute per song, so I would need about 1 MB of RAM, which is 16 times the memory available on the Commodore 64.
So I tried to think of a viable solution.
Compression was not suitable for two reasons:
the first is that compressed data must be decompressed, and this requires valuable machine time.
the second is that, even using the most efficient algorithms, I still would not have been able to store 1 MB of samples in 64 KB of RAM (or in less space, of course).
Discarding compression... how could I find 1 MB of RAM for the C64?
It occurred to me that the EasyFlash cartridge has 1 MB of RAM that we can use as we please.
It's true, this hardware was released in 2009.
However, with the addition of this RAM, I can use the C64 and in particular the 6510 (CPU) and 6581/8580 (SID) chips exactly as they were manufactured, achieving my goal without too much difficulty.
Here is a second important point to consider.

We can better appreciate the potential of the SID today thanks to these new tools at our disposal!

Over the years, I had already become interested in audio reproduction techniques involving the C64 and, in particular, the SID.
Even in the early years of the Commodore 64's arrival on

the market, some of these features were already being exploited to a certain extent.
I remember, for example, the voice saying "GHOSTBUSTERS" in the intro to the famous game released by Activision in 1984 for our beloved machine, but that's not the only example.
The game "Impossible Mission," also from 1984, also exploited this speech synthesis feature.
Famous among fans is the phrase "Another visitor. Stay a while... stay forever!" which is recited in the game.
Not to mention S.A.M. (Software Automatic Mouth), released in 1982, a software that reproduced written English text in an amazing way (for those years).
These results can be achieved by using the DAC (Digital-Analog Converter), which is accessible by acting on the register that manages the volume of the SID.
However, the results that can normally be achieved are modest, as only 4 bits are available for this conversion.
This allows for a reasonable resolution to simulate voice content, but it is not possible to go much further unless other strategies are used in conjunction with the SID volume register.

A high-quality result, for example, was achieved by the great Antonio Savona (whom I would like to take this opportunity to greet), who released his own player in 2017.
This player uses the SID volume register, together with lookup tables, to improve the perception of the audio being played.
He also adopted the same medium I chose to store the data managed by his player, namely an EasyFlash cartridge.
Through pre-processing (which includes a vector quantization technique and refined statistical techniques such as K-means) of the signal to be processed, starting from the single sample to be reproduced, the value is calculated and taken from a lookup table, written to the SID volume register, and then reproduced.
Thanks to skilful management of machine cycles, his player achieves a sampling frequency of 48 kHz!
For those who want to learn more:
- https://csdb.dk/release/?id=162941
- https://csdb.dk/release/?id=162951
- http://brokenbytes.blogspot.com/

But let's get to the method I decided to use.
Personally, I preferred to avoid pre-processing because it seemed a bit like playing with a deck of 'marked' cards.
Pre-processing shifts the burden of data processing to the PC, facilitating the tasks of the C64, and the SID in particular.
This is a fair and legitimate strategy, considering the results that can be achieved, but my idea was to use a faster, simpler method that did not require excessive data processing.
I therefore preferred sampling that used 8 bits instead of just 4 bits, to ensure good sound quality.

Here I made use of studies published in a couple of articles in issues 20 and 21 of C=Hacking, dating back to 2001-2002.
More precisely, these articles studied the possibility of using pulse width modulation (PWM) to reproduce sampled audio with good quality.
Some code examples concerned the C64 with SuperCPU, but there was something that could run on a "breadbin" C64 without any additions.
Of course, with the limited amount of RAM available on the C64, it was not possible to play more than 3 or 4 seconds of a song.
However, this problem can be circumvented by using EasyFlash.
This choice involved modifying the code I had initially found and decided to use, as I had to integrate it with the cartridge's RAM bank management.
This management requires additional machine cycles (read also 'delay') and without corrections, the delay introduced generated a rather annoying background "tick".

For those who are curious, the code I originally started with is as follows:

```
//////////////////////////////////////////////////////////////////////////////
; 7.5 bit sample player for C64
; (C) 2001 Levente H rsfalvi

; Ideas obtained from:
;
; - Steve Judd  (C= Hacking 18)
; - Dag Lem (reSID emulator engine)

; Runs on a stock C64 (PAL or NTSC), and plays an unsigned 8-bit sample
; using PWM.

; Sampling rate: 68 clock cycles
; 17734470 / 18 / 68 = 14488.946078 Hz for PAL
; 14318180 / 14 / 68 = 15040.105042 Hz for NTSC

; Table Driven Assembler + Comlink
; (probably easy to compile using another 6502 cross assemblers too)

; This routine is 'free software', I just don't bother quoting GPL


#ifndef LO
#define LO(x) (x&$ff)
#define HI(x) (x>>8)
```

```
#endif

        *= $0900

        *= *-2
        .word *+2                   ;Load address for the binary

        sei
        lda #$0b
        sta $d011                   ;blank screen
        lda #$35
        sta $01                     ;ROMs off

        lda #$0f                    ;Volume + filter
        sta $d418
;        lda #$07                   ;Lowpass filter (but it sounded so
;        sta $d417                  ;bad on my 6581R4 that I commented
;        lda #$a0                   ;it out
;        sta $d416

        lda #$00                    ;Set SID freq registers
        sta $d400
        sta $d407
        sta $d40e
        lda #$40                    ;$4000 --> the wave accu is
        sta $d401                   ;incremented by $004 each CPU clock
        sta $d408                   ;cycles
        sta $d40f
        lda #$f0                    ;maximum sustain level for all chns
        sta $d406
        sta $d40d
        sta $d414

        lda #LO($0a00)              ;start address of sample in memory
        sta ptr
        lda #HI($0a00)
        sta ptr+1
        lda #$cd                    ;end address high. Zpg ($fb) is used
        sta $fb                     ;because of timing reasons

        ldx #$49                    ;pre-load values for the SID command
        ldy #$41                    ;registers...

ptr     =*+1
loop    lda $face    ;4             ;Load next sample...
        sta $d402    ;8             ;put to the sample width reg LO
        stx $d404    ;12            ;RESET SID voice
        sty $d404    ;16            ;...then start wave counter
        adc #$01     ;18            ;Increase sample value
        sta $d409    ;22            ;(presumably, there'll never be a
        stx $d40b    ;26            ;carry), then go on for next SID
        sty $d40b    ;30            ;channels.
        adc #$01     ;32
        sta $d410    ;36
        stx $d412    ;40
        sty $d412    ;44

        inc ptr      ;50            ;Increase sample pointer
        bne loop1    ;52  (when jumps, 53)
        inc ptr+1    ;58
        lda ptr+1    ;62
        cmp $fb      ;65
        bne loop     ;68

        lda #LO($0a00)              ;Reload it (timing is broken, but no
        sta ptr                     ;matter since it only happens per one
        lda #HI($0a00)              ;full sample replay)
        sta ptr+1
        jmp loop

loop1   bit $face    ;57            ;timing...
        bit $face    ;61
        bit $face    ;65
        jmp loop     ;68            ;Then loop.

i_sample =$0a00                     ;Just to include the sample binary
                                    ;(TASM + Comlink feature)
        .export i_sample

.end
```

//

The credits for the program are already indicated in the source code.
We can describe the proposed algorithm as follows:

- Init: volume at maximum, sustain level of voice 1 at maximum, attack begins. The frequency is set (=$4000), then the wave counter is incremented by 4 each processor clock cycle.

Loop:
- Load the value of the next sample and insert it into the low register of the pulse width ($d402; make sure $d403 is 0).
- Set the test bit and clear the test bit (counter reset). (These are the two precalculated values $49 and $41, which are initially stored in registers X and Y, respectively, editor's note).
- Increase the sample pointer, delay a little, then execute the loop. The delay must be 64 clock cycles + the time the test bit is kept set (4 cycles for each opcode, if using STX $d404: STY $d404 immediately with preloaded values).

What will happen? The 8-bit sample value is inserted directly into the pulse width register (the MSBs of the pulse width register are cleared!...). The wave counter is started (test bit released) and increases by 4 for each CPU cycle (=counts 256 in 64 cycles).
After a while, the counter will reach the value in the pulse width register. This happens exactly after (8-bit sample value /4) cycles, as above. In this cycle (or the next one?...) the SID changes its pulse output to 0. VoilÃ !

This is the explanation provided by the author of this source.

It is important to emphasize the fundamental role of the two values preloaded into registers X and Y and then stored in the control registers of each SID entry.

The meaning of these values $49 and $41 is explained in more detail in my comments.

```
// pre-load values for the SID command
// registers...
```

```
//  NOISE  PULSE SAW  TRI  TEST  RING SYNC GATE
//   0      1     0    0    0     0    0    1

// #$49


//  NOISE  PULSE SAW  TRI  TEST  RING SYNC GATE
//   0      1     0    0    0     0    0    1

// #$41

/* Playing a sound:

1. Set frequency and ADSR, and in case you use the pulse wave also
set the pulse width.
2. Set the waveform and the GATE bit in the control register.
Setting the gate bit will start the ADSR envelope generator.
3. Wait for as long as you wish the note to be played.
4. Clear gate bit. This will start the release phase of the
ADSR.
(https://www.oxyron.de/html/registers_sid.html)
*/
```

The role of the TEST signal is not highlighted in these comments. The TEST signal, as already mentioned, allows you to play the sample and hear it for a specific period of time. After this period of time, the SID transforms its pulse output to 0.

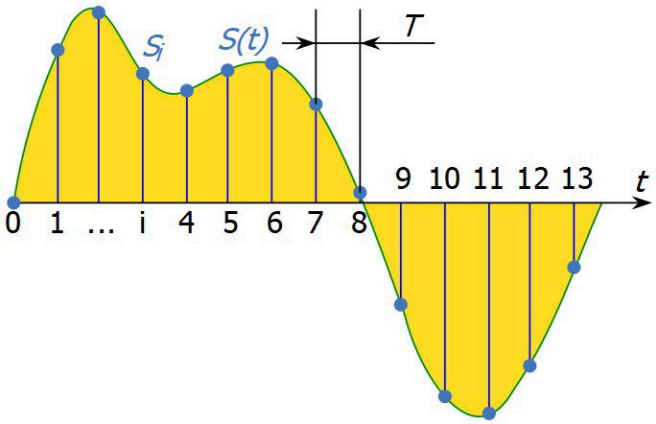
For a more detailed explanation, I suggest carefully reading the following articles:

o "Pulse Width Modulation, continued" by various.
Tying up some loose ends from last issue's digi article. ---> c=hacking issue 21

o "The C64 Digi," by Robin Harbron <macbeth@psw.ca>, Levente
Harsfalvi <levente@terrasoft.hu>, and Stephen Judd <sjudd@ffd2.com> ---> c=hacking issue 20 + ADDENDUM with example

These articles can be found at this address: http://www.ffd2.com/fridge/chacking
and contain, among other things, a mathematical explanation of how audio sampling works.

It may be superfluous to add, but I will do so anyway, to emphasize the quality of this work, the amount of study and testing that must have gone into its creation, all the work of Levente Harsfalvi and others, all of whom are already mentioned in the credits contained in the code.



Representation of signal sampling. The continuous signal S(t) is represented by a green line, while the discrete samples are indicated by blue vertical lines.

Source: https://it.wikipedia.org/wiki/Campionamento_(teoria_dei_segnali)

But what do we mean by audio sampling?

Sampling is a technique that consists of converting a continuous signal in time or space into a discrete signal, evaluating its amplitude at usually regular time or space intervals. In this way, following a subsequent quantization and conversion operation, it is possible to obtain a digital string (discrete in time or space and amplitude) that approximates the original continuous one.

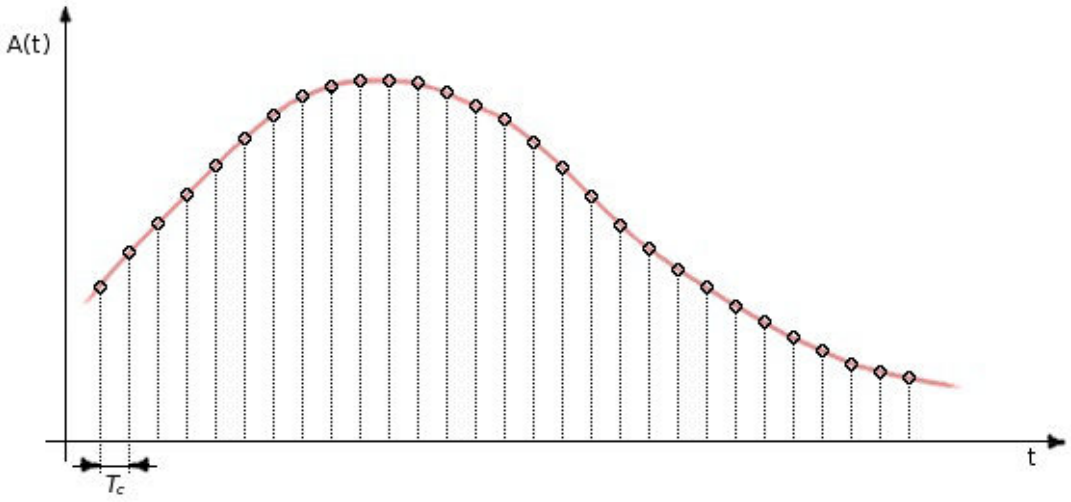
If we denote the sampling frequency as fc, i.e., the frequency at which we extract the samples, and denote the maximum frequency of the spectrum of the signal to be sampled as fm, Shannon-Nyquist sampling theorem tells us that if:

fc > 2*fm

then our sampling frequency will allow us to reconstruct the analog signal we intend to reproduce without any loss of information.

If the sampling frequency we have adopted does not meet this condition, an effect known as "aliasing" will occur, causing distortion of the reconstructed analog signal.

To summarize the concept in simpler terms: if we take samples at the right frequency, we will obtain an audio



Waveform sampling
Source: https://otolabdidattica.wordpress.com/2011/02/23/glossario-campionamento/

signal of satisfactory quality; otherwise, the quality may not be the best.

I won't go into further detail so as not to bore the reader too much, but...remember this 'aliasing'.

In the articles I have mentioned, you will find most of what you need if you want to delve deeper into the subject. Furthermore, if the sources already mentioned are not sufficient... Google is your friend, always keep that in mind! :D

As I mentioned earlier, I have also added EasyFlash management, based on its SDK.
I found this information at this address:
https://codebase64.org/doku.php?id=base:code_sample

However, EasyFlash allows you to have up to 1 MB of RAM available, which allows you to play about 1 minute of music. To overcome this limitation, I adapted the player code to use two new-generation cartridges, gmod3 and gmod4, which allow, at least on paper, up to 16 MB of RAM to be available!

Unfortunately, it seems that gmod3, which is emulated by VICE at the time of writing, will not be physically developed.
Instead, based on an email conversation with Jens Schönfeld of Individual Computers, who is developing this new hardware for the C64, it seems that gmod4 will be developed further, until it is physically realized and then marketed.

Let's now consider sampling at a frequency of 14400 Hz. According to what has been mentioned (Shannon-Nyquist sampling theorem), it would be possible to reconstruct signals with frequencies up to 2*fm.
fm, i.e., the maximum frequency of the spectrum of the signal to be sampled, would be 7200 Hz (14400/2).
What happens if there are higher frequency signals in the audio track to be sampled?
Higher frequencies would not have enough samples to describe them, but those same samples would describe a lower frequency.
This frequency is called the aliasing frequency and, being low, would fall within the audible band. Therefore, we would have added a frequency to the initial signal that did not exist before the sampling operation, while losing the high frequency that was present in the original signal.

In addition to the problem of aliasing, there are other factors that contribute to altering the sound quality of the sampled track. One of these factors is that we do not have infinite levels to describe each sample, but only 256 (i.e., we only use 8 bits to perform our sampling).
Finally, there is the problem of the "carrier" signal, which is always present when modulating a signal.
You can shift this signal to inaudible frequencies or apply a low-pass filter.

The author of the source I presented to you also faced the problem of eliminating the "whistles" that unfortunately some people hear because they are within the audio frequency band that can be heard by the human ear (approximately 20-20000 Hz).

The author's article suggests decreasing the sustain level on the SID to reduce these annoying whistles, but this is not always advantageous and, moreover, decreases the average volume of the audio signal produced.
The use of a low-pass filter is also mentioned, and we find some traces of this (with comments) in the code, but although it works (with some modifications), it drastically reduces the quality of the reproduced audio because it exaggerates the low frequencies, halves the mid frequencies, and significantly lowers the higher frequencies. In fact, a good low-pass filter does exactly that :)
In his article, the author hoped that, using a process similar to the one he described, it would be possible to reproduce an audio track with a higher frequency (perhaps with a lower resolution) in order to shift these whistles to higher frequencies and thus make them inaudible.
I believe this is the best solution, but of course it was necessary to review the code once again! :D
By studying the code of the player I reproduced in this article, you will discover that I use two voices instead of three. This allows me to save instructions (and therefore machine time) and therefore I can go up to higher sampling frequencies.

Some of you may already be thinking... what if only one voice were used? Well, I tried it and I can tell you that... it works!
I can sample at even higher frequencies, saving code and therefore machine cycles.
The downside is that I always have 1 MB of RAM available with EasyFlash, and as the sampling frequency increases, so does the number of samples needed to play each second of the song, and I can no longer play 1 minute of the song.

I would like to take this opportunity to thank my long-time friend Davide Scuderi, who assisted me in various tests on real hardware relating to the various audio "clips" that I have published, both on my personal Facebook profile and on my YouTube channel.

If I hadn't been reasonably certain of the quality of the playback of these clips on real hardware, my work would probably have remained in a drawer, because it would have been a mere exercise in style or little more.

Before saying goodbye, I thought I'd also show you the source code of the player as I revised and modified it in its version that runs on gmod3. This version did not require excessive modification of the code to make it work using the features of this cartridge.
In this specific case, i.e., using gmod3, I was able to keep the player code separate from the audio sample and start playback by linking the virtual image of the cartridge, which therefore contains only the audio samples, to the player.

This approach was not possible with EasyFlash and gmod4. In both cases, the player code resides in the cartridge together with the audio samples, making the code much more complex.

By the way, before I forget, I would like to thank Jens Schönfeld of Individual Computers and his staff for their invaluable help in migrating the player code from EasyFlash to gmod4. I hope that this new cartridge will soon be available on the market and enjoy the success it deserves.

Below is the player code, modified and revised as appropriate to play the audio samples on a gmod3 cartridge, written in accordance with Kick Assembler syntax.

```
//////////////////////////////////////////////////////////////////////
//////////

.label GMOD3_BANK0 = $de00
.label GMOD3_CTRL  = $de08

.const f_sample=$A0
.label f_sample_loc=$02
.label backup=$03
.label backup_x=$95

.pc=$0801

:BasicUpstart(main)

.pc=$080d

main:
alfa:
        jsr grafica

        lda #0
        sta $de00
        sta $de01
        sta $de02
        sta $de03
        sta $de04
        sta $de05
        sta $de06
        sta $de07
        sta $de08

        sta banklo+1
        sta bankhi+1

        lda max_contat_0
        sta contat_0

leggi_tasto:
```

```
          lda $DC01              //READ SPACEBAR IF
          cmp #$EF               //NOT PRESSED
          bne leggi_tasto     //THEN JUMP TO MAIN
                                 //PROMPT
          sei

          lda #$37        // Bank out kernal and basic
          sta $01         // $e000-$ffff

          lda #$7f        // Disable CIA IRQ's
        sta $dc0d
        sta $dd0d
        asl $d019   // Ack any previous raster interrupt
        bit $dc0d   // reading the interrupt control registers
        bit $dd0d   // clears them

ss:
          ////////////////////////////////////////////////////
////////  //legge configurazione registri SID e la applica, registro
per registro

          ldx #$1C
ssid:
          lda sid_conf,x
          sta $d400,x
        dex
          bpl ssid

          lda $d011
          and #%11101111
        sta $d011                       //blank screen

        lda #$00                        //start address of sample in
memory
        sta loop+1
        lda #$80
        sta loop+2

        lda #f_sample                           //end address
high. Zpg ($fb) is used
          sta f_sample_loc               //because of timing
reasons

        ldx #$49                        //pre-load values for the
SID command
        ldy #$41                        //registers...
          stx backup_x

omega:
          .fill $100-(omega-alfa)-$0d,$ea

loop_x:
          ldx backup_x
          lda #$80                      //start address of sample in
memory -HI
        sta loop+2

loop:   lda $face    //4                //Load next sample...
        sta backup
        sta $d402    //8                //put to the sample width
reg LO
        stx $d404    //12               //RESET SID voice
        sty $d404    //16               //...then start wave
counter

        //adc #$02     //18             //Increase sample value
          nop

        sta $d409    //22               //(presumably, there'll
never be a
        stx $d40b    //26               //carry), then go on for
next SID
        sty $d40b    //30               //channels.


        inc loop+1   //50               //Increase sample pointer
        bne loop1    //52               (when jumps, 53)

        inc loop+2                      //58      6
          lda loop+2                            //62      4
        cmp f_sample_loc                //65      //cmp in pagina
zero ritarda 3 cicli.. 3
        bne loop                                //68    2 oppure 3
quando salta


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

        inc banklo+1
        bne no_c

          inc bankhi+1
no_c:
          ////////////////////////////////////////////////////
////////

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

          lda backup   //4                //Load next sample...

        sta $d409    //22                //(presumably, there'll
never be a
        stx $d40b    //26                //carry), then go on for
next SID
        sty $d40b    //30                //channels.

        //adc #$02     //18               //Increase sample value
          nop

          sta $d402    //8                //put to the sample width
reg LO
        stx $d404    //12                //RESET SID voice
        sty $d404    //16                //...then start wave
counter

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

          ////////////////////////////////////////////////////
////////
bankhi: lda #$00                        // upper 3 bits of the bank
number
        and #7
        tax

banklo: lda #$00                        // lower 8 bits of the bank
number
        sta GMOD3_BANK0,x
          ////////////////////////////////////////////////////
////////


          lda backup                    //sto leggendo un sample 0?
          bne cnt                       //se no, si continua a
ciclare....
          inc contat_0                  //se si incrementa contat_0
          bne cnt                       //se contat_0 non si è
ancora azzero continua a ciclare
                                        //altrimenti si
esce!
exit:
          lda #%00111011                //bitmap mode on,Screen on,
normal screen contents are visible,screen height=25 rows
        sta $d011

        jmp main

cnt:
          jmp loop_x                    //3
```

```
loop1:
        nop
        nop
          nop

          inc $d020 //6

          jmp loop

//-------------------------------------------------------------------
---------

grafica:
{
        ldx #0
lp2:
        lda screen+256*0,x
          sta $0400,x
        lda colors+256*0,x
          sta $d800,x

          lda screen+256*1,x
          sta $0500,x
          lda colors+256*1,x
          sta $d900,x

          lda screen+256*2,x
          sta $0600,x
          lda colors+256*2,x
          sta $da00,x

          lda screen+256*2+$e8,x
          sta $06E8,x
          lda colors+256*2+$e8,x
          sta $dae8,x

          dex
          bne lp2

          lda #<hires
          sta $72
          lda #>hires
          sta $73
          lda #$00
          sta $74
          lda #$20
          sta $75

          ldy #0
lp3:
          lda ($72),y
          sta ($74),y
        dey
          bne lp3

          inc $73
        inc $75

          lda $75
          cmp #$40
        bne lp3

          lda #0
          sta $d020

          lda #0
          sta $d021

          lda $d018
          ora #$08
          sta $d018

          lda $d011
          ora #$20
          sta $d011

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

sid_conf:
.byte $0  //$D400
.byte $40 //$D401
.byte 0             //$D402
.byte 0             //$D403
.byte 0             //$D404
.byte 0             //$D405
.byte $f0 //$D406
.byte 0             //$D407
.byte $40           //$D408
.byte 0             //$D409
.byte 0             //$D40a
.byte 0             //$D40b
.byte 0             //$D40c
.byte $f0 //$d40d
.byte 0   //$d40e
.byte 0   //$d40f
.byte 0             //$D410
.byte 0             //$D411
.byte 0             //$D412
.byte 0             //$D413
.byte 0             //$D414
.byte 0             //$D415
.byte 0             //$D416
.byte 0             //$D417
.byte $0f //$D418
.byte 0             //$D419
.byte 0             //$D41a
.byte 0             //$D41b
.byte 0             //$D41c

contat_0: .byte $FD
max_contat_0:       .byte $FD

.pc=$1000
screen:
*=* "screen"
          .import binary "res/titanic/screen.dat",2

colors:
*=* "colors"
        .import binary "res/titanic/colors.dat",2

.pc=$4000

hires:
*=* "hires"
          .import binary "res/titanic/hires.dat",2

////////////////////////////////////////////////////////////////////
//////////
```

In this code, you will notice that I have eliminated the use of one entry in order to save machine cycles and thus increase the sampling frequency.

I have provided for the "repeat" of the last sample when changing the RAM page read by the cartridge to move on to the next one.

If this were not done, you would hear an annoying "click" about every second in the background.

In the original code, the data read was incremented before moving on to the next SID entry.

However, this led to distortion when moving from the

initial numerical value to the final value, going from a value of $ff to a value of $00. So, I commented out the addition, while retaining the necessary machine cycles.

You will notice several optimizations to improve overall execution times.

With this instruction, for example:
.fill $100-(omega-alpha)-$0d,$ea
I run the player code within the same 256 Kb RAM page.

I think that's really all for now.
I hope I haven't bored you too much.

If any of our readers would like further clarification and/or information, please send an email to the RetroMagazine editorial staff (retromagazine.redazione@gmail.com) and I will try, as far as possible of course, to accommodate you.

Best regards to all of you, friends and readers of RetroMagazine World.
See you soon!



Music sampled on Commodore 64 - "Total Eclipse of the Heart", B.Tyler (1983) https://www.youtube.com/watch?v=zx6Fv8hIb0o



Music sampled on Commodore 64 - "Rasputin", M. Boney (1978) https://www.youtube.com/watch?v=-4NMAdRKIq4



Music sampled on Commodore 64 - "Wonderwall", Oasis (1995) https://www.youtube.com/watch?v=ZJPRqEOUDcc

The weight of Tetris

Interview with Alexey Pajitnov, creator of a legendary game

by Ksenia Kamash (XENIA KAMA) – Translated by Carlo Nithaiah Del Mar Pirazzini

Russian streamer and YouTuber **Xenia Kama** (who will be assisting us in the editorial office) managed to interview the famous developer.

XENIA KAMA: Alexey, good morning. You are probably one of the most famous game developers in the post-Soviet space in general. "Tetris" has been called the most popular game in the world. Do you feel the weight of your popularity because you created such a famous game?

Alexey Pajitnov: Yes, of course, I do. But I've gotten used to it because the game has been around for 40 years. It quickly became very popular, so you could say that I was successful many years ago. And then it became a kind of routine. I've given many interviews and presented various awards for Tetris competitions. I've received quite a bit of mail from avid Tetris fans asking for autographs. They send me all kinds of cartridges, photos, and cards to sign. That's my burden. But I like it.

XK: Tetris has its own history. Would you like to tell us about it?

AP: Tetris originated from a puzzle called Pentomino. It looks like a square box measuring 5Ã—12 or 6Ã—10, containing all possible pentominoes. A pentomino is a shape made up of five squares. They can be neatly fitted together in various ways, forming letters such as Z, X, and so on. When I started creating similar games on the computer, I had an idea: to create a two-player game based on this set.

XK: What were you doing at the Computing Center when you created "Tetris"?

AP: I was working in the technical cybernetics laboratory at the Computing Center of the USSR Academy of Sciences. I was mainly focused on microprocessors and microprocessor systems. More specifically, in the field of automatic speech recognition. We were developing a speech feature extraction



Alexey really likes his game... he takes it with him wherever he goes



Alexey in a stock photo

device that was connected to the BESM-6 computer: we could input speech fragments into the BESM-6 memory and record them on tapes, then recognize them using various methods. Now, it probably sounds funny, but at that time it seemed like science fiction.

XK: Tell us how Tetris came onto the world stage.

AP: The first publisher of the game was a British man of Hungarian origin named Robert Stein. He saw it at the SZKI institute, as he later recounted. It was a kind of computing center that dealt with computer issues in Budapest. Stein would go there and see what software they had, buy some, publish some. He saw "Tetris" there and liked it. We signed a formal contract with Stein. He probably interpreted it as a full license agreement and started publishing "Tetris."

At that time, neither individuals nor legal entities could have contacts with foreign organizations. Only specifically organized agencies dealt with such contracts. The Ministry of Foreign Trade had several agencies that bought and sold various items abroad. I visited them all during the few months I was working on this. I was rejected by Licensintorg, never made it to RFTA, and elsewhere I was asked not to bother them. No one could talk to me because they were all too busy. But I met a rather nice guy at ELORG who took an enthusiastic interest in my "Tetris." I felt relieved because the matter was in the hands of someone at ELORG, and I started writing new games and focusing on other things.

After a while, it turned out that "Tetris" had already been published, and I was reprimanded for sending an incorrect telex. I told them that I had responded as best I could and that they could check it for themselves. They explained that what I had written was correct, but that it was not right for an inexperienced person to communicate, as my response was interpreted as a green light for publication, even though they had not yet received it.

In the end, all this commercial mess was already set up, and at the end of 1988, this version of Tetris was released on all PCs in America and Europe.

In 1988, Tetris took first place in all video game competitions. The game was distributed on dozens of home computers:

Sinclair, Commodore 64, Amiga, Atari, etc. Mirrorsoft published it everywhere. Spectrum HoloByte made an excellent PC version. And since the game, as you know, is highly addictive, it quickly became popular.

But we didn't grant the rights for the console. The initial contract only covered computers, devices with a monitor and keyboard. Consoles had a TV and a joystick, so they weren't included in the list.

At that time, the rights management process was not well defined, so Stein sold this version to someone else. It was eventually published, and when my friend Hank Rogers came here for the Game Boy rights and brought the "Tetris" cartridge published for Nintendo, they said to him, "Hey, we didn't give anyone these rights." Actually, it was lucky that he was there because we had just solved that problem. He proved to be quite solid, serious, and honest in business, which allowed us to secure a great contract for handheld machines, the platforms that the Game Boy belonged to. That's why the release of the Game Boy became a significant event in the history of computer games.

Emigration and business in the US

XL: How did you move to the US? Nowadays, stories like yours are quite common in the IT world, but at the time I imagine it was quite unusual.

AP: At some point, I realized that working on speech recognition was a dead end, a wrong choice from the start. What really attracted me was the excellent access I had to the BESM-6 computer. I tried to find an activity that somehow aligned with my gaming interests and had a legitimate aspect to it.

That's when it dawned on me that games are not exclusively technical products and that if they have a connection to a field of science, it's most likely psychology.

I went to psychologists, to the psychology department, to acquaintances of my father who were also psychologists, and said, "Guys, I'm a good programmer, I have access to the computer. Maybe I can do something for you? Are there any methodologies or anything that interests you? How can we develop psychology with the help of computers?" Everyone was interested in this. I ended up with two or



Hank Rogers and Alexey Pajitnov



El Fish by Maxis in all its glory

three business contacts, and we eventually computerized their methodologies. I put a lot of effort into it and gained popularity in the computer psychology community. That's how I met my future business partner, Vladimir Pokhilko. It turned out that we were both invited to create games, and we collaborated successfully a couple of times. We enjoyed working together, and he became my more or less permanent gaming partner.

Vladimir persistently persuaded me to move. We left the USSR in January 1991. During our absence, the country ceased to exist; we didn't even have a chance to look back. Technically, we had nowhere to return to, but the country that had issued our passports and sent us to work was gone. We didn't think about going back because there were so many opportunities; after all, that was why we had come. Later, we received our green cards and settled down.

XK: So, you started your business with Vladimir Pokhilko and devoted yourself to game development in the United States.

AP: We didn't just develop games, but also technologies for games. Our first project wasn't exactly a game; it was called "El Fish," an electronic aquarium with fish. Maxis published it on PC and later on other devices. Creating a visually appealing aquarium on screen was a challenging task at the time. The fish even reproduced and could be crossbred. It was an interesting project.

XK: Did you consider yourself an entrepreneur? Did you enjoy it?

AP: To be honest, not really. It was rewarding to realize I could do it, but I didn't particularly enjoy it. I felt a sense of pride when I managed to create something new on the computer. However, when I managed to close business deals, I didn't feel much satisfaction. That's why I never considered myself a serious businessman. Of course, I could organize something if necessary. After all, I spent a year and a half as president of a company, and it didn't go bankrupt during that time. I understand what business is and how it works. But honestly, I don't like doing it, and I don't want to do it.

The number of games developed by Alexey Pajitnov

XK: I tried to count how many games he developed, but

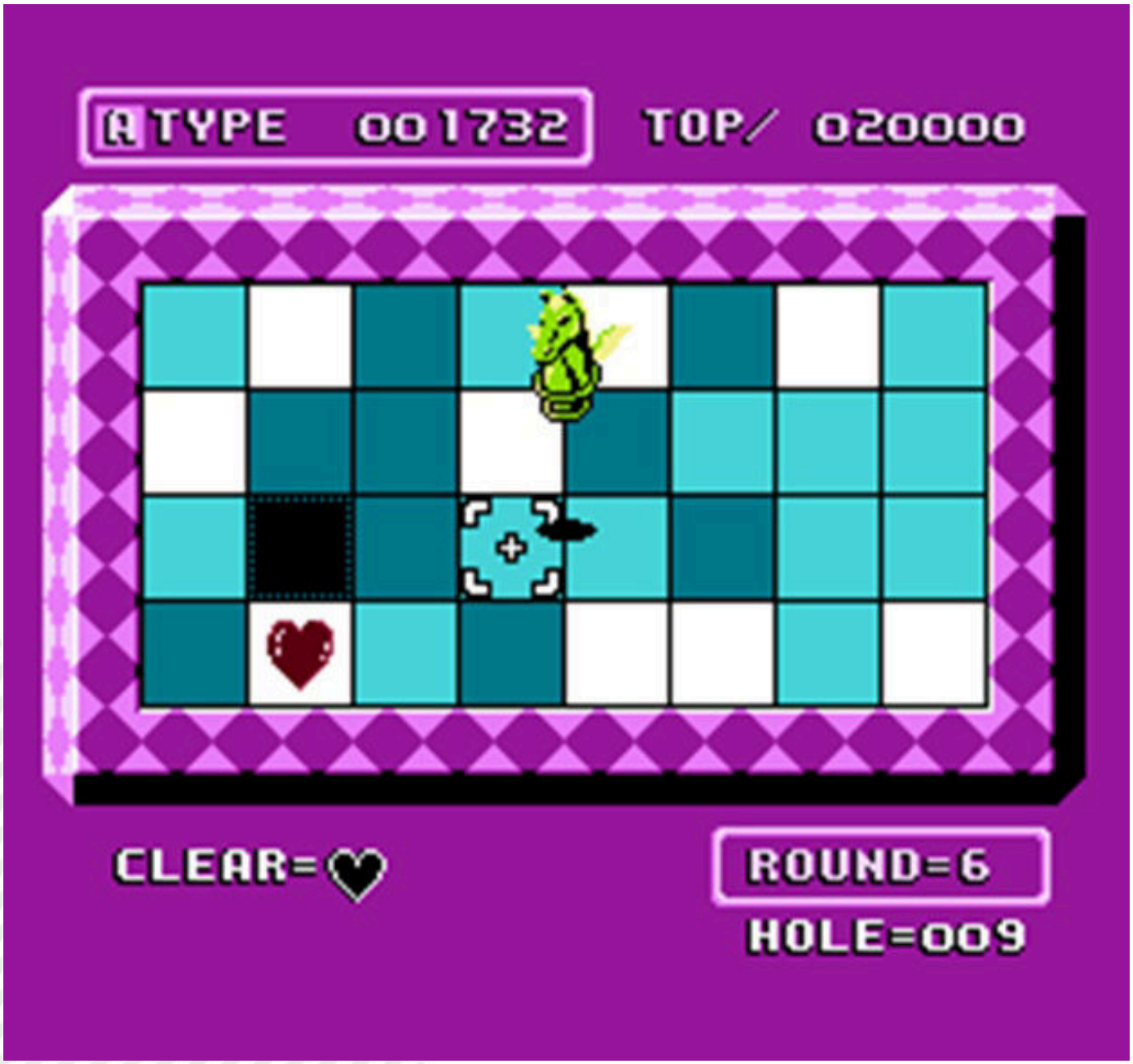
I couldn't. Partly because there are so many, and partly because there are so many variations of "Tetris." It's unclear whether they are separate games or part of a family. So how many games did he develop?

AP: I don't know how to approach this either. It's even harder for me to count them because I love them all. Both those that have been widely published and those that have had limited releases or were created by accident. For example, I developed the game "Knight's Tour." It's a game with real chess pieces, and it's one of my favorites. The joint venture "Dialog" bought it from me in 1987. They gave me a computer with an 80-megabyte hard drive: I was happy because I finally had my own computer. Then this game was released once, believe it or not, on a Nintendo disc.

For a while, the Famicom had a small-scale system on a disc. The disc had a spiral track. It was almost like magnetic tape but on a disc.

The game released was called "Knight Move," which was my "Knight's Tour." Then, with my approval, it was published and republished by Spectrum HoloByte. They got too into 3D graphics and excessive detail, which caused the game to slow down. As a result, it didn't sell at all. So, one of my favorite games wasn't published properly.

Actually, it's one of the first games that remains among my favorites. I made it shortly after "Tetris," around '86 or '87. At that time, I created about five games. Most of them were included in "Microsoft Entertainment Pack:



Knignt Move/Knight's Tour in the Nintendo Famicom Disk System version

The Puzzle Collection" for MS-DOS. I'm very proud of it. It's a bit outdated by today's standards, with a somewhat cumbersome interface, but it's a good collection of 10 honest and very original puzzle games. Six of them were mine, and the other four were created by someone else. In that collection, there is a game called "Muddled Casino." It was my first game ever: a puzzle game.

I invented it even before "Tetris" for the Electronika 60. Then the guys ported it to PC, improving it considerably while retaining the original idea. Even though it's a bit heavy, I have positive feelings about it because it was the first thing I created.

The best games

XK: What is the best computer game for you, regardless of genre?

AP: I love puzzles, and about every two months, I browse the App Store looking for something that catches my eye. Although, honestly, browsing it is disappointing because the offerings are pretty primitive and there aren't many interesting ones.

There was Monument Valley, a beautiful black-and-white puzzle game. And also a pretty decent game called Empires and Puzzles.

I've been playing it in small doses every morning for seven years and have already reached the highest possible levels. However, I don't have particularly warm feelings towards it; it's more of a habit.

Another is "The Witness," a collection of puzzles combined with 3D exploration. You walk through a meticulously designed space and encounter puzzle screens that you have to solve. The puzzles range from very simple to extremely challenging. Then I like Warcraft. I love that saga.

XK: If you don't mind me asking, what level did you reach in Warcraft?

AP: I wasn't in the top 100 players, obviously, but I probably achieved everything a normal person could achieve in Warcraft. I got hooked on it and then made friends with people who played together. But I was also interested in Warcraft as a designer, so unlike normal players, I took a broader approach. I studied players and characters and tried to explore all the quests for every specialization in the game. I played not only to win, but also so I wouldn't miss anything.

Advice from a normal guy who wants to get into software development

XK: We're gradually coming to the end. I hope our interview will be read by future game developers. Do you have any advice for them?

AP: I'm just a normal "guy," not a genius, guru, or da Vinci, someone who has stumbled many times. I have experienced many failures and created both ugly and beautiful games, successful and unsuccessful ones. I hate giving advice, especially in a creative field such as game development.

So I can only give one piece of advice: listen to yourself. The best and most important things can be achieved when you don't try to do something better than others, but, on the contrary, aim to create something you can't tear yourself away from. This approach shapes a serious developer.

Japan Episode 25: Gachagacha...pon!

by Michele Ugolini

Dear readers, are you ready for an exciting episode on Gachapon?
In this second episode dedicated to gachapon, I will talk about the "adult" content of some machines in Japan. Yes, you heard right, among the many wonderful miniatures that can be collected, the Japanese have brilliantly thought that objects related to the world of red lights could also be collectible. Clearly, this is done in a different way from our Western interpretation, so... let's proceed in order.
As already mentioned in the previous article, the Gachapon craze has been back in Japan for a few years now. We Italian collectors will remember the Exogini: miniatures of wrestlers from space, a trend born from the original Japanese idea of Kinkeshi.
In addition to these miniature figures, in Japan there are entire production lines of miniatures of almost any object imaginable: miniatures of kitchen utensils, miniatures of animals, miniatures of mascots, etc.
So... what miniatures are missing from this endless list? Miniatures of human anatomical parts are missing, as well as the accompanying adult toys to associate with the various anatomical parts that our imagination and perversion could bring to life in our heads. Do you think I bought a ball of these adult gachapon? No, I didn't, but on my next trip... I will!
These gachapon machines can be found along the main tourist shopping streets, just like the classic machines available to the public. The only difference from the classic gachapon machines is the location of these adult machines: always next to an open store. I have only seen these stores in operation during the day, so I am not sure if it is permitted to leave these machines unattended outside the store during closing hours at night. Adult gachapon are only available to adults, and the distinctive feature of the machine is that there are no images of the contents of the various balls. There is a nice red inscription on a black background, and the ball that comes out will contain an adult item, at the modest price of 500 yen.
I saw these adult gachapon in the tourist district of Namba, not far from Osaka. Apparently, and above all, personally, based on my travels, I don't remember finding any in Tokyo.
For the record, as well as for my blog, I would like to mention the presence of these adult machines, also called "Eropon," installed on the landing next to the stairs of the adult floor of Nobunaga Shoten in Teramachi. In addition to the distinctive graphics of the Eropon, the capsule containing the ball is also unusual: it is completely black, and you cannot see what is inside until you pull it out of the ball tank. The machine said, "For only 500 yen, you can get the most erotic smile of your life. Fear nothing, take it!" From blogs, I read that some prizes are quite expensive and when you receive a voucher, you can exchange it for a prize at the cash register. In the Eropon machines I saw, there were many empty black capsules in a basket next to the machine, so buying these balls seems to be quite popular. Or perhaps the owners of these machines have placed the open capsules there specifically to show future buyers the success of that product.
I am attaching a link to Nobunaga Shoten so you can browse Japanese erotic fantasies. https://www.e-nobunaga.com/



I will also show you a series of images of all the "surprises" inside the darkened boxes, obviously in Japanese style, extremely perverse and highly pixelated. Just kidding! On second thought... we can allow ourselves one image of sexy panties in our magazine. There's nothing wrong with that compared to what might actually be contained in other black boxes. And here is the answer at last, after endless doubts and an effervescent imagination that has already gone into orbit. Sexy panties! Yes, in these boxes, of this particular Eropon, there were panties, women's panties, probably cute, certainly uncomfortable, made of synthetic fabric, not very "green." Who knows how many washes they will withstand? Who knows if they will ever be worn? Who knows how many people who inserted coins would like to get their coins back? In the meantime, here are two links you can consult regarding the map of these gachapon in the city.
https://islaywhiskey.hatenablog.com/entry/2018/02/10/103133
https://logblog.blog33.fc2.com/blog-entry-650.html
This type of vending machine is probably found in major cities, especially in streets dedicated to tourists, streets full of knick-knacks that can be found in any Chinese shop near your home. Friends, while we let our imaginations take flight, I invite you to visit Japan and discover the soul of the Land of the Rising Sun, starting from its origins and magical millennial traditions, and why not, discover what these mysterious balls contain. See you next time, dear friends, for an article entitled Gachagachagacha-pon, where we will talk about the peculiarities of these mysterious and very popular machines.
See you soon!









Fireteam Rogue for Super Nintendo, ambition without action is nothing.

The story of a title with incredible potential!!

by Takahiro Yoshioka and Evan Granovitz - Carlo Nithaiah Del Mar Pirazzini

Fireteam Rogue was a promising action-adventure game developed between 1994 and 1999 by Accolade, which unfortunately never saw the light of day.

One of the most impressive things about this game is the incredible amount of concept art available, which demonstrates the quality and passion put into this product. Almost all of the game's artwork was created by the late visual artist Betty Cunningham and is available on her website at: http://www.flyinggoat.com/main.html.



Fig. 1 - The character Shadowblade by artist Betty Cunningham

our time developing screens with ridiculous data to adapt the game levels, when in reality we should have cut enemies and levels to make it work.

There were also personal conflicts that culminated in the project producer leaving just as we were releasing the beta version. Shortly after, the product manager at Accolade questioned whether the game was worth all the effort and how much longer it would take to finish it. It wasn't clear to me how much work it would take to finish it and whether the title would be able to compete in the market, so we decided to shut it down."



Fig. 2 - One of the complete levels of the game available with the Alpha version

Excessive ambition

The cancellation of Fireteam Rogue was due to excessive ambition. The developers (or rather, the people in charge of marketing the game) were running amok with grandiose claims: 100 hours of gameplay, characters larger than most other action games, digitized voices, in-game speech, and much more.

All this, combined with poor team management, constant internal disputes, and the end of funding, led to catastrophe. Russell Borogove (one of the developers) still says today: "The project had problems in development. We spent all

The aforementioned Betty Cunningham stated on her website that the game was complete.

Graphic designer Scott Ruggels recalls: "Both game projects (Fireteam Rogue and the unreleased Megadrive game, Cybernauts: The Next Breed) were directed by John Skeel. I don't know what happened to him after Accolade, but after the $2 million budget for the game was spent, with about $750,000 spent on promotional materials, including a 6-foot-tall plastic roll with a life-size computer-generated image of the main character inside, and the prototype, which, in all honesty, wasn't

much fun to play, the game was canceled, along with many others shortly after the new management took over, and we went our separate ways, with me ending up at 3DO during the final days of development of "Real Player." The title was finally abandoned in 1996, but the project continued for a couple more years.

Marketing

Fireteam Rogue was a lavish and pompous game. During the early 1990s, there was a new boom in the comic book market (partly due to Image Comics and the "rebirth" of DC and Marvel).

Accolade intended to capitalize on this sector as well. It created a comic book that introduced the story. A fifteen-page comic book that would be released along with the game manual itself on SNES.

In addition to the comic book, there were plans for action figures and even a television program (it is unclear whether this was to be live action or animated).

Money, money, money. Add to this a hologram created for a tour called the EGM Super Tour, which focused on comic book conventions and large shopping malls.

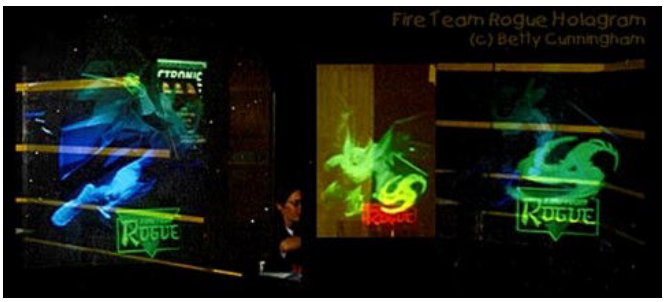


Fig. 3 - Hologram used at the EGM Super Tour

Fireteam Rogue received a "lukewarm" reception in the media at the time. Nintendo Power magazine featured it in February 1994. In a lengthy preview, they showed some images from the game and noted the "limited color palette reminiscent of Megadrive titles."

Game Players in October 1994 praised the title for its "serious potential" and noted that the characters of that size were innovative.

Nintendo Power itself, but in 1995, criticized the slow development and strange, jerky graphics of the game.

The first prototype

The prototype was revealed to the retro community in 2005 and showed the incompleteness of development at

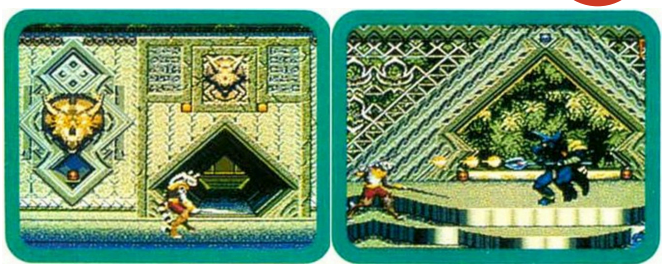


Fig. 4 - First images of Fireteam Rogue from the February 1994 issue of Nintendo Power (scanned from Retromags)

the end of 1994.

The ROM image is available on several industry websites, including SNES Central.

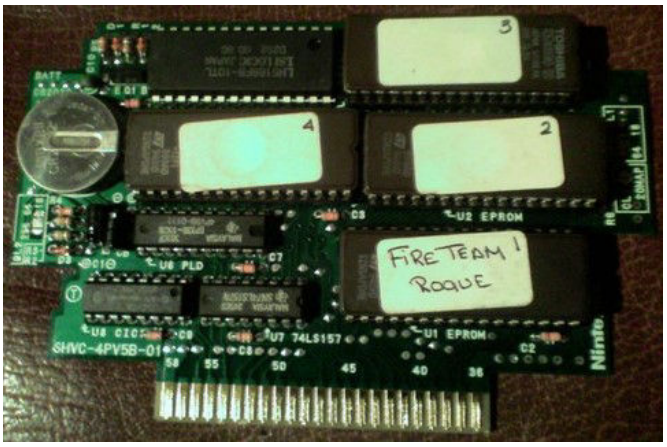


Fig. 5 - The first prototype of the cartridge

It shows several levels, but only one appears to be completed correctly, that of Shadowblade.

The sound effects are rather minimalist, although the music is well done. The game is minimalist: no extra lives

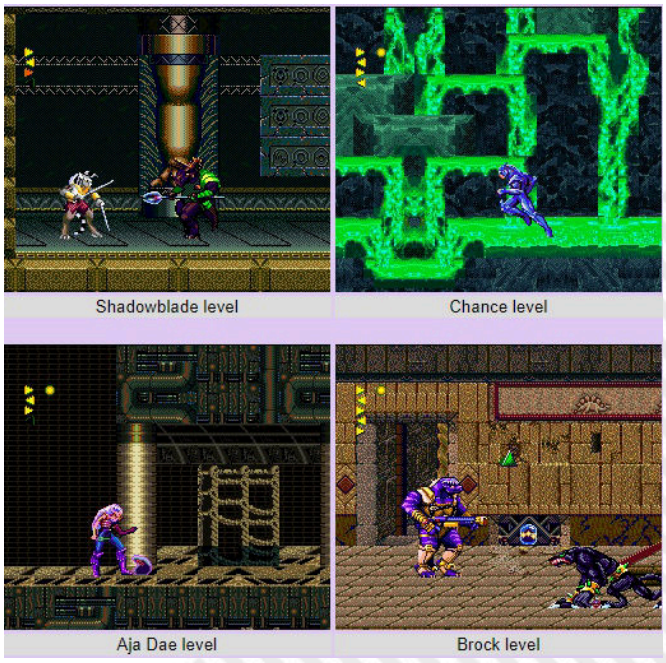


Fig. 6 - An example of the levels available in the prototype

or continues. The animations are a bit jerky and the controls are barely decent. There is even a lag in the character's movement.

The graphics are bland and the color palette is limited.

Next Prototype

The second Alpha arrived in 2010 and showed a level of development similar to that of late 1995.

This Alpha 2 showed a more advanced state of completion but the same gameplay as the previous one.

The Shadowblade level was slightly modified compared to the first version, while the other three levels showed additional states of completion but remained unfinished.

The main addition to this version is the use of MODE 7 for some stages of the game.

In the flight stage, you have to jump into an aircraft. Quite successful.

These shooting stages are a pleasant novelty in a huge mass of ambition.

Passwords have been added to continue the game, but continues are still missing.



Fig. 7 - Here we can see the flight phase

The end

As we have seen, it all ended in arguments and a lack of funds.

Fireteam Rogue had all the makings of a promising title.

A beautiful fantasy plot, graphics that were impressive (in appearance) for the time, and a very strong marketing push.

The shooter levels worked quite well and could easily be compared to other similar titles for the SNES.

That said, the available Alpha ROMs show a game that we cannot define as "playable."

A product with a huge development time and definitely out of context. The Super Nintendo was still active on the scene, but the new generation of consoles had already been on the market for some time, and that's what people wanted to see.

The last Alpha was shown at the 1995 Winter CES and was savaged by critics.

Poor level design, jerky graphics, and few ideas. All this was Fireteam Rogue.

The lesson the game teaches us is that focusing too much on hype and story before creating solid gameplay can sink the ship.

The story rivaled most RPGs of the time, and the ideas could have steered the title towards a "modern" version of Super Metroid.

Instead, a development cycle mired by mismanagement, delays, and bombastic proclamations made this title another footnote in 16-bit history.

You can find the ROMs here:

First beta:

https://mega.nz/file/17YA2ZSb#YaDOt0d8rHIuH7NOdsACY8a6GXieHz9m5fsmwa_Yyk8

Second beta:

https://mega.nz/file/sqBDWZDb#MO742MjE9ZWReEwv3585kjG6MU1Smk4Hu0ncHLXyA8E

Sources:

- Betty Cunningham's website with lots of concept art.
- Interview with Jim Barnett, discussing various Accolade games (including Fireteam Rogue) at the 1994 Consumer Electronics Show in Chicago.
- Electronic Gaming Monthly, Preview - includes several screenshots, Publication date: September 1994, Volume: 62, Page: 150.
- Gamepro, Preview (includes four screenshots, none of the scenes are present in the two prototypes), Publication date: February 1995, Volume: 77, Page: 157.
- Nintendo Power, Pak Watch (preview, mentions the

ambitious gameplay and storyline and the "limited color palette." Shows two screenshots of the Shadowblade level and concept art of Broc.), Publication date: February 1994, Volume: 57, Page: 111

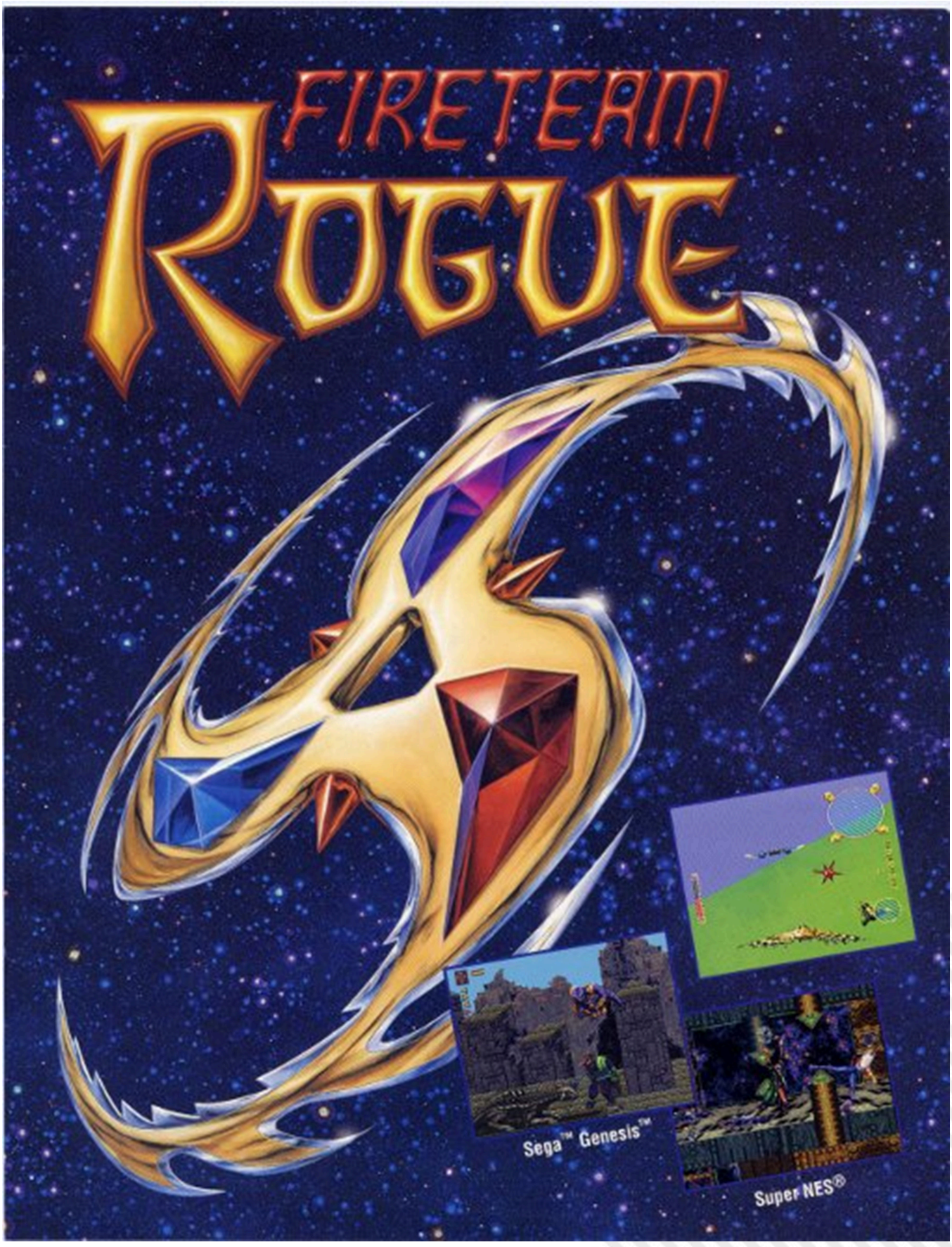


Fig. 8 - The game presentation sheet at WINTER CES in 1995

FIRENZE VINTAGE BIT 2024

edited by Associazione Firenze Vintage Bit Onlus (written by Alberto Apostolo)

For all retrocomputer enthusiasts and collectors, we are pleased to announce that on Sunday, December 1, 2024, the sixteenth edition of Firenze Vintage Bit will take place in Lastra a Signa (Florence).
This year, the event will be held in the premises assigned to the Santa Maria a Castagnolo Primary School located in Via Gramsci 139 in the municipality of Lastra a Signa (Florence).
The aim is to entertain exhibitors and audiences of all ages by displaying computer and console models that have made computer history and calculating machines that are usually seen in museums.

Admission is free (good manners and common sense are required when accessing the premises and exhibitors' stands).
Please note that Firenze Vintage Bit is not a market and is not for profit. Enthusiasts and collectors wishing to make exchanges should go outside the area reserved for the event.

How it works

It starts on the Saturday before the event opens to the public. Registered exhibitors bring their computers and consoles, set up their assigned tables, connect their peripherals, and check the connections. It is also an opportunity to try out other exhibitors' machines and exchange opinions and suggestions on software or hardware adjustments.
The day ends with a convivial dinner at a characteristic local restaurant.
Sunday (from 9:00 a.m.) is dedicated to the public of all ages, who can try out the machines on display and ask questions of the exhibitors, who will be happy to answer and recount entertaining anecdotes. The closing from 1:00 p.m. to 2:30 p.m. is reserved for a lunch break together (reservations must be made in advance of the event with the organizing staff).
The event ends at 6:00 p.m. to facilitate the return of exhibitors who have traveled from further afield.

Firenze Vintage Bit is not limited to the exhibition alone. Each edition is associated with a theme, or a main topic to be illustrated.

The theme of this edition is "Simulators and Education."

In the morning, there will be a 45-minute conference entitled "A journey into the world of simulation," offering an overview of the different types of simulations with applications in the educational context. In the afternoon, there will be another short presentation (on the computer strategy game "The Sumerian Game") followed by a musical essay (video game soundtracks).

A brief history of past editions

After participating as an exhibitor at Varese Retrocomputer in 2007, Walter Pugi came up with the idea of creating an event in central Italy (possibly in Florence). With the invaluable help of Maurizio Morandi, the first edition of Firenze Vintage Bit was held in Campi Bisenzio (Florence) in 2009, achieving unexpected success (repeated again in 2010 and 2011). From 2012 to 2022 (except for the 2020 edition, which was organized as a video call due to COVID-19), the editions took place in the evocative medieval setting of the Antico Spedale di Sant'Antonio in Lastra a Signa (Florence). In 2023, it was hosted in Signa (Florence) in the halls of the magnificent Villa Alberti.
This year, however, it returned to Lastra a Signa, having received the availability of the Santa Maria a Castagnolo Primary School (SMAC).

As mentioned above, past editions have also been characterized by a theme,



Fig. 1: FVB 2009. FVB founding group with some guests. From left: Stefano Donati and Luca Bisti, who brought the prototype of the RealSpectrum emulator they had created, Walter Pugi (creator and founder), Gianni Zamperini (promoter), Maurizio Morandi (founder), and Alberto Sarcinelli (founder).



Fig. 2: FVB 2010. Exhibitor's table.



Fig. 3: Antico Spedale di Sant'Antonio.

addressing topics such as graphics, music, programming, and the anniversaries of the most iconic models. Finally, there have always been musical and choral performances that have been highly applauded by listeners.

The Firenze Vintage Bit Association

Established in 2016, it received ONLUS (Non-Profit Organization of Social Utility) status in 2017. The association supports the organization of the event and acts as a liaison with local administrations and other institutions.

Over the years, the tireless work of its volunteers has enriched the association's program with various initiatives carried out at its headquarters in Lastra a Signa (courses in the history of information technology, programming courses, and a computer and peripheral repair workshop). In addition, the association collaborates with schools and is a guest at various social events in Florence and Tuscany, always receiving enthusiastic approval from the public.

Since 2024, the Firenze Vintage Bit Onlus Association has been twinned with the Varese Retrocomputer association, in memory of the first exhibition organized by Walter Pugi and to share ideas and opinions in order to continuously improve the range of activities and events on offer.

The Firenze Vintage Bit Onlus Association has a Facebook group

https://www.facebook.com/groups/firenzevintagebit

and a YouTube channel

https://www.youtube.com/channel/UCUQzraVIBquit-KnFIRrDTw

Useful information

Information about the event (including directions and registration details for exhibitors) can be found on the website

www.retrocomputer.org





Fig. 4,5: FVB 2018. Inside the Old Hospital of St. Anthony.



Fig. 6: FVB 2018. Group photo of exhibitors.



Fig. 7: Repair workshop (Marzo 2023).



Fig. 8: "Renai in Fiore" event (Aprile 2023).



Fig. 9: "School Festival" event at SMAC Primary School (May 2024).

Con il patrocinio del
COMUNE DI LASTRA A SIGNA
Assessorato alla Cultura

Firenze Vintage Bit 2024

16° edizione

SIMULATORI E DIDATTICA

Graphic Design: Walter Pugi



Lastra a Signa
Scuola Primaria
Santa Maria a Castagnolo
1 dicembre (09:00-13:00 / 14:30-18:00)

FVB RETROCOMPUTING
www.retrocomputer.org

Lastra a Signa
Città di Enrico Caruso

info@retrocomputer.org
www.retrocomputer.org

GRAFIKART FORNITURE PER UFFICIO
Radio Incontro
SPONSOR UFFICIALE
vignolbags

NEW GAME

QUINTY

Year: 1999/2024
Editor/Developer: Game Freak
Genre: Puzzle game
Platform: Super Nintendo

The unreleased SNES remake of Game Freak's (the creators of PokÃ©mon) debut game has finally been recovered.



Quinty (also known as Mendel Palace in the US) was originally released for Nintendo NES back in 1989 by the creators of PokÃ©mon and later ported to SNES with a technical upgrade to suit the 16-bit Nintendo platform.
It is a puzzle-action game in which the player controls a character who flips tiles on the floor to defeat enemies. It is a strange way to "fight," but it is enjoyable and has interesting mechanics.
The SNES version of Quinty was never officially released, making this online rediscovery particularly interesting for fans and those who preserve software for this console.



The 'illicit' recovery in October 2024 at the expense of Game Freak has definitely brought a good title to light.

Aesthetically, it is well made and colorful. Very 'kawaii' in appearance and well animated. There is good sound and everything is quite intuitive, even if it is entirely in Japanese.



The game concept is interesting and the mechanics, once learned, engage even the most skeptical (like me) of players.
But beware! The title is suitable for puzzle game enthusiasts and lovers of the Japanese world, as well as those familiar with the NES/Famicom version (definitely a niche market in Europe).



For everyone else, it's a title to know about and preserve.

by **Roberto Del Mar Pirazzini**







OUR FINAL SCORE

» Gameplay 80%
The game mechanics are very enjoyable and exciting. The multiplayer version for up to 4 players via multitap is interesting.

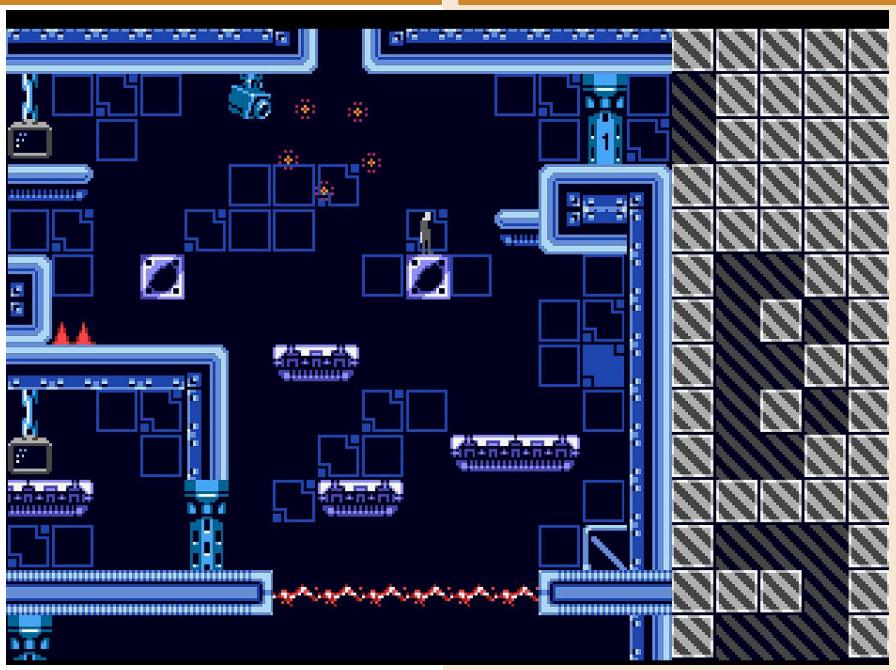
» Longevity 75%
The Japanese language is not a limitation, but it is certainly not immediately accessible. Some levels can become monotonous.

NEW GAME

ESCAPE 2042

Year: 2017
Editor/Developer: Orion Soft
Genre: Action/Puzzle
Platform: Atari Jaguar
Website: https://www.orionsoft.games/

In 2042, a totalitarian government is in power. The ruling elite hides behind a fictitious democracy, disconnected from the lives of the people and devoted solely to its own interests.
The regime considers any dissident a conspiracy to be eliminated. Anyone who does not think like them is considered a terrorist and locked up in maximum security prisons.
Shun, a computer engineer and member of the Truth Defenders coalition, imprisoned in Bulor 24 prison, escapes.
Of course, my friends, we will be the ones to guide Shun in this daring escape.

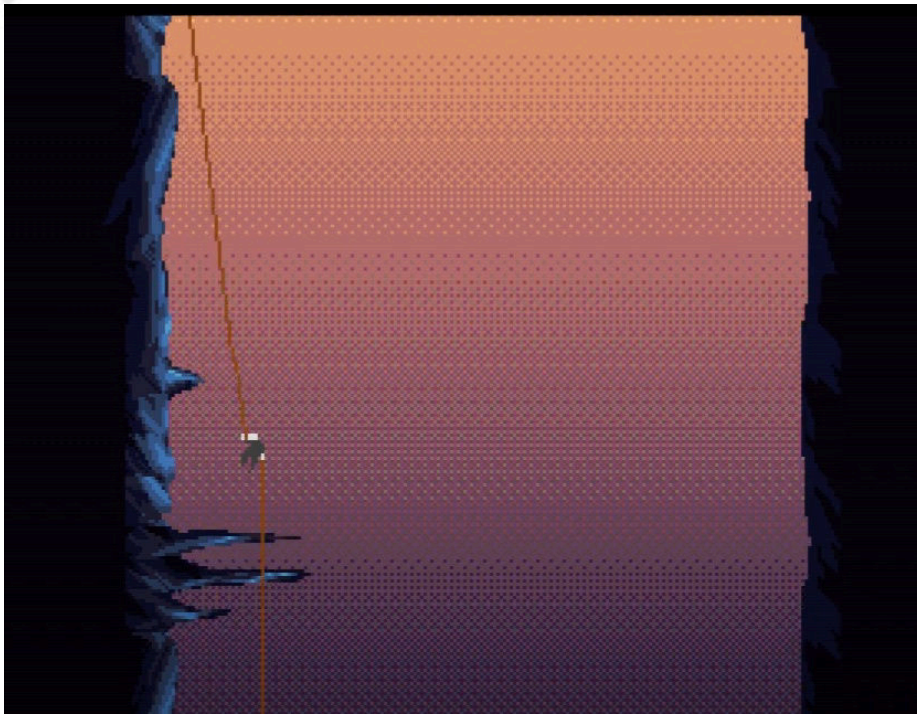


Escape 2024 takes place in three different settings (prison, desert, forest) and is interspersed with two mini-games, including a shoot 'em up and a pure action game. The aim of the game is to survive by avoiding any contact with the guards.
Avoid security cameras, guards, and traps. Along the way, we can collect useful items such as grenades or exercise our hacking skills by blocking the computer system or deactivating security systems.
There are two difficulty levels. Normal allows for a more "simplified" way to complete the adventure, while Hard mode has no weapons to defend yourself with and the mini-game sequences last twice as long.
This title looks like a platform game but actually incorporates many different elements and styles.

I found the same tension as in Impossible Mission in the early levels of the game, which is a good thing.
This Jaguar version is identical to the other versions released for Sega Megadrive, Dreamcast, and other retro platforms.







OUR FINAL SCORE

» Gameplay 90%
A mix of different game styles with various mini-games and a good degree of challenge.

» Longevity 85%
A title that will keep you glued to your gaming station for quite some time.

The controls are precise, and the development of puzzles and levels is well-structured and intelligent. It's a title that never severely punishes the player but requires attention and study of the level.



The graphics are very refined. 8-bit pixel art with 16-bit accents. It's hard to explain, but personally, I really liked this artistic style, which suits Escape 2024 perfectly.

The soundtrack is also commendable. Although it doesn't have too many tracks, it's really good and clean with a crisp, clear sound. It has a very retro 80s style that I particularly loved.

If you missed this little gem in 2017, remedy that immediately. Escape 2042 deserves to be played on your Jaguar.

by **Giampaolo Moraschi**

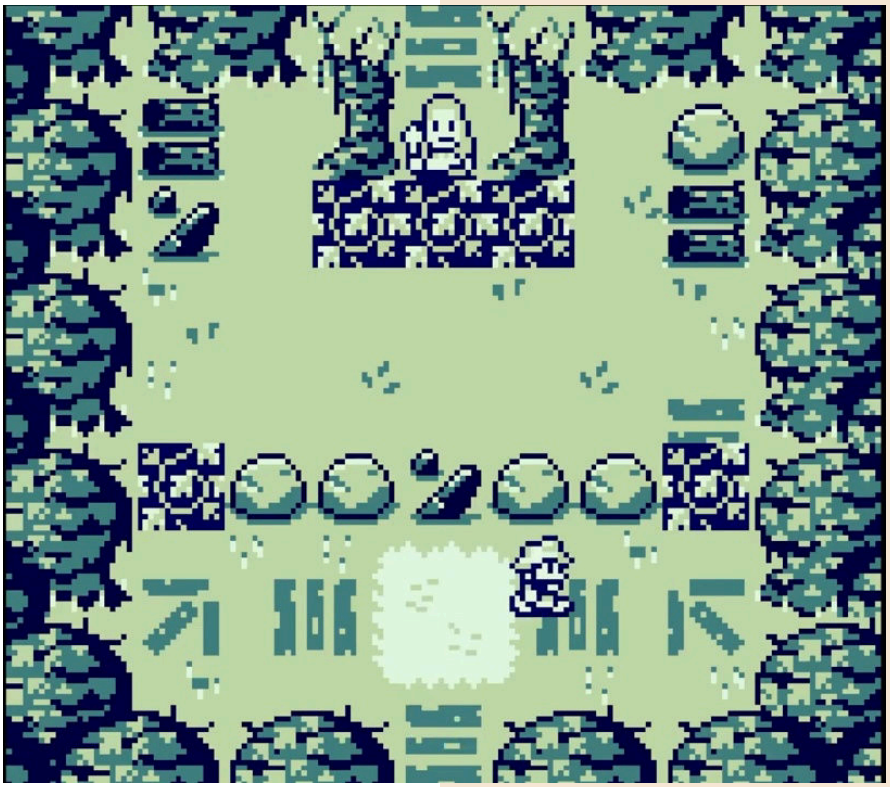




NEW GAME

KUDZU

Year: 2024
Editor/Developer: MegaCat Studio
Genre: Action RPG
Platform: Game Boy



Kudzu, the latest Zelda-like game from MegaCat Studio, goes beyond simple nostalgic evocation. This title features a lot of "Game Boy" game design. It feels like playing one of the titles from the glory days of the little handheld console.

Players looking for something beyond the general vibe of early 90s games will see the appeal of Kudzu, with its top-down view and easy-to-understand mechanics.

In the title, the player takes on the role of apprentice gardener MAX, who embarks on an adventure to save his mentor, Zoen, from the dangerous Kudzu plants that are infesting planet Earth.

Mechanically, it plays a bit like Link's Awakening. As the player progresses through the story, they acquire gardening tools that allow them to attack enemies and eliminate environmental hazards.

Each new tool replaces the previous one and grants access to new parts of the map, much like in the worlds of Zelda, offering the player new paths to take. The dungeon areas require puzzle solving and boss fights. While the former are not difficult, I must admit that defeating the end-of-dungeon bosses will make you sweat buckets.

I found Kudzu's narrative approach and gameplay progression very interesting. You start the game, discover the basic quest, and then set off on your adventure. The story starts in the simplest way possible and becomes clearer as you play.



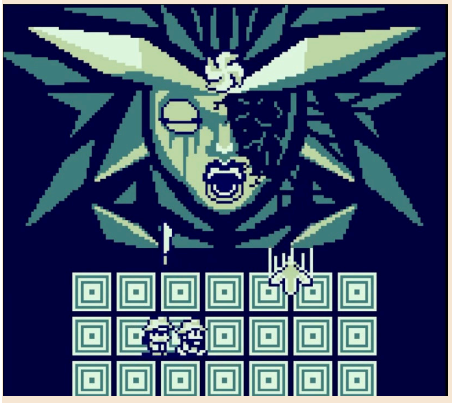
Unfortunately, the title suffers from a certain repetitiveness in the basic battles, which dampens the enthusiasm for adventure and discovery.

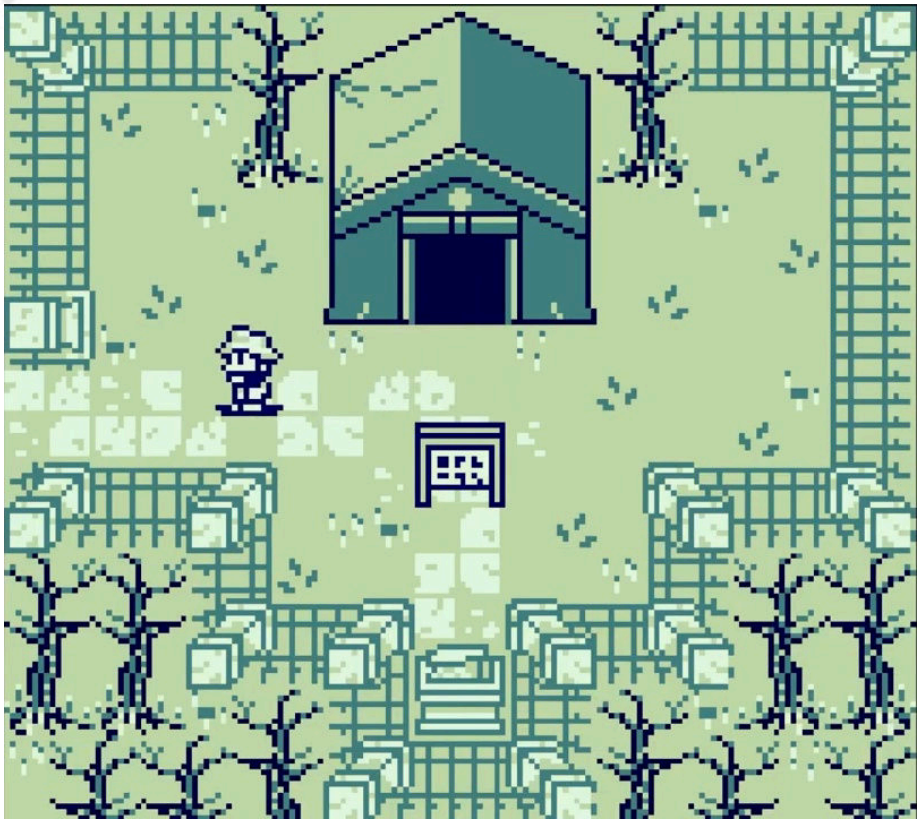
Technically, it is a well-developed product with varied graphics and good animations.

There are a few glitches in a couple of boss fights, but nothing that compromises the understanding of the battle.

The music is nice, but the sound effects are very low and subdued, almost absent.

It is a title with a lot of character and lots of details and surprises, and it will guide you on a beautiful adventure.

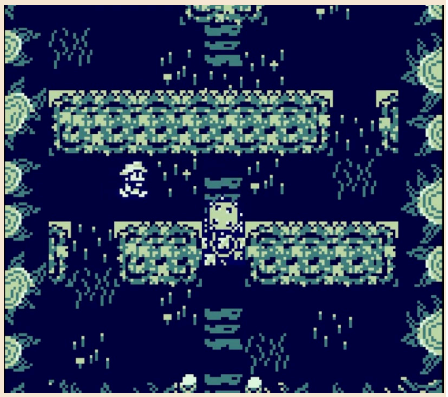




OUR FINAL SCORE

» Gameplay 75%
A beautiful story, written with care and well developed in the game.

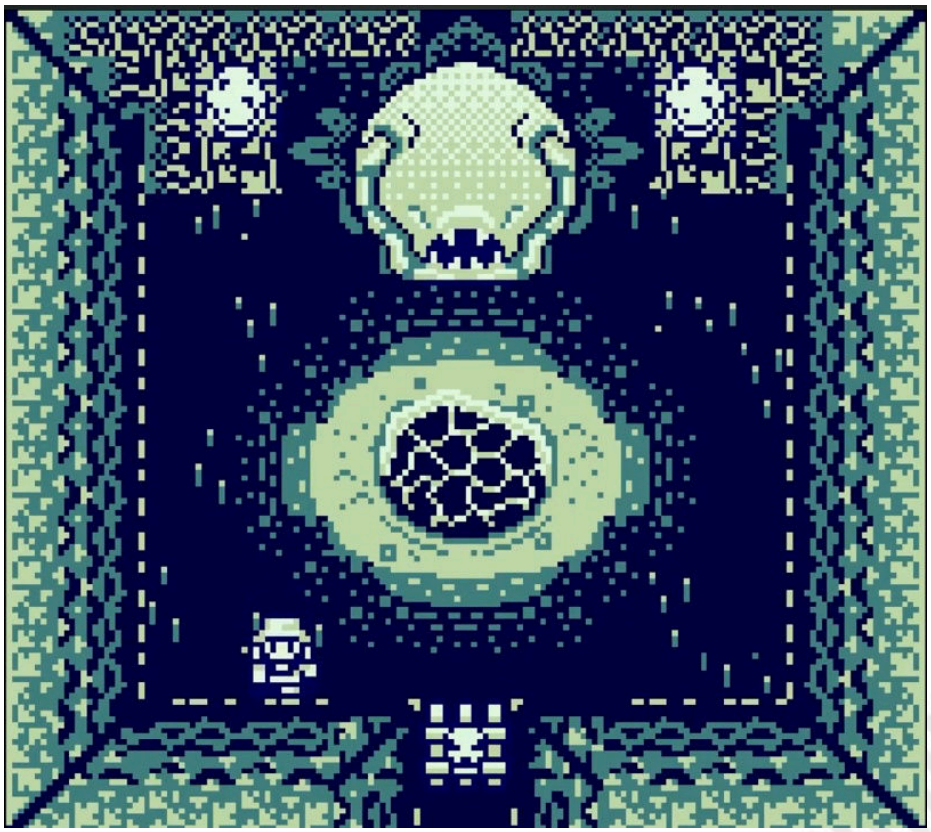
» Longevity 70%
The boss fights are really tough (not all of them) and there are some unpleasant graphical glitches.
There is also a lack of "appropriate" sound effects.

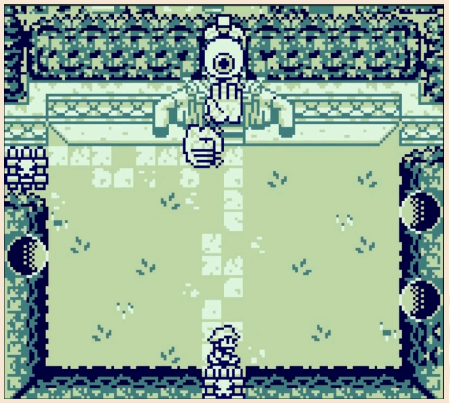


It may not be the heir to Zelda, but it doesn't claim to be.
Give it a try.

Website: https://megacatstudios.com/products/kudzu?srsltid=AfmBOopYqRDZu3ZUOUJgezFKPuDvq_jCVTCY4ZU2jZCpNCmVLCMXw2au

by **Roberto Del Mar Pirazzini**







NEW GAME

ZEPHYR'S PASS

Year: 2024
Editor/Developer: Incube8 Games, CRTOGRPHR
Genre: ACTION RPG
Platform: Game Boy Color
Website: https://incube8games.com/products/zephyrs-pass-gbc-digital-edition

Zephyr's Pass is one of the new console JRPGs that has captivated me the most lately. It's an exciting title with a beautiful narrative, created to be played on Game Boy Color.

It mixes Final Fantasy, Zelda, heroic warriors, fearsome and funny sky pirates, kidnappings, personal redemptions... In short, it's a game with lots of action that is very easy to play.

You start out as Abel, a young hero and delivery boy who finds himself, despite himself, involved in a story much bigger than himself.

He must join forces with sky pirates to defeat Thane, a mad despot who wants to replace the God of Wind.

In order to defeat him, Abel must recover the ancient pieces of the wind armor, each of which gives him new abilities and the strength to save his land (and the damsel in distress).

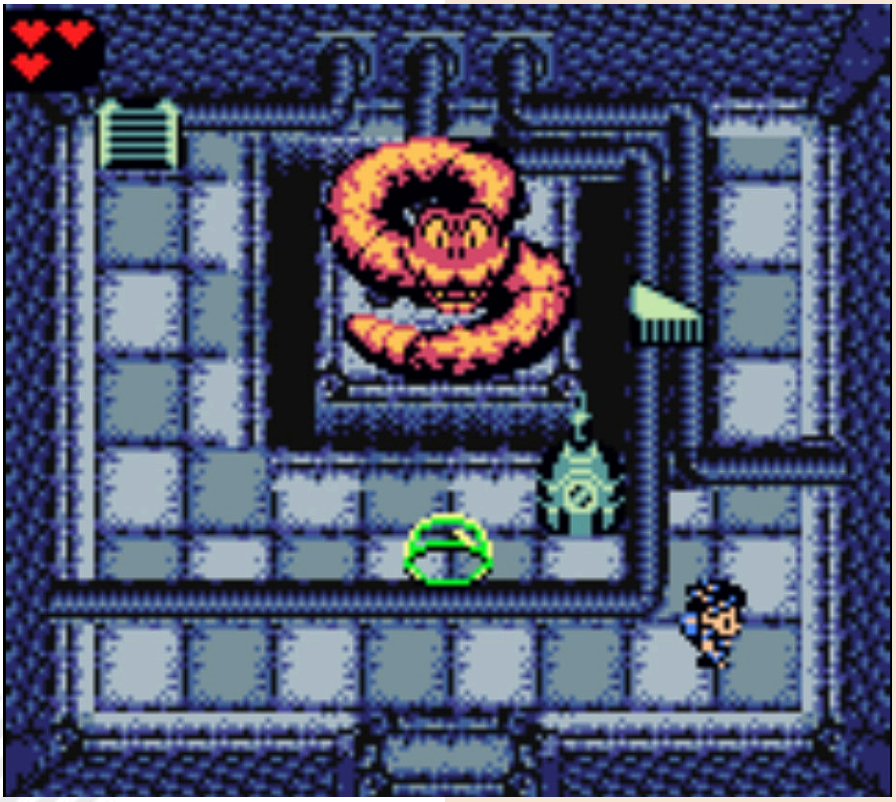
Disarming in its simplicity of gameplay and use of the few buttons at our disposal, Zephyr's Pass has proved to be a wonderful surprise among the new titles released in recent months.

It has a nice menu system that allows us to keep track of the map, equipment, saves, and quests by pressing SELECT.

The directional button moves Abel, the B button uses the sword, and the A button is used to defend and use items or, in the case of non-player characters, to dialogue.

The developer has poured all their love for the Nintendo platform and the great titles of the past into this game, giving us beautiful aesthetics using the very few colors available

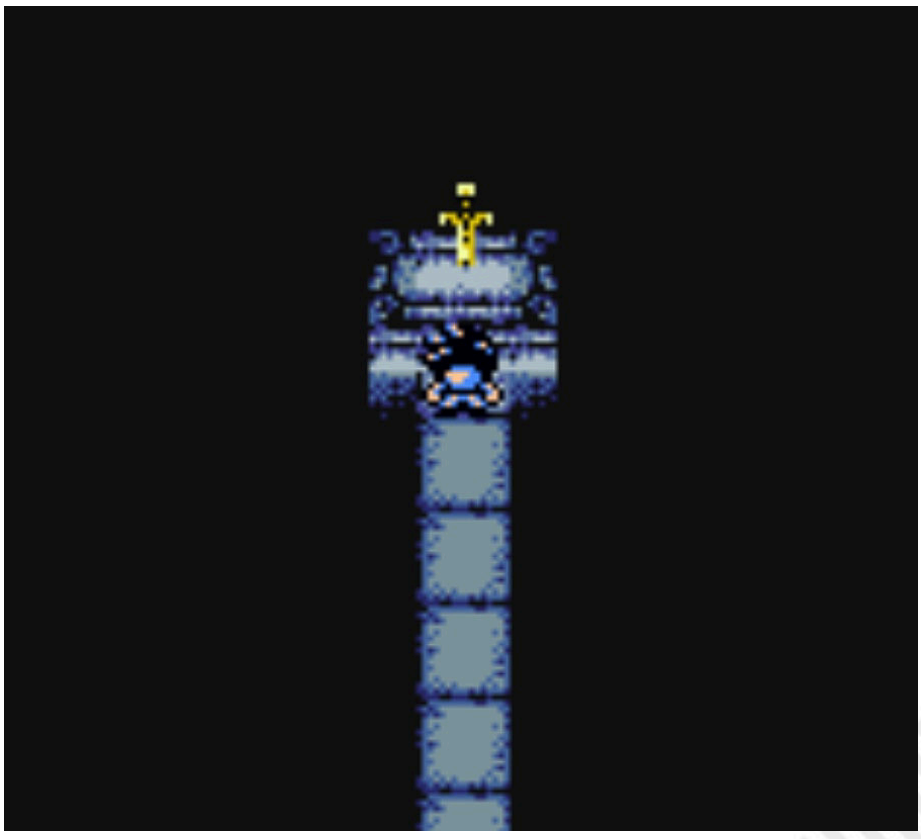






on the Game Boy Color.
The animations, narrative screens, background graphics, and characters are truly beautiful.
But the game's strong point is its narrative, which grabs us and never lets go. The story of Zephyr's Pass is meant to be played in one sitting and experienced.
The puzzles in the game are well designed, and the action sequences are simple but satisfying.
An almost flawless title available for digital download and in physical format on cartridge.
Perhaps the only real flaw in this case is the very high cost of the physical medium.
However, traveling through the clouds in a flying ship is a unique experience. Don't hesitate to try it.

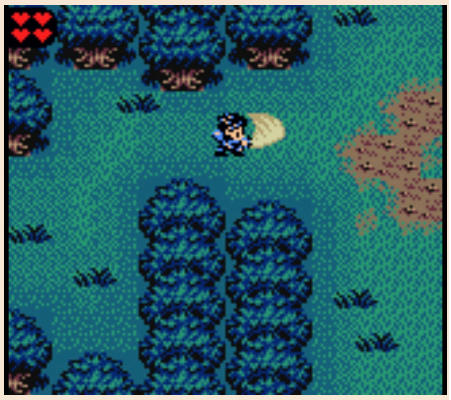
by **Carlo Nithaiah Del Mar Pirazzini**



OUR FINAL SCORE

» Gameplay 90%
Simple, intuitive, and with good gameplay mechanics.

» Longevity 90%
The story is well written and it's difficult to put the game down before reaching the end.









NEW GAME

KING'S QUEST VI
HEIR TODAY, GONE TOMORROW

Year:1992/2024
Editor/Developer: Marko/Atarimania
Genre: avventura grafica
Platform: Atari ST
Website: https://www.atarimania.com/game-atari-st-king-s-quest-vi-heir-today-gone-

It's nice to revisit King's Quest after all these years. The Sierra Online saga that was all the rage in the late 80s and early 90s.

This sixth episode, called "King's Quest VI: Heir Today, Gone Tomorrow," was first released in 1992, written by Roberta Williams and Jane Jensen.

It is recognized as the high point of the series for its historic 3D movie created by Kronos Digital Entertainment and the voice acting of actor Robby Benson, who provided the voice of the game's protagonist. It was also the last Sierra Online title to be released on floppy disk.

I was therefore amazed to see it again after so many years in the Atari ST version. A version created by Marko/Atarimania who took the original files from the PC version and the AGI Quest launcher.





It is a demake of the original.

It took 18 years of work to release it (from 2006 to today) and the result is excellent.

King's Quest VI was assembled and





OUR FINAL SCORE

» Gameplay 80%
One of the most iconic sagas for this genre of games. Perhaps a little dated in terms of gameplay, but still beautiful.

» Longevity 90%
There is so much to do, explore, and see. Including multiple endings.

compiled using Agi Studio. This was a development tool used by Sierra On Line for its early adventure games.

When AGI debuted in the early 1980s, it was innovative technology for its time and was released not only on PC but also on Apple II and Apple IIGS, Amiga, Atari ST, and Mac. It was a simple game engine that included a language called Game Adaptation Language, a compiler, and an interpreter. The language was high-level and resembled C. It was compiled into bytecode and executed by the interpreter. In 1988, with the launch of SIERRA'S Creative Interpreter, it was slowly abandoned.

Returning to the title, we can say that it has everything (or almost everything). The kingdom of the Green Islands, the conspiracy, the quests, and even the multiple endings.

There are a few bugs, but nothing terrible. In fact, the developer invites anyone to report errors so that they can be improved in future releases.

The King's Quest saga is a milestone for graphic adventures and deserves to be played and rediscovered, especially in this version.

by **Giampaolo Moraschi**







SPELUNKY

Year: 2004
Editor/Developer: PaulKo64
Genre: Platform
Platform: Commodore 64
Website: https://paulko64.itch.io/spelunky64

The critically acclaimed indie platformer Spelunky has arrived on the Commodore 64.

The main developer of the project is Paul Koller, who worked on the title together with Mikkel Hastrup, who was responsible for the audio. The game began development in 2017 and slowly made its way straight to our joysticks.



The procedural map generation has been retained, and most of the features of the original version are present. We can crack our whip, throw sticks of dynamite, climb, rescue those lost in the caves, avoid monsters, and dodge traps.

Sure, it's more stripped down, but the attention to detail in the animations and sprites is maniacal. Koller has packed everything into 64kb, and the





OUR FINAL SCORE

» Gameplay 90%
Gigantic maps, always different and a ton of things to do, see, jump and hit. A title developed with great passion.

» Longevity 90%
A game that requires a lot of practice but is very rewarding. Never repetitive.



work is superb.

We can load it onto our C64s to see it run at its best or play it in emulation on VICE or similar.

The original Spelunky was created in 2008 for Windows by Derek Yu and proved to be a winning title due to its gameplay mechanics and ease of learning. In 2009, it was ported to Xbox Live Arcade and later converted to PS3, PS Vita, PS4, and Nintendo Switch. In 2020, due to its success, the sequel Spelunky 2 was released for all new platforms.

We can be satisfied with the latest titles developed for our retro platforms. Once again, we are faced with a title that is fun to play, beautiful to look at, and innovative thanks to its procedural map generation system.

The only downside is the difficulty. Spelunky is a game that requires thought and is not a simple platformer. It can be quite punishing if taken lightly and requires logic and reasoning.

Hurry up and buy this title on the developer's website for only $2.99. It's worth it.

by **Roberto Del Mar Pirazzini**





NEW GAME

DARK KNIGHT
SHADOW OF MADNESS

Year: 2024
Editor/Developer: Mananuk
Genre: Platform/Action
Platform: ZX Spectrum
Website: https://mananuk.itch.io/dark-knight-zxspectrum



"He's the bat man, he's Batman... he wraps himself in his cape, it's Batman..."

I bet you sang it, right? Thinking about your childhood or maybe Cristina D'Avena (if you're over 40, I know you thought of her). Anyway, I've got your attention and your thoughts to tell you about a new game for our ZX dedicated to the Caped Crusader.

Developer Mananuk has created a fun 4-level action game that pleasantly surprised us.

A terrorist group has released a terrible deadly virus on Gotham City, and what's more, the scientists, the only ones capable of finding the antidote, have been kidnapped.

And when all this happens, who are you going to call? Not the Ghostbusters, because they're a bit old and retired, but he's always on the hunt and ready to defend the city of Gotham.

The game is divided into four different levels. In the first level, we have to jump from one platform to another, freeing the kidnapped scientists. Obviously, we have to avoid dying a painful death due to various traps and enemies.

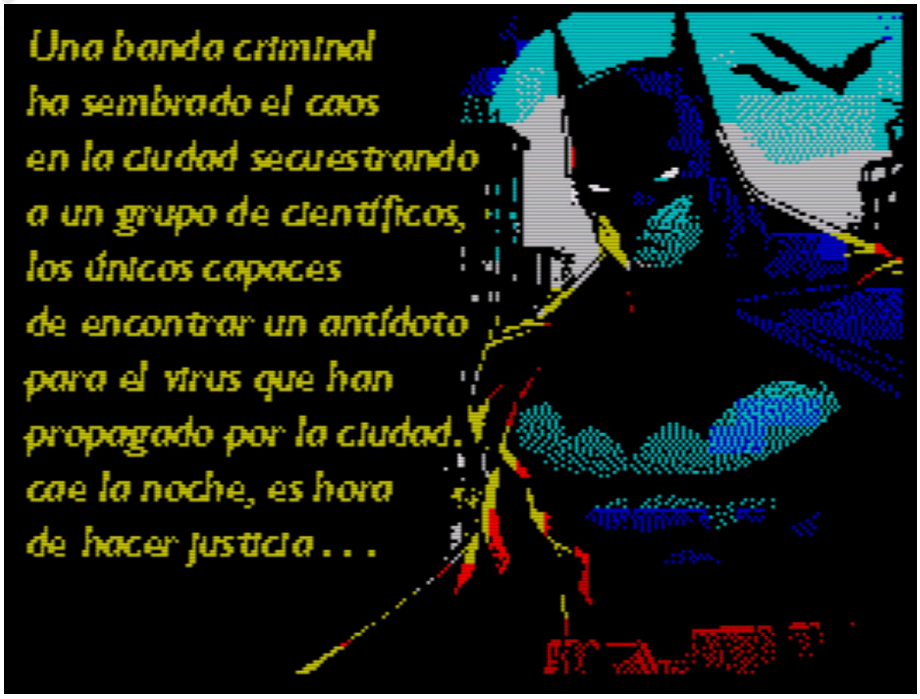
The second level is set in the streets of Gotham and is a fun driving game with a top-down view. In the third level, we'll be hunting for a virus sample to take to the lab. The fourth level is the final "rendezvous," complete with a boss battle.

But only four levels? Yes, and thank goodness! Because they're definitely tough to tackle and require practice and numerous games before they can be completed.



Our masked hero can perform several special moves: double jump, throw his batarang, and climb walls. They seem trivial, but they can all be performed with a series of well-





OUR FINAL SCORE

» Gameplay 80%

The controls seem easy to learn but require manual dexterity. The 4 levels are really well done and fun.

» Longevity 80%

Fun is guaranteed, and if you've purchased the game, the expansion offers an even greater challenge.

executed stick movements.

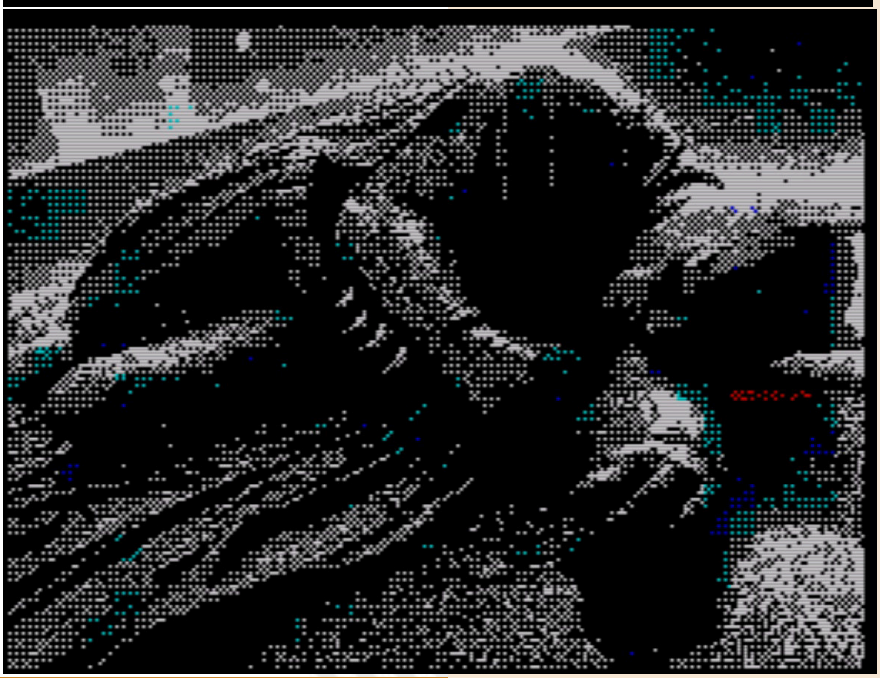
The game is available in a free digital version, but if you donate a few pennies, you'll be sent the expansion with Robin, a new vehicle, and a higher difficulty level.

Dark Knight is a good title. It's one of those games that you have to play with commitment and appreciate for its quality and ideas. It may not be the best gaming experience on ZX, but it's worth trying, especially if you're a fan of the character created by Bob Kane for DC Comics.

by **Giampaolo Moraschi**









NEW GAME

HAYATO'S JOURNEY

Year: 2024
Editor/Developer: Master Linkuei
Genre: Action/Platform
Platform: Sega Megadrive
Website: https://master-linkuei.itch.io/hayatos-journey

Hayato's Journey is a clear homage to a genre much loved on Sega 8/16-bit platforms, that of Japanese-style platform/action games.

It is an unofficial homage to Kenseiden, a cult classic for the Sega Master System, but it has little in common with Kenseiden other than the protagonist. Instead, it owes much to the Shinobi saga, with its increasing level structure, prisoner rescue, and final boss.

Hayato, the protagonist, is the descendant of the Great Hayato, a legendary member of a family of Dragons who fought against the evil Warlock centuries ago.

The evil sorcerer Yonensai, seeking revenge, decides to end the Hayato clan's lineage by killing and kidnapping the Dragon family.

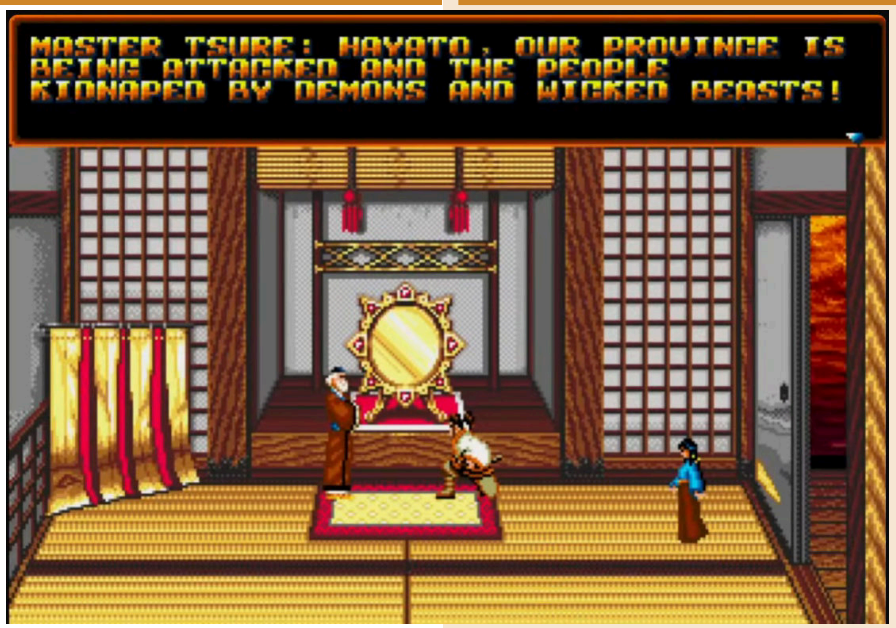
The protagonist, equipped with the sacred sword of the Dragon King, sets out in search of the evil sorcerer, traversing numerous scenarios filled with evil creatures, spirits, monsters, and malevolent entities.

The title is developed with the versatile Scorpion Engine and is the first title for Sega Megadrive completed with this engine.

It borrows some sprites and backgrounds from various video games of the past and develops into a classic platform game with action elements: destroy the enemy, recover power-ups, defeat the boss... and so on.

For variety, some levels will see us on horseback, on a boat, and on a wooden board. It's a nice way to refresh the action, which could often seem monotonous.

As I said, it's a tribute to Kenseiden, a title that mixed action elements



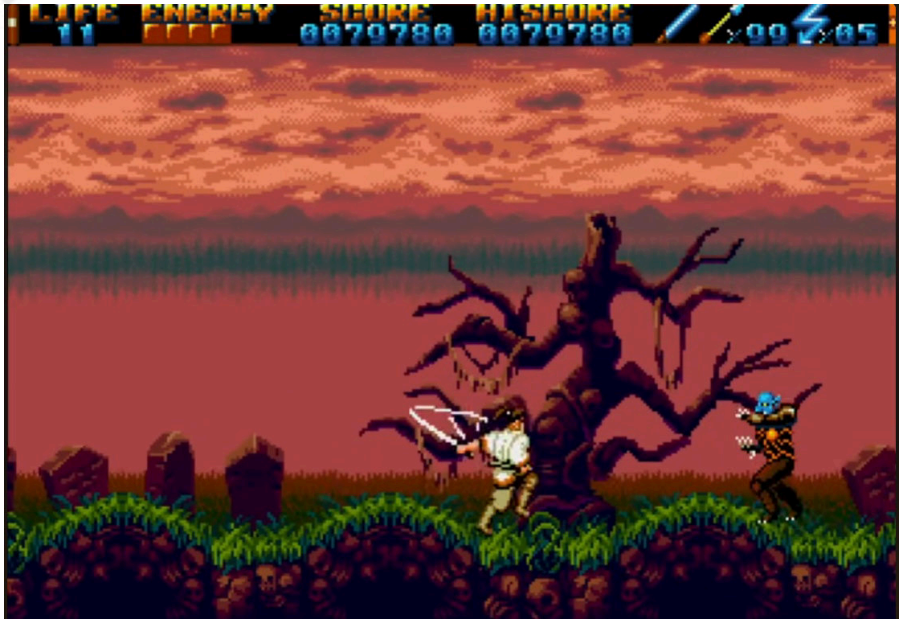
with exploration, where we could upgrade the character in order to continue.

In Hayato's Journey, all this has been removed, and the action is reminiscent of Shinobi and its sequels. A choice that may turn up the noses of fans of the original title for the Sega Master System.



From a technical point of view, it's a





OUR FINAL SCORE

» Gameplay 75%
Lots of action and various power-ups, but a strange choice of level design and an incorrect balance of power-ups and final bosses.

» Longevity 70%
Without power-ups, it's a difficult game... with power-ups maxed out, you'll finish it in no time.

peculiar title. We have some interesting elements such as different backgrounds with parallax effects, excellent use of colors and, on the other hand, the choice to often use elements from other video games.

This is a clever and useful choice, but in reality it creates more confusion and does not harmonize the aesthetic aspect.

The sound, on the other hand, features well-made sampled voice effects that are rare on the Megadrive in terms of quality and clarity.

The soundtracks are also interesting. It is a varied game with many levels, but it suffers in terms of playability. Without power-ups and the correct use of optional weapons or smart bombs, it is difficult to progress. Paradoxically, with the upgraded sword, bow, and the aforementioned smart bomb, the game becomes rather simple. I would have balanced it better.

I would also have added more power-ups to make the action more varied.

In conclusion, it is a good title that deserves to be tried and enjoyed if you love classics like Shinobi.

by **Giampaolo Moraschi**







NEW GAME

SUPER DASSALO LAND

Year: 2024
Editor/Developer: Gumpy Fuction
Genre: Platform
Platform: Game Boy color
Website: https://gumpyfunction.itch.io/super-dassalo-land?download

Tommy Dassalo is a real-life stand-up comedian. This quirky, irreverent Australian character is the star of the GBC platform game I'm presenting today.

Tommy is ready for the cultural event of the millennium: the recording of his stand-up comedy special "SCAM ARTIST."

But just as everything is about to start, the terrible (?) ghost Jiko appears and traps our bespectacled hero between the pages of his joke book... along with his dog Kewpie.

Here, Kewpie will be Dassalo's mentor. A very peculiar mentor: he talks and is very irritable, vulgar, and cannot stand his master's stupidity.

We players will obviously be Dassalo's human helpers through three worlds full of bizarre monsters, traps, jumps, obscenities, and madness, and we will be equipped with a super whip microphone and various jumping abilities.

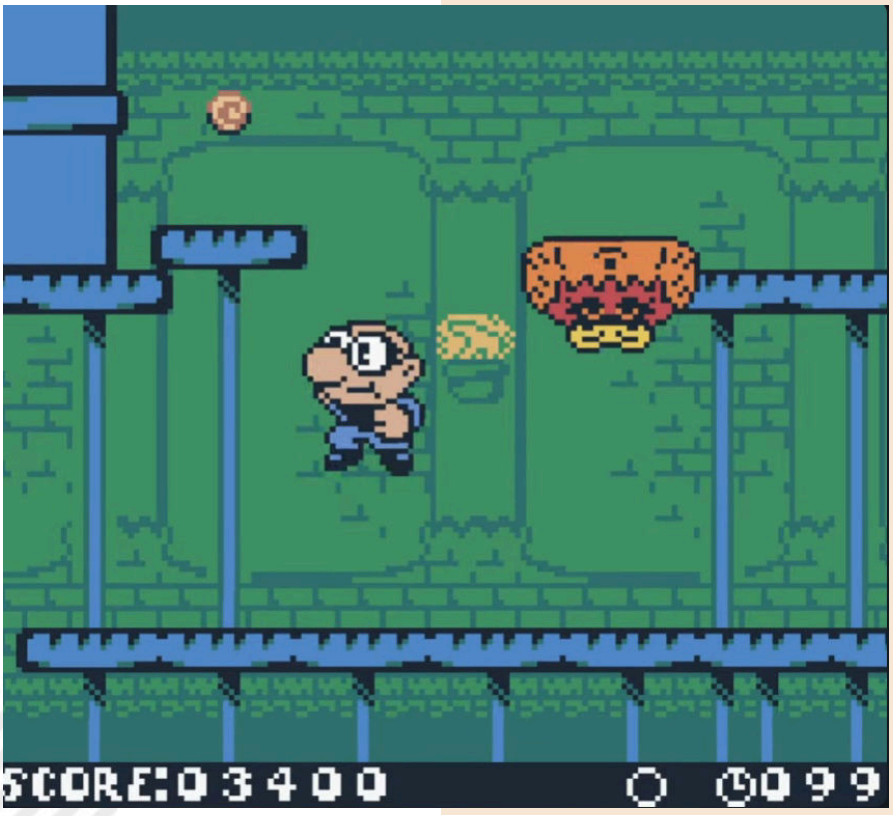
It's a platform game full of vulgar humor that may not be suitable for younger children, but it had me in stitches with the follies of its protagonist/inspirer.

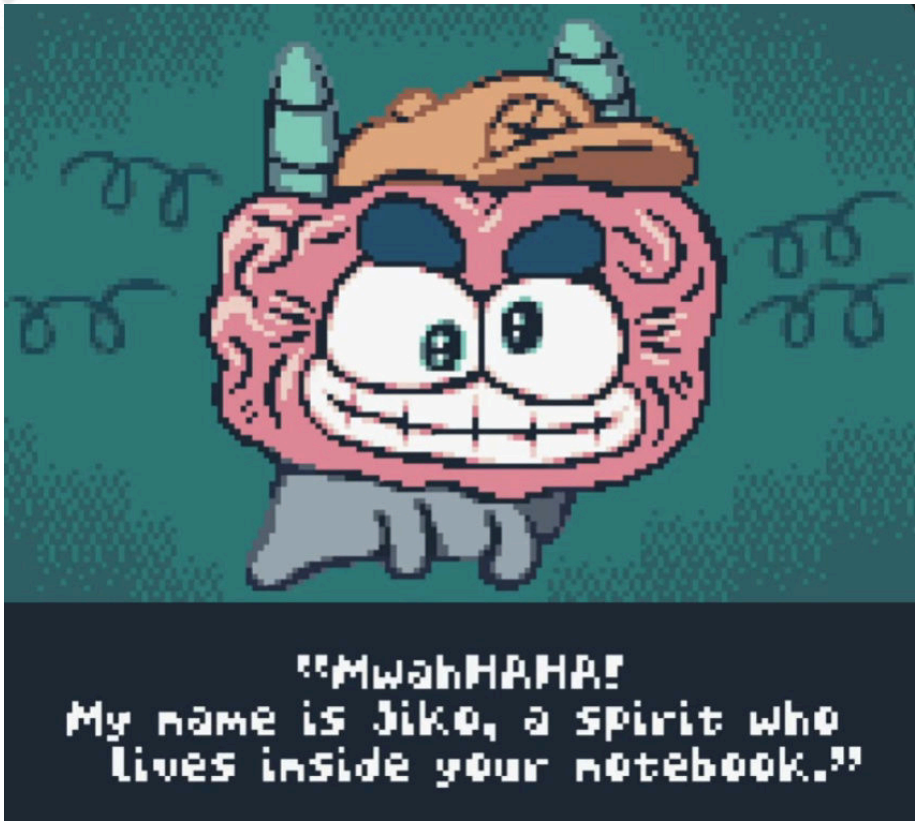
Nothing new for this kind of game on Game Boy, but it's a free title that you can play for a while.

The real flaw of the game is its terrifying simplicity. Reaching the third world takes no time at all, and you're left with a bitter taste in your mouth.

Yes, because the protagonist is the least heroic character you can imagine. Chubby, bald, and wearing two large







round glasses, he is constantly harassed by his dog, who tries not to lose his patience while explaining what our weird hero has to do.

The game was supervised by the real Tommy Dassalo and therefore suffers from his comedy, which borders on the inappropriate.

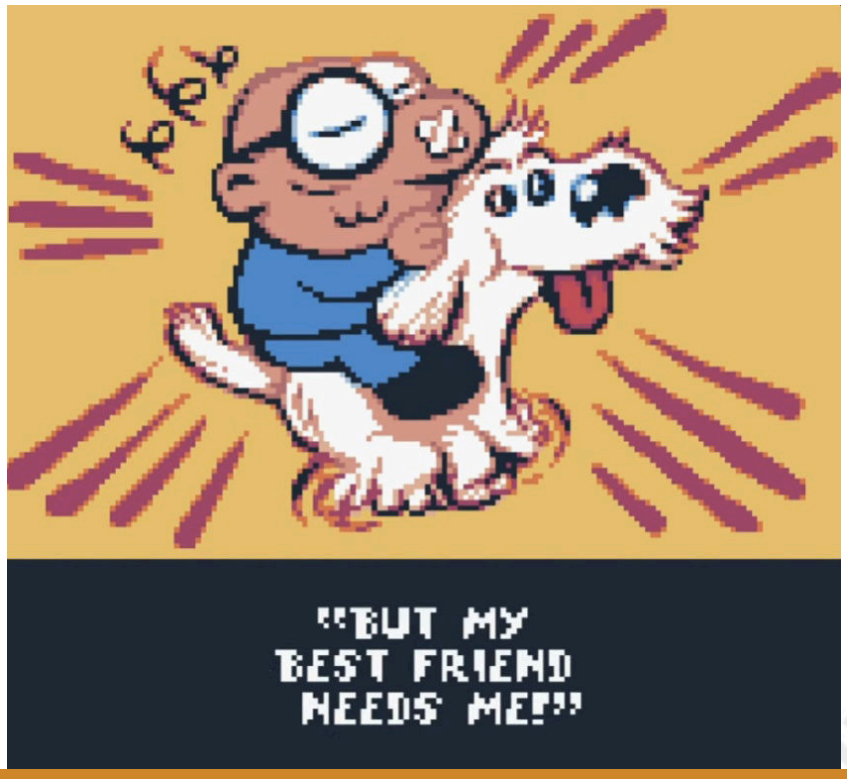
Technically, it's pleasant but never reaches peaks of excellence. I would have worked a little harder on some of the backgrounds, which risk being confused with platforms or blind jumps.

Easy to learn and too, too, too easy.

Consolation? It's free and you can even play it in your browser.

P.S. The final battle to the sound of poop is devastating!

by **Marta Rossmann**

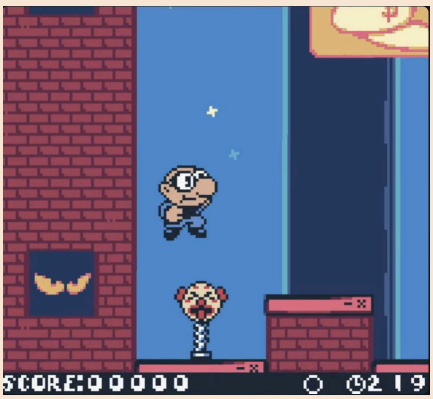


OUR FINAL SCORE

» Gameplay 70%
A platform game with everything in its place, but rather similar to a thousand others.

» Longevity 50%
The crazy humor struck me, but the game is too easy to finish.





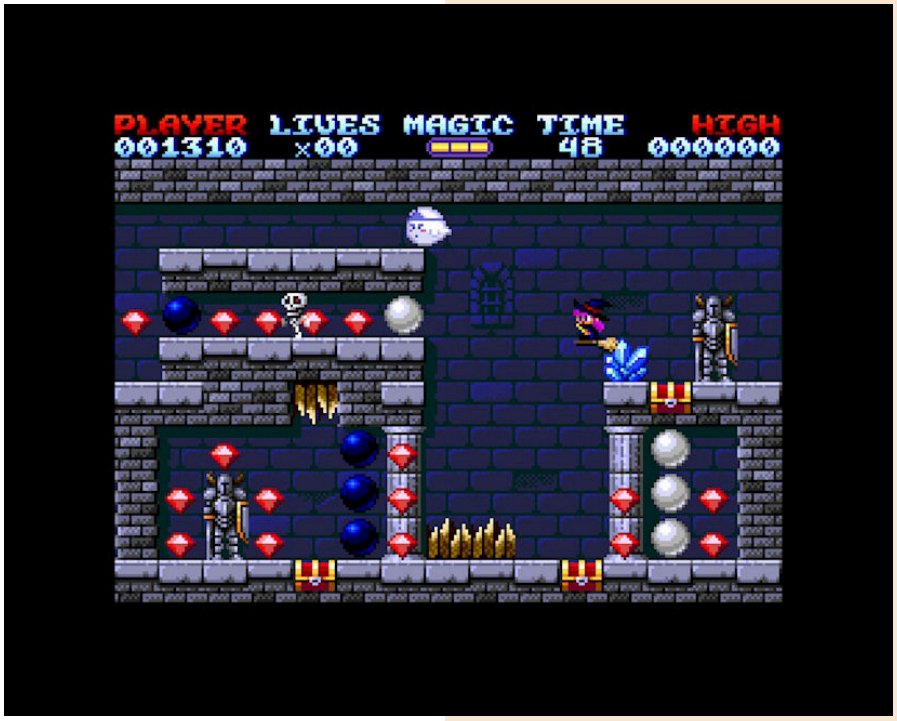




NEW GAME

BLACK & WHITE

Year: 2024
Editor/Developer: Pat Morita Team
Genre: Platform
Platform: ZX Spectrum NEXT o clone
Website: https://patmorita.itch.io/black-and-white-spectrum-next



The winning formula for a good game? Simplicity. Few frills and few things to know can make the title perfect, and this is the case with Black & White. A game that is simple in its mechanics and perfect in its entertainment value. A classic platform game that puts us in the shoes of the twin witches Candel and Alice.

These two mischievous girls have the thankless task of restoring order to the kingdom of Kalela, and they must do so by recovering the diamonds scattered across six levels to explore: the High Mountains, the Dark Forest, Coco Beach, the Black Castle, the Caves, and the Aztec Ruins.

Unfortunately, a mishap has caused them to merge into a single entity. What does this mean? It means that the two little witches are fused together and you need to use both of them to collect all the diamonds scattered throughout the levels.

The 'colors' switch when one of them runs out of magical 'mana'. This is an interesting aspect that determines how the level is completed. It's a curious but very intriguing system, which gives the player a little extra spice to tackle the game world.

It's worth remembering that our heroines can't jump, only move around the level, and that their wands can only stun their strange opponents.

See? All the mechanics are there to make the adventure exciting! Needless to say, the levels offer a different approach to collecting precious gems, there are several "kawaii" but deadly enemies, and to help us, valuable items and extra time have been added. That's right... time! Yes, because the timer to clear the level is constantly decreasing, and you'll need to hurry. I followed the development of the title for several months and was not





disappointed. I truly believe that game development on ZX Spectrum NEXT is a dimensional gateway, a link between the old titles of the 80s and the new wave of retro-style games.

The graphics are wonderful. Colorful, perfectly animated, and with a lot of variety in the landscapes and sprites. The sound is perfect, with beautiful background music and perfect sound effects.

Everything is done with one button and the joystick, and the game is a joy to play with its two difficulty levels (normal and special).

The title was created for ZX Spectrum Next or one of its clones (such as N-GO), but it works great on FPGA devices with Next cores or in emulation (the CSPect emulator for Windows is recommended).

It costs only $5.99 and is well worth it (and even more), and a future 'physical' release will certainly be worth keeping an eye on.

As far as I'm concerned, this title (along with Alien Neoplasma 1 and 2) is one of the killer applications for the NEXT.

by **Carlo Nithaiah Del Mar Pirazzini**

OUR FINAL SCORE

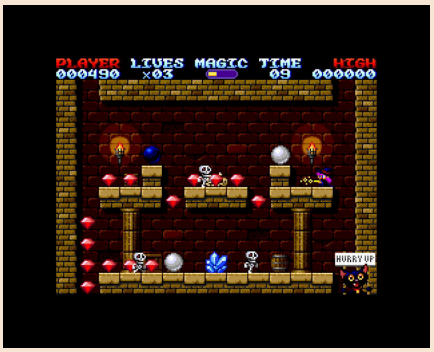
» Gameplay 95%

The gameplay is simple but with intriguing mechanics in switching between the witches and approaching the levels.

» Longevity 95%

There are 6 worlds with 60 levels to explore and two different difficulty levels. Awesome!











NEW GAME

DATA MAN

Year: 2024
Sviluppatore: Broke Studio/ Darkbits
Genre: Shoot em up
Platform: Nintendo NES, Windows, macOS
Website: https://darkbitsgames.itch.io/data-man



Virus alert! Virus alert!

Master and his henchmen have invaded the system and won't stop until every piece of data is corrupted.

That's where the Data Men come in, miniaturized heroes tasked with cleaning up the PCBs from the terrible parasites that want to eat them.

The goal is simple: blast every single insectoid monster that tries to devour the CPUs of our beloved boards.

Our Data Men are equipped with laser guns and grenades. The gun has a reload time and can be upgraded, and the grenades are limited in number, but it is possible to recover more.

The game is a nice top-down shooter reminiscent of Smash TV but set on PCBs of all types, shapes, and colors. Destroying these little monsters thrilled us, and we must mention this title among the best games for the Nintendo 8-bit of 2024.

There are 20 game levels and 4 boss fights with increasing difficulty. Each level can be completed after three waves of monsters that pop up from everywhere.

It is possible to play in pairs for incredible gameplay.

This title was created for three different platforms. By purchasing the digital ROM for NES, you will also have access to installers for Windows and Mac.

The guys at Broke Studio have also created a nice cartridge to insert into our Nintendo consoles, which also includes a nice manual.



Data Man is a well-made game that combines minimalist but very pleasant technical aspects with captivating gameplay that makes you want to see more and more.





OUR FINAL SCORE

» Gameplay 95%
Simple and with well-designed gameplay.

» Longevity 95%
Beautiful in single player but deadly in double player! Levels with increasing difficulty and an additional difficulty level to discover.

The difficulty is well balanced, although it is not a title to be taken lightly: a lot of manual skill is required to tackle the numerous game levels.

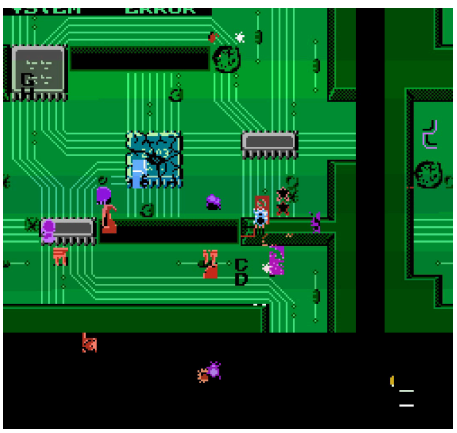
The possibility of never dying but being slowed down is nice, and the PCB destruction sequence complete with system crash is beautiful.

If you own a physical NES, buy this cartridge right away. If you have an Everdrive or want to play it in emulation, what are you waiting for? Buy the digital edition!

A masterpiece!

by **Carlo Nithaiah Del Mar Pirazzini**







NEW GAME

RUFF IN TROUBLE

Year:1992/2024
Editor/Developer: Shanti77
Genre: Platform
Platform: Atari 8bit
Website: https://forums.atariage.com/topic/374485-ruff-in-trouble/

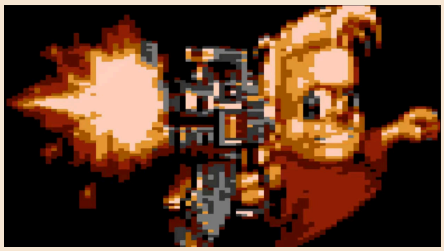
The ABBUC competition 2024 dedicated to Atari 8-bit machines has just ended. An impressive level of titles in the competition, all of the highest quality.



The winner was Ruff in Trouble, created by Shanti77. It is a platform game that pays homage to the beautiful Ruff 'n Tumble on Amiga, one of the last titles for the Commodore platform in the past era.

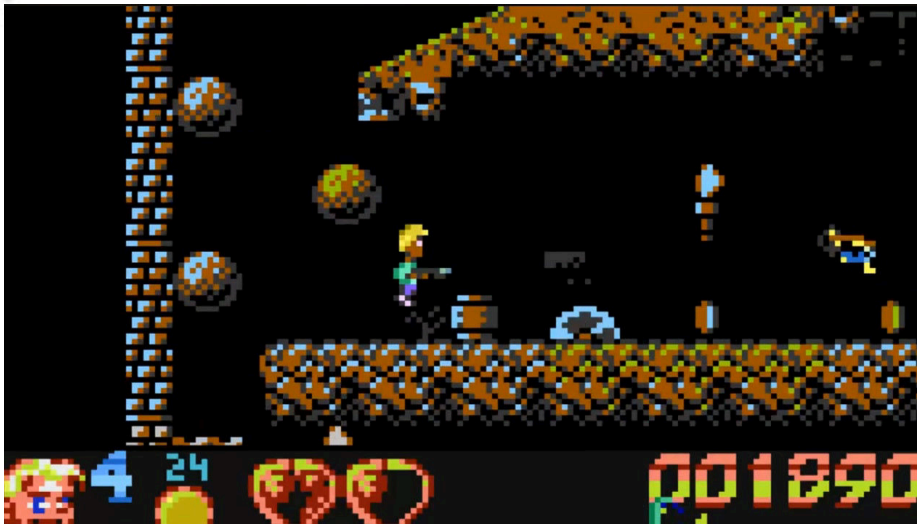
The game sees Ruff trying to escape unscathed from an inhospitable alien world full of dangers. To exit the level, you must collect spheres scattered around the perimeter of the map. Once collected, the gate will open and you will access the next level.

The game requires an Atari with 128 KB of RAM to run, supports dual-button joysticks, and runs smoothly in Pal mode (it is compatible with NTSC but with some slowdown).



Shanti's technical work is excellent, showcasing detailed, colorful, and carefully animated graphics.





OUR FINAL SCORE

» Gameplay 92%
Large, detailed levels combined with good use of game controls and a high level of challenge right from the start.

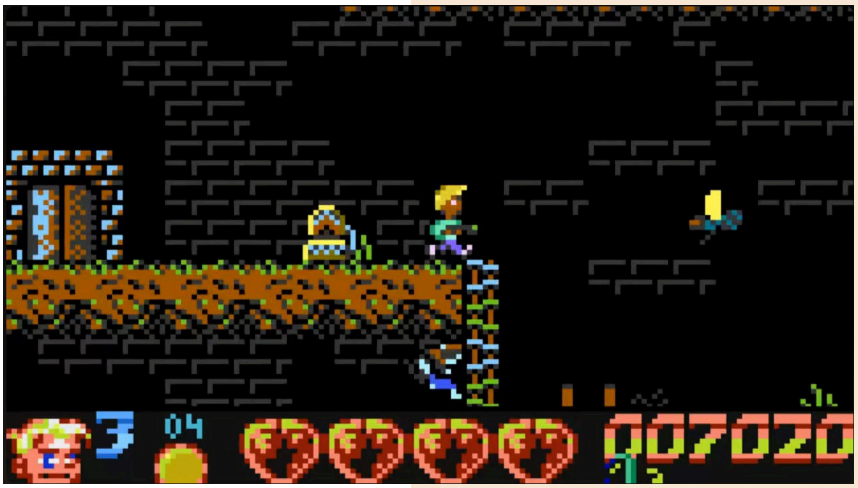
» Longevity 90%
Requires practice and discipline, but it's a lot of fun!

The level maps are fun to explore, and you can lose yourself in appreciating the chosen details.

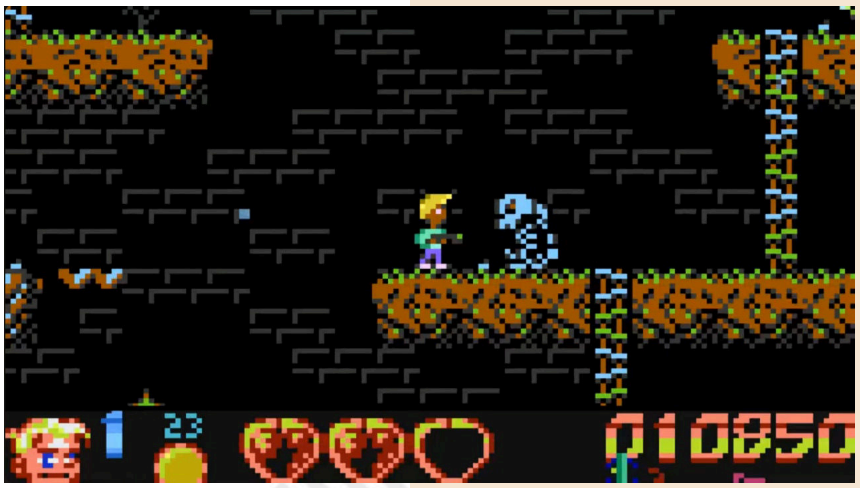
Apart from the incredible graphics, the difficulty level is actually as challenging as Ruff 'n Tumble was on the Amiga. There are quite a few traps to deal with, and enemies must be approached intelligently to avoid premature death. Another strong point is the way you approach the level. It's a multidirectional title that allows you to carefully explore every possible nook and cranny.

The challenge to win this edition of ABBUC was definitely high, but Shanti has given us a really well-made title that deserves to be played and supported.

by **Giampaolo Moraschi**









NEW GAME

SUPER MARIO BROS MINI

Year: 2024
Sviluppatore: Mico27
Genre: Platform
Platform: Game Boy Color
Website: https://mico27.itch.io/super-mario-bros-mini

Since 1985, we have known everything about Super Mario Bros, the first chapter in the "Super" saga of Nintendo's famous Italian-Japanese plumber.

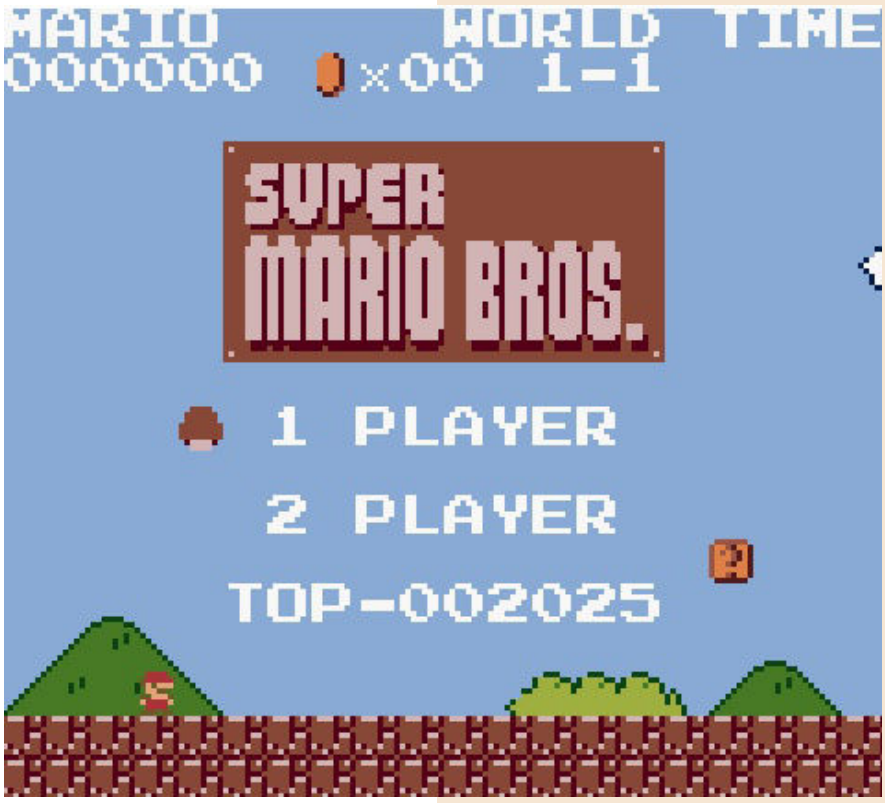
We have seen the mustachioed hero run through colorful worlds to collect coins and save Princess Peach from the clutches of Bowser, the arch-enemy of the entire saga.

It was an incredible success for Nintendo and the real killer app for the NES when it launched in the United States and Europe (the NES had already been available in Japan for two years). Since the legendary first installment, we have seen sequels on all Nintendo home consoles, all splendid examples of gameplay.

But the charm of the first Super Mario Bros. is timeless. We've seen Speed Runners try to complete the game as quickly as possible, other players discover all its incredible secrets... we've even seen the beautiful conversion for Commodore 64 and the demo for Intellivision (both "blacked out" by Nintendo).

All that was missing was this version for Game Boy Color, created by Mico27 with GBStudio.

The entire original game is contained within this tiny pixelated file, and that's not all! Mico27 has added a new game mode with 8 additional worlds, new graphic themes, new







OUR FINAL SCORE

» Gameplay 95%

Super Mario Bros has perfect gameplay mechanics even in 2024 and even on GB Color. All this is enriched by two additional modes.

» Longevity 95%

It includes the entire original game (including warp zones), 8 new worlds, and the random version. What more could you want?





levels, a new power-up, some "friends," lots of new bosses, and much more.

As if that weren't enough, he has added a 'Random' mode that allows you to tackle the adventure in a truly alternative way.

All this can be accessed by pressing B in the main menu and selecting what you want to tackle.

GBStudio is a powerful and well-developed build that allows professionals to create titles, demos, or experiments in a simple way but with great attention to the final quality.

The work here is truly remarkable. Everything from the original game is there. Perfect sound (within the limits of the GBC) and meticulously crafted "kawaii" mini graphics.

This is one of those products to consider, test, and try before Nintendo wipes it off the face of the earth (let's hope not!!!).

Finally, for those who are curious, I would like to point out that the developer has made the source codes available on their GitHub page at the following address:

https://github.com/Mico27/SuperMarioBrosMini.

A great title for GBC.

by **Carlo Nithaiah Del Mar Pirazzini**



NEW GAME

SUPER MARIO ECLIPSE

Year: 2024
Sviluppatore: Nintendo, Gamebanana
Genre: Platform game
Platform: Nintendo GameCube
Website: https://gamebanana.com/mods/536309

Sunshine has always been one of the games that fascinated me most on GameCube. Because of its development history, its content, and its extremely difficult task of matching the success of Mario 64.

So playing this hack was a magical experience.

To be honest, I didn't know what to expect when I downloaded it. I first watched the trailer and thought it was a completely new campaign, but it was a pleasant surprise to discover that the game is a reimagining of the classic with some interesting new elements and ideas that should have been implemented in the original.

The changes in movement are the biggest new feature, with the addition of the long jump and ground pound (straight from 3D World). These two new moves really help a lot in the game and in the missions. It's satisfying to chain these moves together with the rest of the set, which was already great.

Another thing that helps the flow of the game immensely is that it doesn't kick you out of the levels once you complete a mission. Wow! In the original game, it kicked you out even when you collected all 100 Shine coins! Now, depending on what happens in the level, it may take you



back to the mission selection screen or simply let you continue wandering around and looking for more Shine to collect. There is also a quick restore option if you want to change missions in the level you are in, another godsend. Blue coins are also easier to collect, another controversial aspect in the original Sunshine. The new levels, which were discarded in the original title, seem to belong in the game with everything about the way they were









OUR FINAL SCORE

» Gameplay 92%

A title that greatly improves on the previous and much-maligned Sunshine, adding a ton of extra features, all of which are great fun.

» Longevity 85%

The frame rate issue in some places is really annoying, but the title promises hours and hours of gameplay and collections to unlock.

designed. They fit well with the overall level design and gameplay dynamics. Four levels with 8 missions and lots of things to collect, excellent work.

Eclipse turned out to be full of things to do, collect, new challenges, incredible secret endings, and a frightening amount of post-game content.

Perhaps the sore point of Eclipse is the sometimes shaky technical implementation. In some levels, the frame rate drops in a deadly and annoying way. It will surely be improved with the upcoming releases that the developers have promised on their page. All in all, the rest holds up and the game flows despite this problem.

Super Mario Eclipse is one of the best hacks I've ever played, perhaps the best. It is the work of a team that has understood the philosophy of the original game 100% and captured its essence in an (almost) flawless way with all the brand new additional content.

by **Marta Rossmann**



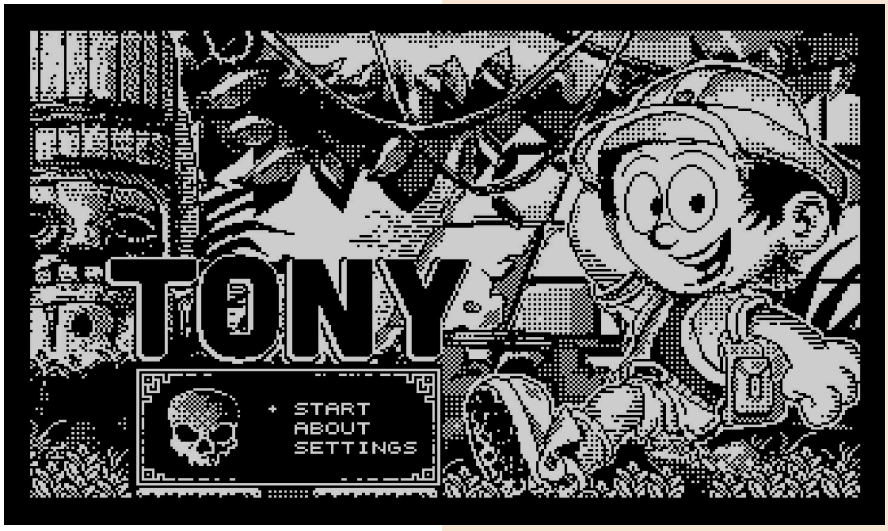


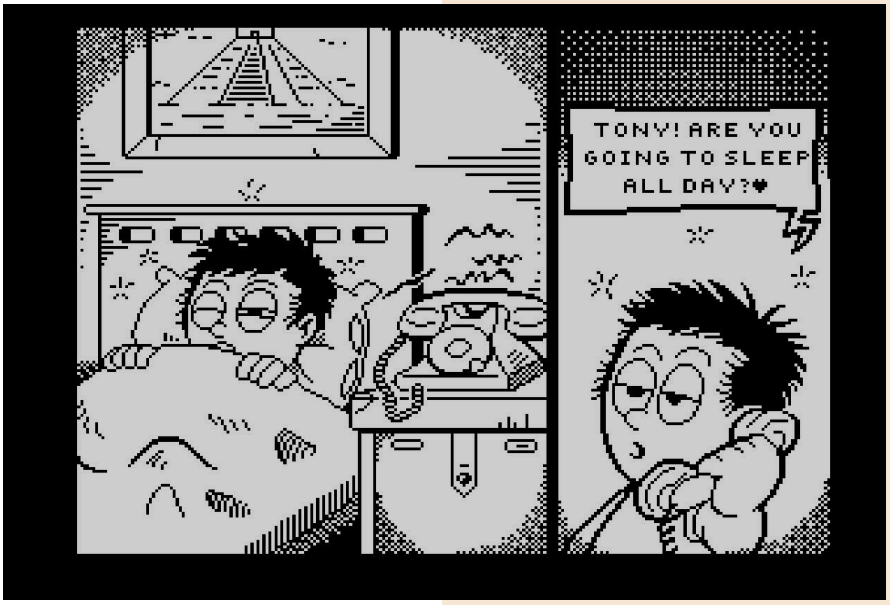


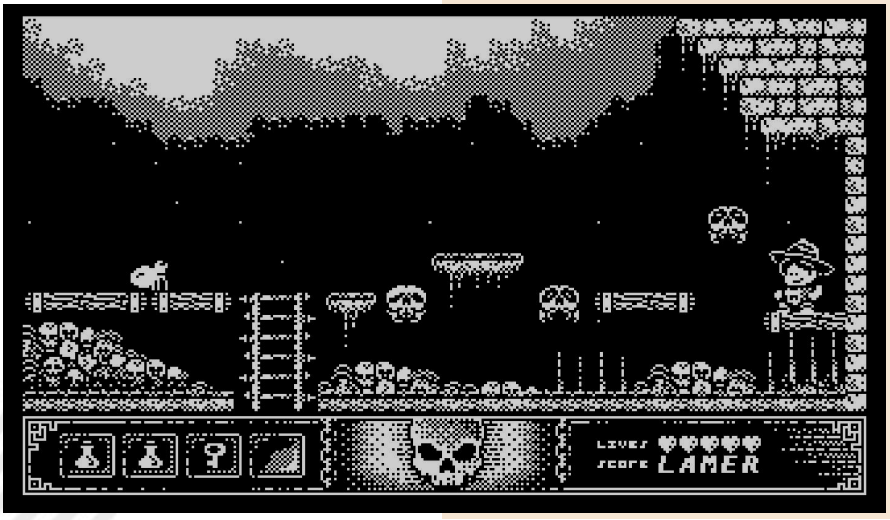
NEW GAME

TONY MONTEZUMA'S GOLD

Year: 2024
Editor/Developer: Monochrome Productions
Genre: Platform
Platform: Commodore 64, Amiga, Atari 8bit
Website: https://monochrome-productions.itch.io/







An ancient legendary mask... A dense mysterious jungle... Cursed temples... A hero without fear or reproach... a joystick... and us players!

This Italian summer, we will embark on a quest to find Montezuma's mask. Not just a simple artifact, but a piece of jewelry with mysterious powers.

A beautiful old-school platformer that pays homage to the great classics of the past, especially Parker Brothers' Montezuma's Revenge.

It also pays tribute to the Polish explorer and filmmaker Tony Halik, known for his adventures across all continents.

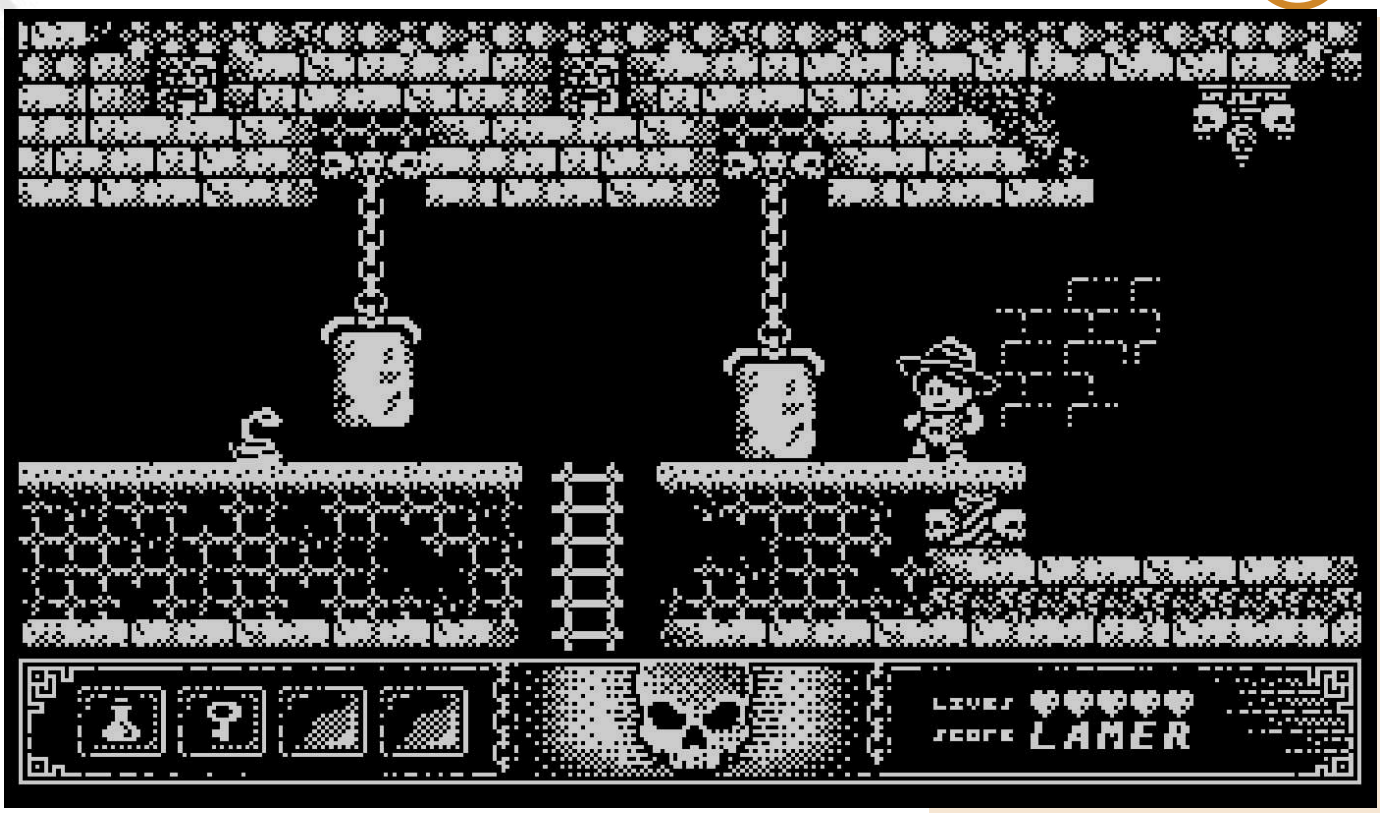
In the role of Tony, the player will traverse Montezuma's castle, full of traps and deadly creatures. It's a complex undertaking, full of pixel-perfect jumps and items to collect.

Essential for sanity and survival are the jewels scattered throughout the levels, potions that allow us to eliminate enemies, golden masks, and keys. The latter two items are essential for progressing through the level.

However, it won't be easy to carry them! The player's inventory has only four slots, so choosing what to put in them will be crucial.

Finally, the player must carefully observe the numerous and (well) hidden secret rooms, full of additional treasures. The developers have placed them in the most unexpected places.

Tony Montezuma's Gold comes in three beautiful versions, each with a



single distinguishing feature: monochrome graphics.

This is a particular stylistic choice, but a very apt one, reminiscent of the "Golden Age" of video games.

Despite the black and white, the graphics in all three versions (Amiga, Commodore, and Atari 8-bit) are very, very pleasing. They are well animated and detailed, but with slight differences in resolution and animation, of course. In all three cases, however, the game looks really good and runs like a dream.

The sound is also nice, especially on the Commodore 64 and Atari. I found the chiptune soundtrack very engaging, while I didn't like the sound effects of the Amiga version.

The comic-style graphics are also very nice, narrating the adventure of poor Tony from his bedroom to the Amazon rainforest.

Let's move on to the most playable aspect.

The game levels are well structured in all three versions and identical. It's the typical game that requires memory and that kind of 'perfect jump' gameplay that is not always easy for casual players. Of the three versions, I found the Amiga version to be the weakest. It is identical in every way but more "difficult" in terms of jumps and character dynamics, which seem more rigid than in the other two versions.

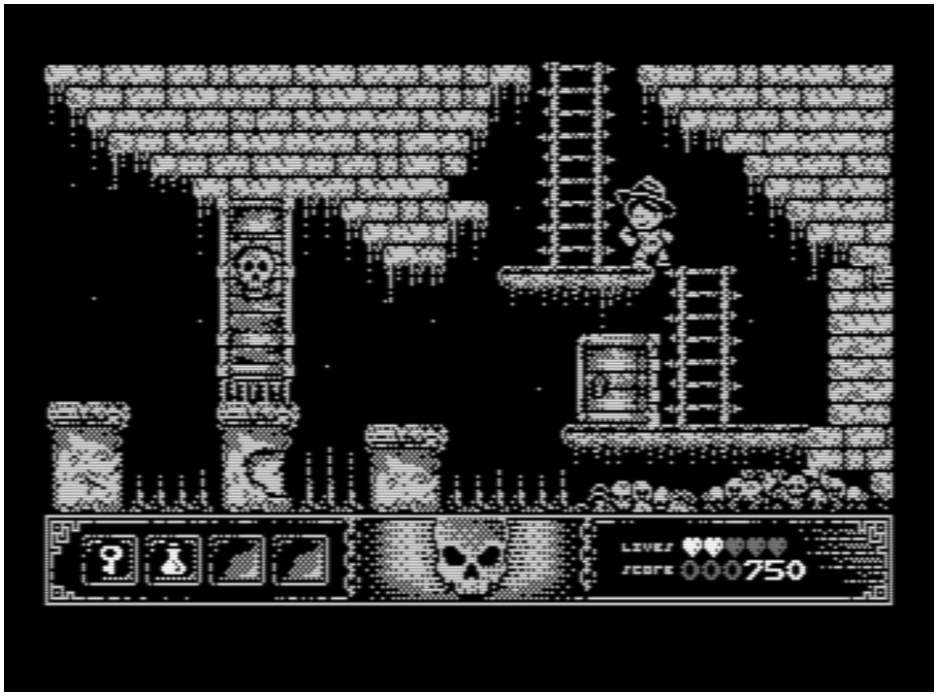
In any case, it is a tough game that requires several hours to master and a good memory.

However, it remains a great product in all three versions and certainly deserves

Fig. 1,2,3,4 - Images taken from the Amiga version. The introductory screen, a frame from the charming introduction, and two images of the game in progress. The monochrome aesthetic is splendid!

Fig. 5,6 (next page) - Images taken from the Commodore 64 version. Intro screen and a stage of the game. Same atmosphere as the Amiga version.





OUR FINAL SCORE

» Gameplay 85%

An average of the versions. The levels are well developed and the gameplay is proven and enjoyable. On the Amiga, the protagonist is less "agile" than in the other versions and some jumps are not quite perfect (this greatly limits the enjoyment).

» Longevity 90%

Difficult for "casual" players but with that nice 80s flavor. Jumps upon jumps to be performed by memorizing the level.

a passing grade and even more.

On Commodore, it is in disk format (divided into two files) and works on real hardware and in emulation.

There are also no problems on Atari in real hardware or on Altirra.

The Amiga version requires a Kickstar 1.2, 68000, 1 MB of Fast RAM, and 0.5 MB of Chip RAM. It runs on OCS/ECS/AGA and, of course, in emulation.

All three versions will have several physical releases on disk (C64 and AMIGA) and cartridge on ATARI with manuals, gadgets, and much more.

Congratulations to the development team for creating such a product in 2024. Well done!

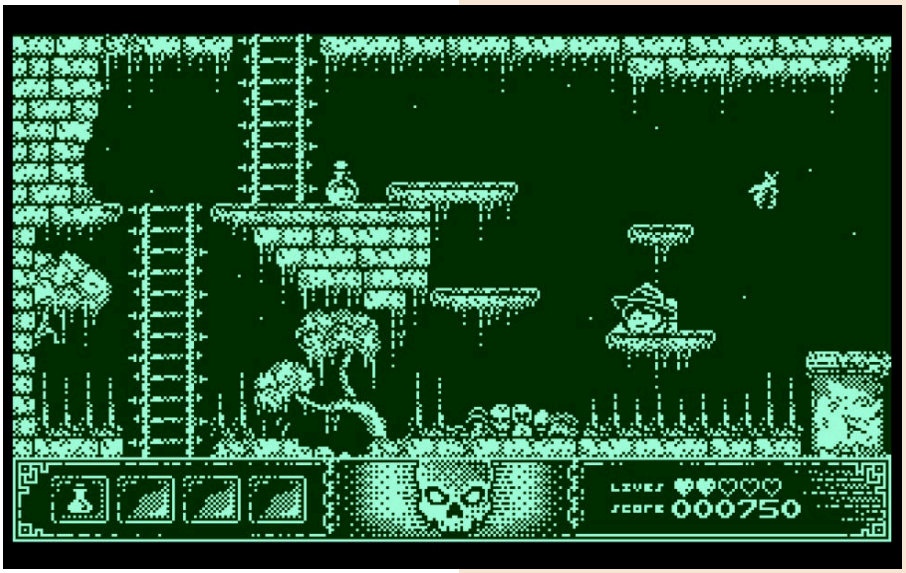
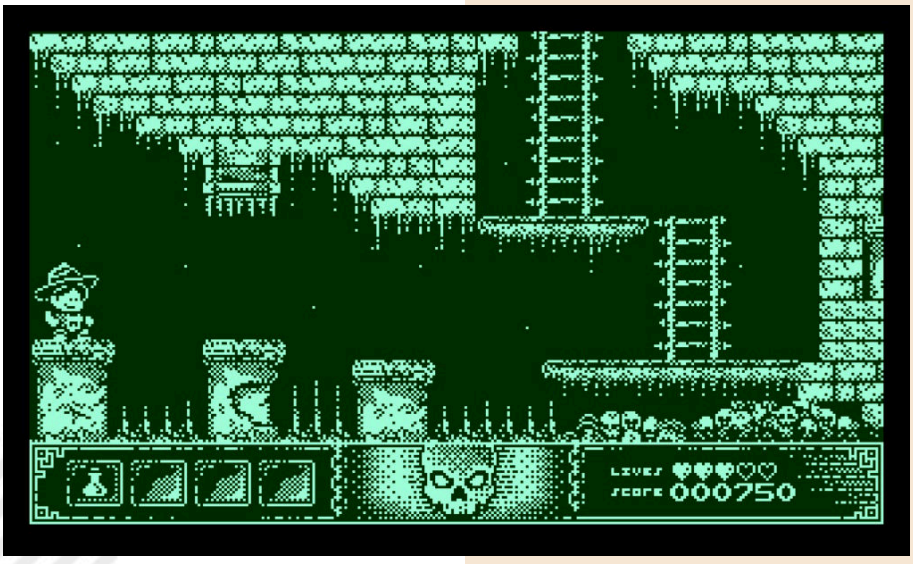


Fig. 7,8 - Images taken from the Atari 8-bit version. Once again monochromatic, but this time in shades of green.

October 2024 update.

The latest updates to the game improve fluidity and gameplay and give (for Atari8bit users) the fun Go!Tony Go!, a running game on a mining cart. A fun little game that adds a few points to the final score.

by **Carlo Nithaiah Del Mar Pirazzini**



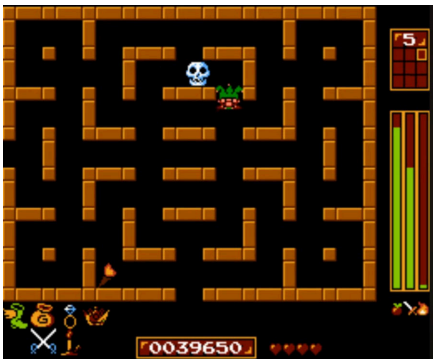
NEW GAME

JESTER

Year: 2024
Editor/Developer: MegaCat Studio
Genre: Maze game
Platform: NES/Super Nintendo
Website: https://www.collectorvision.com/games/jester.php



Jester, the King's jester. You always make everyone laugh, but this time there's little to laugh about.
Locked in a dungeon, chased by terrible monsters and with the aim of collecting all the lost Elemental objects. Difficult, right? Well, life is a joke!
This Jester is a fun game, released in 2023 and in a "physical" version in 2024 for Nintendo 8-bit and Super Nintendo.
A love letter to Le Fou du Roi, one of the games available on the Videoway cable TV system in the 1990s. An analog interactive TV system available only in Canada between 1990 and 2006. Evidently, MegaCat loved it so much that they are re-releasing it now.
It is described as "a classic maze chase game" and is reminiscent of the action mechanics of many similar titles.
The collectible items are interesting, allowing our "jester" to fight monsters, continue to see the light inside the dungeons, and recharge his energy.
The NES and Super NES versions are identical, not even the game speed changes.



As I said, the game is available in physical format with beautiful cover artwork and a well-designed, colorful manual.
However, Jester suffers from boredom. The game is really all the same, and after a while, it will lead us to turn off the console.
It's a shame because the protagonist is really funny.

by **Marta Rossmann**











OUR FINAL SCORE

» Gameplay 70%
A simple game to play.

» Longevity 60%
After the first few levels, boredom sets in more aggressively than the various monsters present.

NEW GAME

DR. DANGEROUS

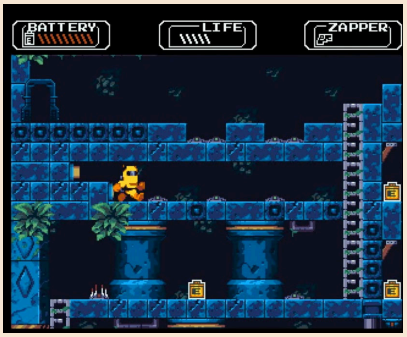
SECRETS OF THE TEMPLE OF XOL'TAN

Year: 2024
Editor/Developer: HooGames2017
Genre: Platform/Puzzle
Platform: Amiga
Website: https://hoogames2017.itch.io/dr-dangerous

Dr. Victor Dangerous is a strange guy. He discovered a secret temple in the Amazon rainforest, built his own laboratory for experiments inside it, and spends all day walking around in a Hazmat suit. In that heat!? Crazy!
Now, one evening during a storm, lightning strikes the laboratory's power supply. Everything breaks down and the automatic safety protocols are activated. The doors close and the poor doctor is violently thrown into a deep underground tunnel.
A tunnel full of dangers where our hero will face the most dangerous and bizarre creatures and recover the backup batteries to reactivate the temple's functions. Obviously, it won't be an easy task, as the creatures are highly aggressive by nature and our doctor is armed with a gun that runs out of ammunition quickly.
In short, you've guessed it, we're dealing with a classic platform game that presents us with little puzzles to solve in order to pass the various levels and doesn't make things easy for us. The gun must be reloaded by recovering ammunition, and it will be the deciding factor in overcoming the many obstacles between us and our coveted freedom.
Dr. Dangerous is a game that doesn't invent anything new. It is reminiscent of Rick Dangerous and even the very recent Tony Montezuma. Well-designed, deep levels full of traps and monsters must be completed using logic and manual dexterity.
However, this lack of innovation is not a limitation or even a problem. Dr.



Dangerous is well made and enjoyable to play, yet another demonstration that if you take care of the technical aspects, gameplay, and level design, you will get a good product even if there is zero innovation.
I enjoyed exploring the underground levels of the temple, and the degree







OUR FINAL SCORE

» Gameplay 80%

Nothing new, but what's there is well done, and the game is a good example of balance, level design, and gameplay.

» Longevity 85%

The difficulty curve is well balanced and the code system for saving levels is simple and functional.

of challenge increases gradually without ever putting the player in difficulty.

The code system for restarting from the level where you 'died' is interesting, as is the option to play with a second button for jumping.

In short, it may not be the platform game of a lifetime on Amiga, but it is a good product made with the increasingly efficient Scorpion Engine (and there are rumors of a Megadrive version).

The title runs on all Amiga 500 PALs with a 512Kb extension and kickstarter 1.3 and, of course, in emulation and on the A500 mini.

by **Carlo Nithaiah Del Mar Pirazzini**





NEW GAME

PARASOL STARS
THE STORY OF BUBBLE BOBBLE III

Year: 2024
Editor/Developer: ININ Games, Taito
Genre: Platform
Platform: Nintendo Switch, Playstation 5, Xbox Serie X/S, Playstation 4, Xbox One
Versione di prova: Nintendo Switch



It almost seems as if Parasol Stars is the "forgotten" chapter of the Bubble Bobble saga.

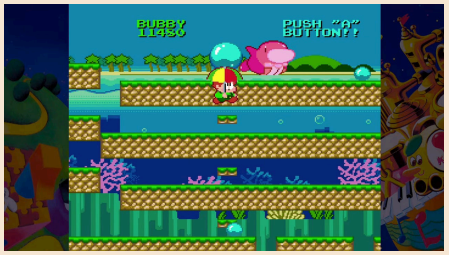
Sure, it has very little to do with the original game by the late Master Fukio Mitsuji in terms of appearance and gameplay. But the tone is on par, and boy is it as fun to play today as it was in 1991, when it was released on the Turbo Grafx16.

Now ININ Games has brought the title back to its full glory on modern consoles, and we at Retromagazine World played it on a modern Switch. It did so at a low price, only ten euros, and with a lot of options. But more importantly, it's still fun after all these years, especially when played in two-player mode.

Personally, I would have liked to see a Bubble Bobble compilation/trilogy, but it's nice to see Parasol Stars get the attention it deserves.

Although the gameplay is different from previous titles in the series (Bubble Bobble and Rainbow Islands), Parasol Stars has similar principles. You have to clear a screen full of enemies using an umbrella and various creative methods. These include picking up enemies on the tip of the umbrella and throwing them at other enemies, as well as collecting bubbles to perform terrible and bombastic special attacks.

There are some deeper strategies such as chaining together multiple enemies you've hit or collecting more fruit to increase your score, lives, and bonuses. Then, at the end of each series of levels, you face a more or less challenging boss. Incidentally, the boss fights are still a lot of fun to tackle.

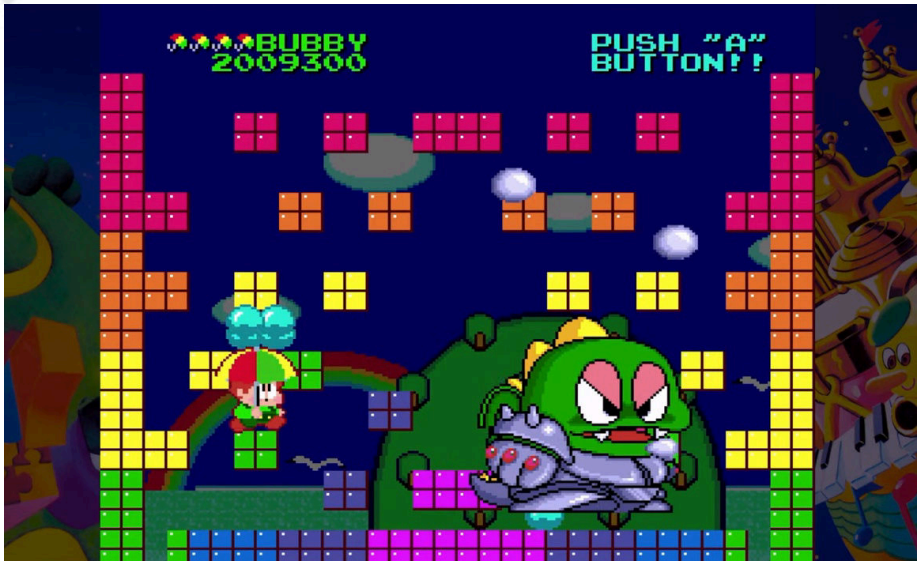




ININ Games has done a masterful job of bringing this Taito classic to its glory in the modern era. I particularly loved the Amiga and ST versions of the game, and I must admit that seeing it on Switch in portable mode took me back to the summer of 1992.

The development team has also added several interesting options: in-game rewinds, a modern save state/load state system, numerous game options, and much more. And, of course, the





OUR FINAL SCORE

» Gameplay 90%
A title that is still fun even after thirty years.

» Longevity 90%
The gradual difficulty curve makes it even more enjoyable.

beautiful two-player version that can be played locally in both cooperative and competitive modes.

Overall, Parasol Stars is still a wonderful game that is worth rediscovering if you haven't played it in a while, and it's the ideal bridge between us and our children/grandchildren. It's a way to show how simple design and good gameplay are fun and transcend age differences.

But how does it look to us thirty years later? The graphics hold up well after all this time, and the level design is bright, colorful, and easily recognizable. It's nice to see some "old" enemies between levels. They bring back memories and peach tea on distant summer afternoons.

The music is always catchy and never annoying. A cheerful tune along the lines of Bubble Bobble.

Does it stand up to comparison with the modern era? Yes, and rather well. It doesn't claim to be innovative or an absolute must, but it remains a great platform game, perhaps to be played on a gloomy day or on the train in portable mode.

by **Carlo Nithaiah Del Mar Pirazzini**





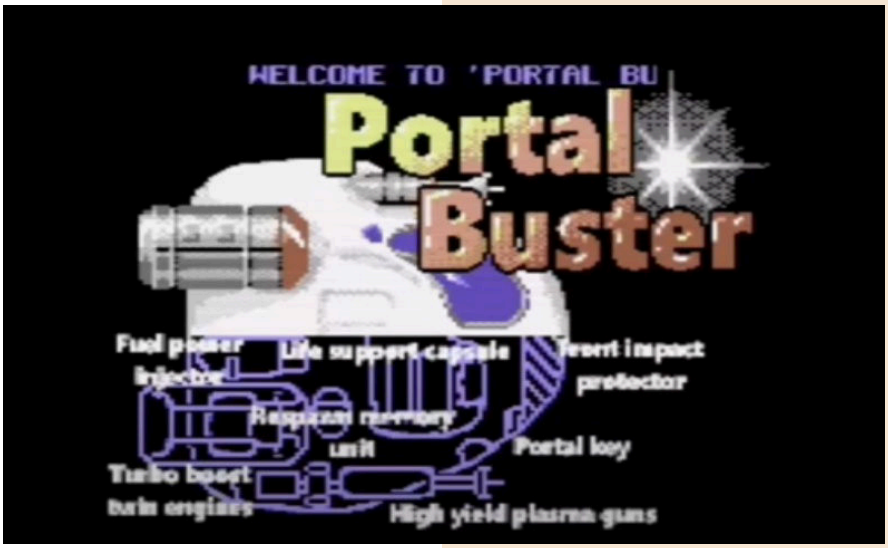


NEW GAME

PORTAL BUSTER

Year: 2024
Editor/Developer: drmortalwombat
Genre: Shoot em up
Platform: Commodore 64
Website: https://drmortalwombat.itch.io/portal-buster

Fast is the name of the game! Super fast, in fact! That's definitely the first thought that comes to mind when playing Portal Buster, a new shooter for our beloved C64.



A decidedly unexpected title that we particularly enjoyed, reminiscent of the classic Uridium but also Armalyte, and which, especially due to its aforementioned speed, puts even the most skilled players of this genre to the test.

The aim of the game is to destroy a mega enemy boss, reaching it through a vast map and overcoming countless obstacles and enemies.

If that weren't enough, the spaceship also has serious fuel problems and must refuel by collecting barrels along the screen.

The multi-scrolling style, as in Uridium, allows you to adjust the speed and slow down the pace of the game.

The developer has also added a couple of "trainers" to make the gaming experience less lethal, for which we are infinitely grateful. Indeed, 15 seconds between refueling stops is definitely not enough!

The game is PAL only and uses VSP for scrolling, but it is fully compatible with C64/128, various emulators, and TheC64 family.



It's fun, which is already a plus these days! Graphically, it's pleasing, and it has good audio thanks to the skillful use of SID.





OUR FINAL SCORE

» Gameplay 80%
A clone of Uridium with crazy speed and a good gameplay system. The trainers help you get the hang of the game style.

» Longevity 75%
Difficult and not for everyone, but really enjoyable once you get the hang of it.

It may be a bit tricky for beginners, but we assure you that you won't be putting down the joystick anytime soon, and one game that ends badly will immediately lead to another.

Highly recommended for fans of the genre and lovers of "harder, faster, stronger" titles.

by **Carlo Nithaiah Del Mar Pirazzini**









NEW GAME

BINARY BATTLE

Year: 2024
Editor/Developer: Haplo
Genre: Dungeon Crawler
Platform: Commodore 64, Plus/4, ZX Spectrum
Website: https://h4plo.itch.io/binary-battle



You've just woken up and sat down in front of your trusty Commodore 64. You grab your favorite floppy disk, but realize it's full of viruses! Damn! The only copy of the source file for the game you've been working on for the last six months...

Don't worry! Your powerful antivirus will help you clean everything up and get back to work.

Binary Battle is a brilliant new take on the dungeon crawler genre.

Inspired by GEOS and the PC game Folder Dungeon, your goal is to recover the code by navigating five challenging levels filled with errors, viruses, and corrupted data.

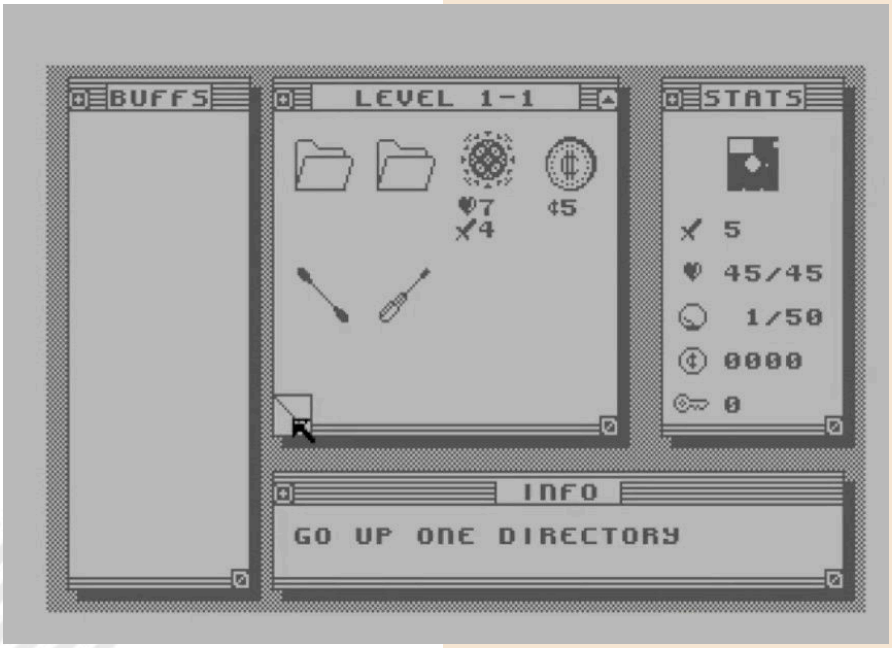
Using the mouse cursor, we move around the GUI environment to target corrupted files or folders and clean them up. If necessary, we can repair the damage, recover virtual coins (useful for upgrades), and restore the system.

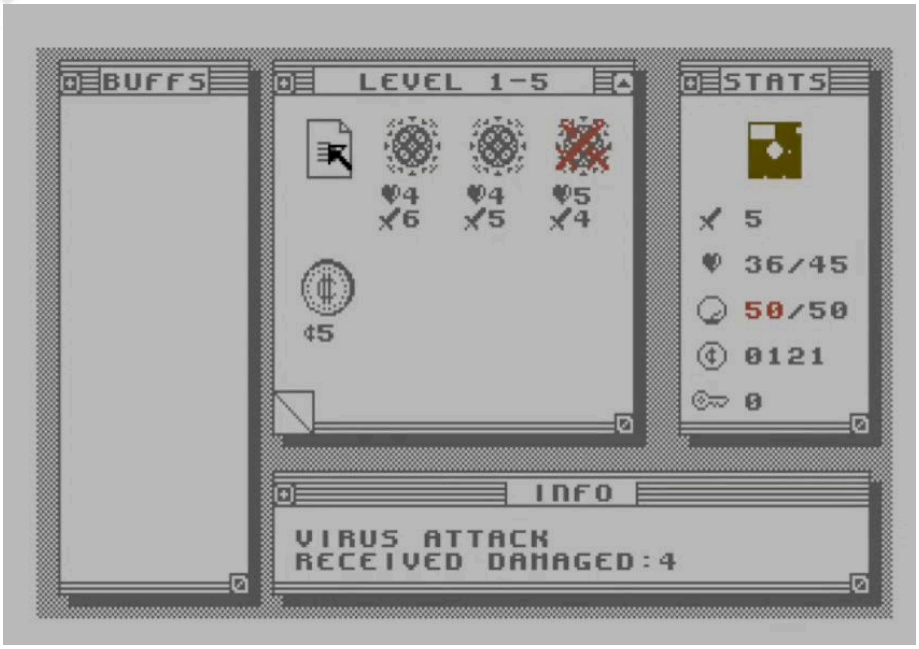
Our antivirus has limited health that decreases with each virus attack. In addition, each action generates heat, and our floppy drive may suffer damage.

A truly curious title that won us over. Appearances can be deceiving. What initially seems like just a nice graphical representation of a Geos Style GUI desktop turns out to be an immersive roguelike with several engaging strategic elements. Managing the health of your disk while eliminating viruses and navigating directories and file structures is an engaging experience, with failures often resulting in another attempt to see if you can reach your goal.



Forget the elaborate graphics, here





OUR FINAL SCORE

» Gameplay 90%

Good level of strategy combined with a unique and appealing style. Add to that the random games every time we restart and the very satisfying feeling we get every time we "clean up" our program.

» Longevity 80%

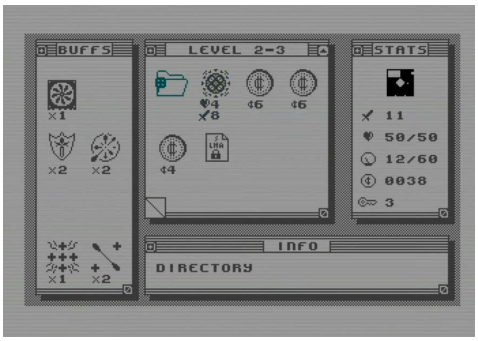
The style may discourage casual players, and perhaps 5 levels are too few.

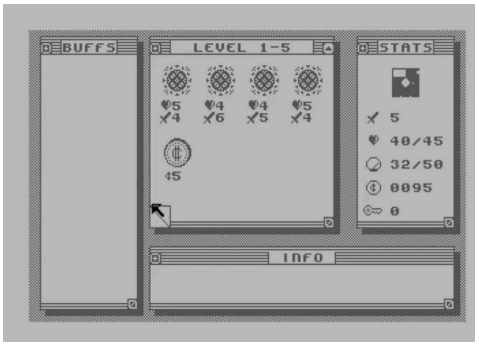
we have a true little masterpiece of gameplay (and a great soundtrack). The three available versions differ minimally from each other. The structure of the game remains the same, and if you're looking for a title that's easy to learn and engaging, Binary Battle is a unique and rewarding experience.

by **Carlo Nithaiah Del Mar Pirazzini**



Versions: Plus 4 (above) **ZX Spectrum (below)**







NEW GAME

SHIFT

Year: 2024
Editor/Developer: Haplo
Genre: Puzzle/Platform
Platform: Commodore 64, AtariST, Amiga, Atari 8bit
Website: https://h4plo.itch.io/shift

Shift came to us way back in 2008, in a Flash game created by Antony Lavelle and Armor Games, and quickly became a viral hit. In a short time, the author published three sequels and numerous ports for various portable and non-portable platforms. Immediate, compelling, easy to play and watch. A must-have for all alternative gamers.
This demake for retro platforms is beautiful. In every version.

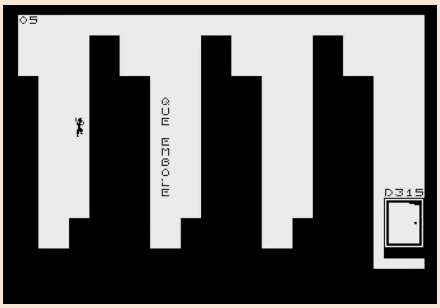


The game seems simple, but it makes us think about how to get our stylized, all-black protagonist out of each level. Between impossible jumps and maximum precision, the beauty lies in "shifting" between black and white, reversing the style of play to make the path easier.

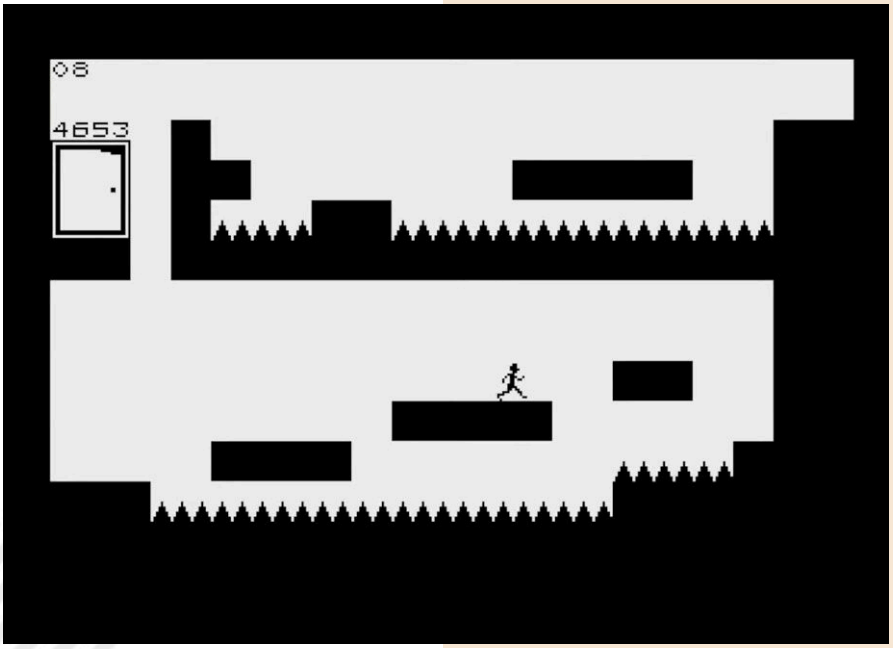
This worked in FLASH and works perfectly on Commodore 64, Atari 8-bit, ST, or Amiga. Same gameplay and same desire to get involved to understand how to approach the numerous levels.

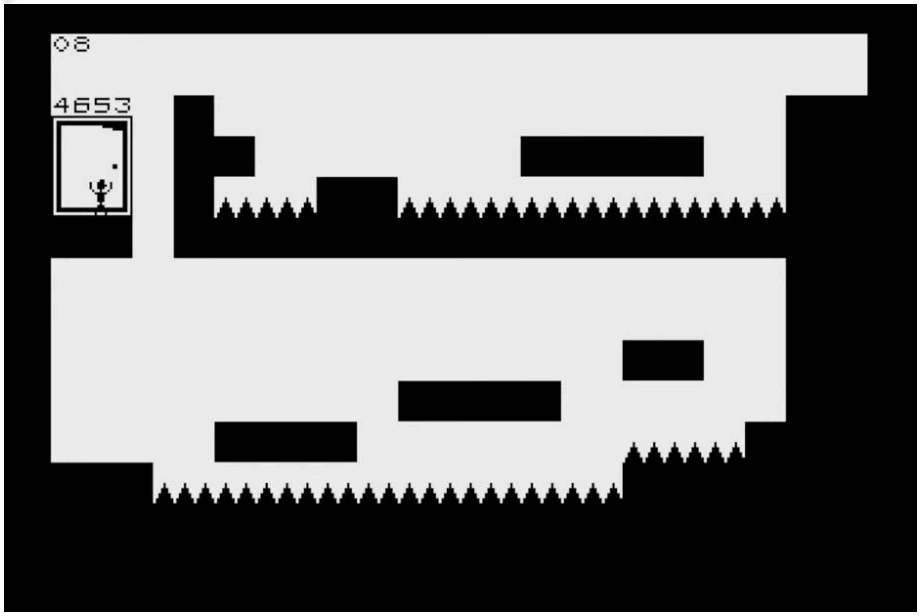
Everything works perfectly with minimalist yet beautiful graphics on all four platforms, excellent sound (really nice on Amiga!) and the already solid gameplay mentioned above.
Absolute minimalism for total fun! Multiplied playability.

The four versions are equivalent. More detailed on Amiga and C64, but definitely beautiful to look at and play even on 8-bit Atari.



Another hit from Haplo, who will surely continue to work on a possible port for other retro platforms and on a





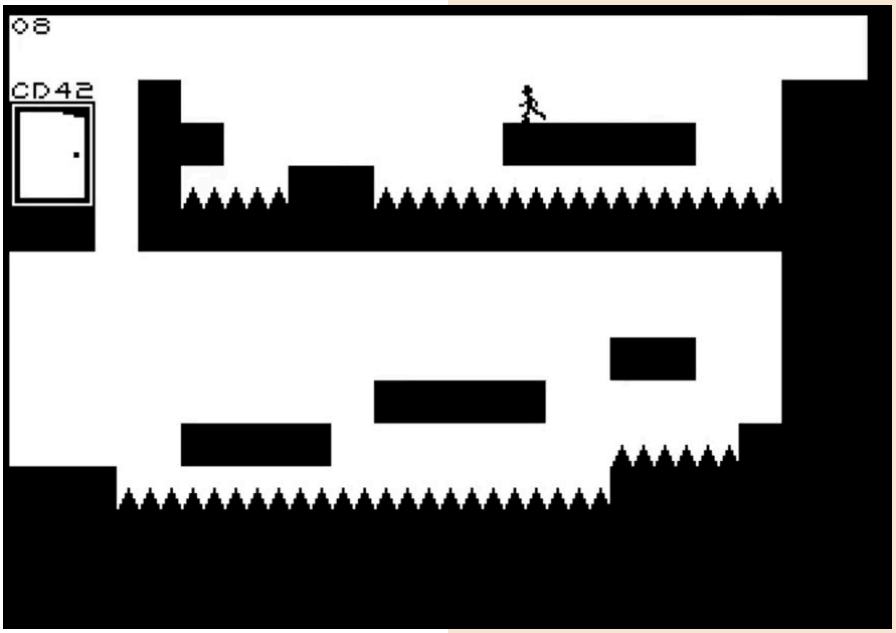
(very difficult) conversion of the second chapter.

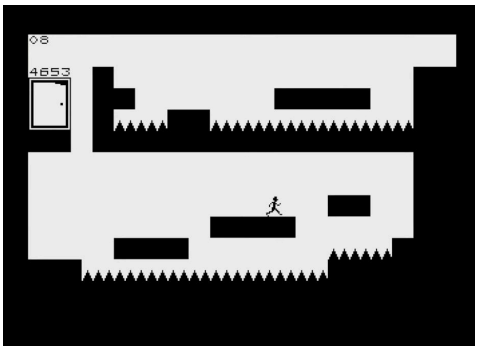
by **Marta Rossmann**

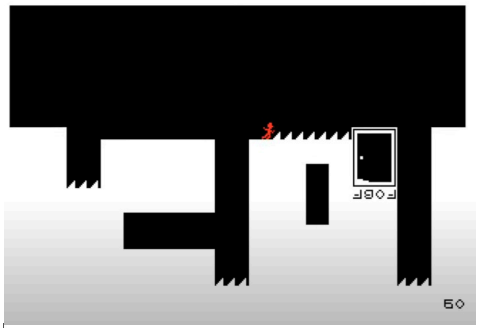
OUR FINAL SCORE

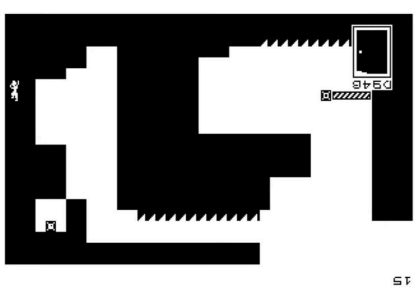
» Gameplay 90%
Splendid game mechanics.

» Longevity 80%
The challenging but never frustrating level design will allow you to appreciate it game after game, and its 37 levels (44 on Amiga) will keep you busy for many, many hours.

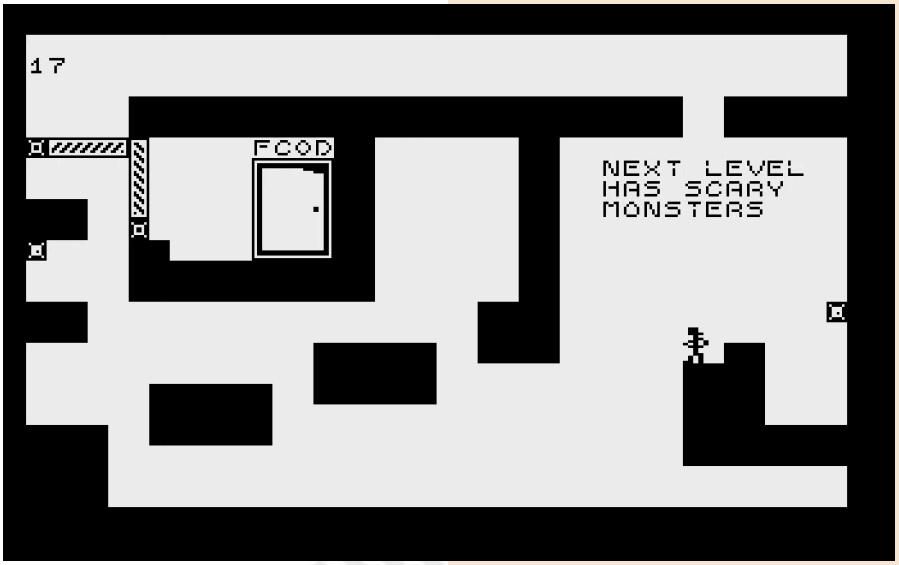








Versions: Amiga (above) **Atari 8bit (below)**



METAL SLADER GLORY DIRECTOR'S CUT

Year: 2000
Editor/Developer: HAL Laboratory
Genre: Avventura grafica/ testuale
Platform: Super Famicom

In 2000, we were already well into the PS2 era, but in Japan, the Super Famicom (the Japanese name for the SNES) continued to be very popular. Metal Slayer Glory Director's Cut was the last title released on this console and was published on November 29, 2000.

While in Europe the publication of titles stopped in 1997, in the Land of the Rising Sun it continued until the end of 2000, demonstrating an uncommon loyalty to Nintendo's 16-bit machine on the part of a hard core of enthusiasts.

This Director's Cut is an improved version of the original 1991 title released on the NES. It is a visual novel created by mangaka Yoshimiru Hoshi.

Although it is a genre that is almost unknown in the West, it features perfectly balanced gameplay and a compelling story. The incredible graphics perfectly complement the game, making it seem far ahead of other similar titles on the market. Its only flaw? The lack of an official English translation.

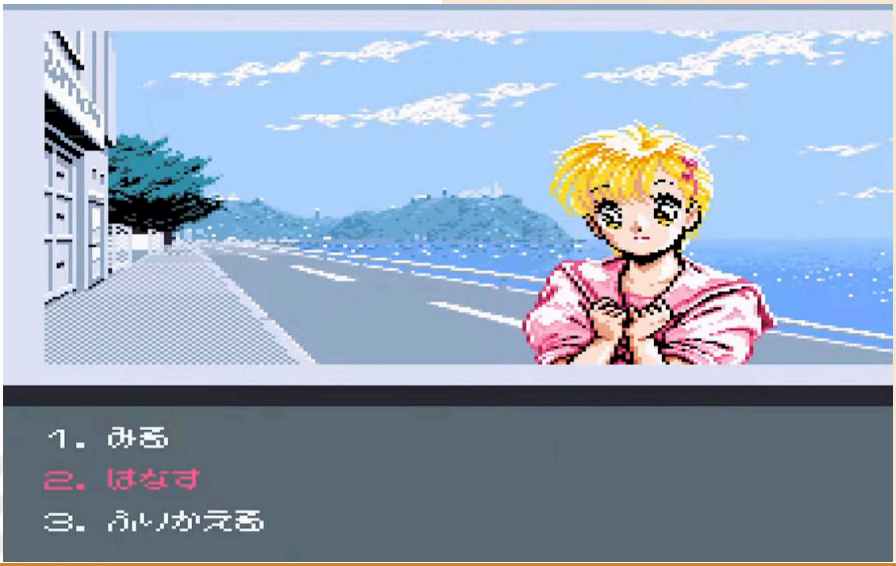
Metal Slader Glory plays out like a normal visual novel, but with fun deviations from standard gameplay. The gaming experience will normally lead us to ask around for clues to reach the next location. As a result of the incredible amount of text, many more dialogues can be activated simply by using the same option twice.



However, you will experience other gameplay styles, such as a dungeon crawler mission or one inspired by a strategy game. Atypical for the genre but definitely fun.

As a visual novel, the story is obviously the strongest aspect of the game. It all starts with Tadashi, our protagonist, and his friend Elina trying out a work mech they bought from Gen's Auto.







OUR FINAL SCORE

» Gameplay 80%

It is not difficult to learn the game mechanics despite the language. Some innovative sections make it less repetitive than the visual novels of the time.

» Longevity 80%

Once you understand the story, it will keep you glued to the controller. A product for true OTAKU.



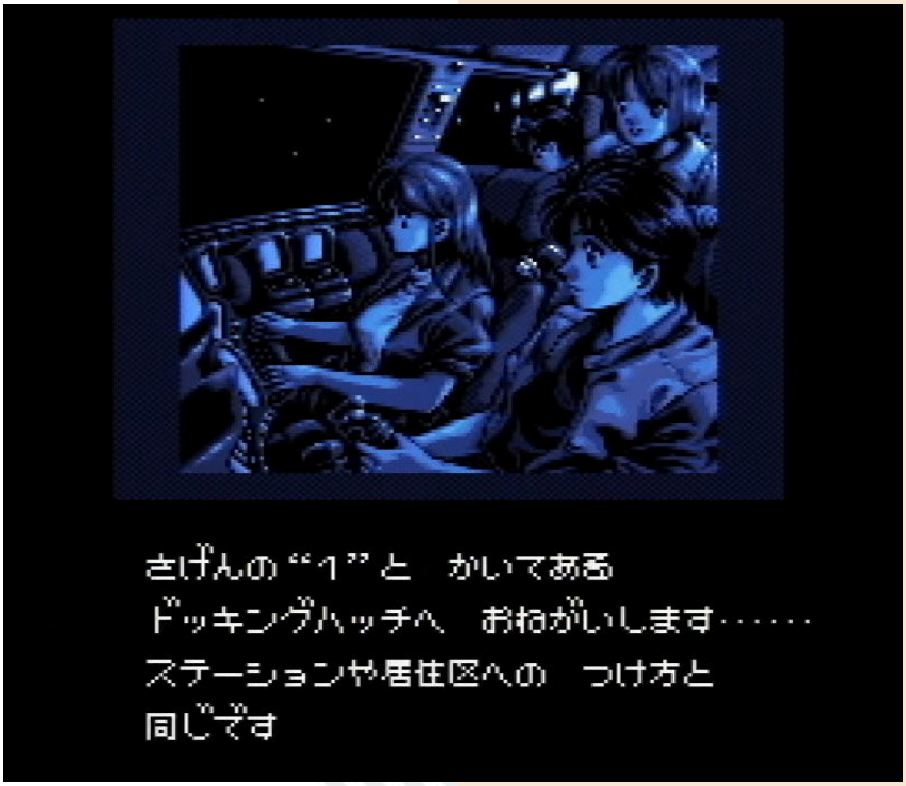


When Tadashi jumps in, the cover comes off and it turns out to be a Slader, a war mech used during a conflict in space, which were supposedly dismantled after the war. However, this Slader sends a threatening message saying, "Earth in danger. Search for the creator immediately." Intrigued by this, he, Elina, and Tadashi's sister, Azusa, head to Gen's house to find out more. This simple premise leads to a conflict greater than any of them could have imagined. The story starts off quite well and establishes the plot organically, while also giving us the characters' personalities.

The visuals are definitely well done. Bright colors, well-characterized characters, and beautiful backgrounds. The music is the highlight, lively and vibrant with emotion.

There are some translation patches available online to make the text and story more understandable, but with a little practice, it is possible to complete it even without knowing Japanese, obviously losing the narrative and especially the beautiful final dialogues. It is reminiscent of the text adventures of the past but with a mecha plot typical of the late 1990s/early 2000s. A historic title that I invite you to rediscover if you love the genre, manga, and apocalyptic mecha themes.

by **Roberto Del Mar Pirazzini**



NEW GAME

QYXAR

Year: 2024
Editor/Developer: 8bit Milli Games/William Hicks
Genre: Shoot em up
Platform: ColecoVIsion
Website: https://consolemoo.com/shop/ols/products/rom-qyxar-colecovision-rom-dng-pln-clc1









For more than forty years, aliens, monsters, mechanical creatures, and evil madmen have been trying to destroy, conquer, and annihilate the human race.

This, of course, happens in video games where prodigious heroes in their spaceships face and destroy the danger.

Yet, after all this time, they continue to invade, and this time they are attacking our ColecoVision consoles.

Qyxar is a fixed-screen shoot 'em up game in which we are tasked with ridding the galaxy of a rather brutal insectoid alien race.

In the most classic of this genre of games, we find ourselves facing hordes upon hordes of warlike buzzing monstrosities. It will not be an easy task. As the levels progress, the waves of attack become increasingly aggressive and faster.

We have a resistance shield that protects us from hits (obviously degrading) and essential power-ups that increase our defensive capabilities.

The ability to move around the screen and not remain static is an interesting feature that adds a little extra spice to the title.

Qyxar has proven to be a good title, fun and carefully developed, available in digital edition for only $14.

Personally, I would have created different types of creatures and worked on a greater number of patterns for the enemy waves, but that's just my personal opinion.

Approved!

by Carlo Nithaiah Del Mar Pirazzini

OUR FINAL SCORE

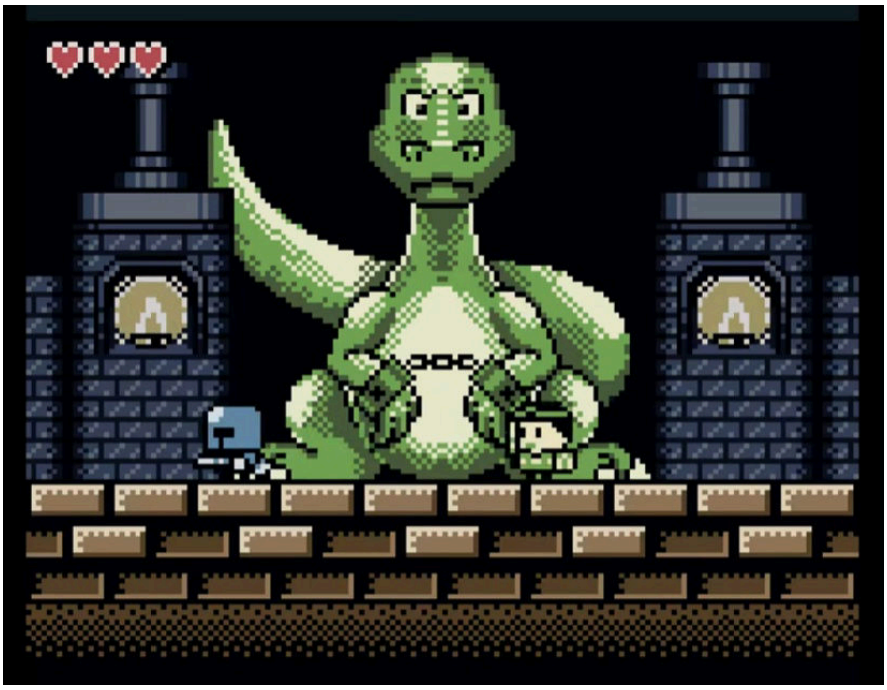
» Gameplay 80%
A well-developed product with some interesting features.

» Longevity 70%
A tough title as you progress through the levels. It can become tiresome in the long run due to the repetitive nature of the action and the waves that are quite similar to each other.

NEW GAME

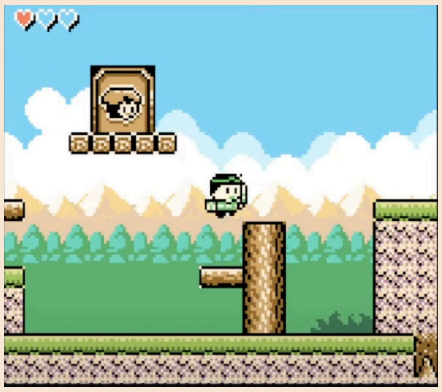
POKETTOHIRO!

Year: 2024
Editor/Developer: Broke Studio!/ David Marin
Genre: Action/Platform
Platform: Game Boy Color
Website: https://pokettohiro.com/









Portable hero! Kawaii-super deformed hero and more! This is a truly engaging game that reminded me of some titles from the past, such as Wonder Boy in Monster Land (my father's favorite game).

It's a beautiful adventure where a group of brave heroes must collect the fragments of the Magic Crystal to prevent an evil Black Knight from destroying the kingdom.

It has "open" gameplay just like Monster Land and the ability to choose different heroes (6 in total) with different game characteristics. From the swordsman to the archer, from the hawk warrior to the fairy. There is a lot of variety in a title that pays homage to the classics of the past.

The story seems trivial but is very engaging, and the game promises a few hours of fun before it is completed (about twenty will suffice!).

The playable characters each have their own physics and abilities. The latter are necessary to overcome obstacles, defeat enemies, and discover the various secrets hidden within the game.

I enjoyed exploring the micro world of Pokettohiro! It drew me in and led me to view every single frame of the map.

It is a digitally born title that can be purchased on the developers' itch.io page, but it is very rewarding in its physical version. The cartridge is beautiful to look at. Nice packaging with a chunky 32-page manual and some gadgets.

I'll conclude by talking about the exterior. Simple but engaging music and beautiful, detailed graphics rich in detail.

If you're looking for a good GBC product, this is the title for you.

by **Ingrid Poggiali**

OUR FINAL SCORE

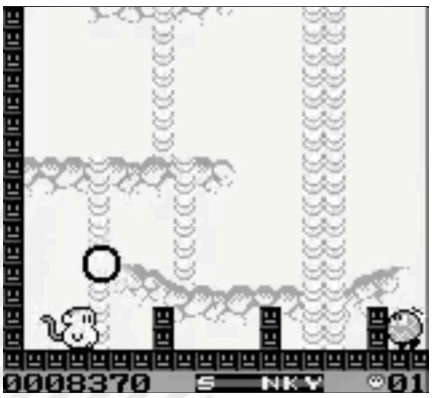
» Gameplay 92%
Lots of different playable characters and a compelling story. Simple gameplay with challenges that are never boring.

» Longevity 90%
The game should be explored in its entirety and experienced with the same spirit as the great classics of the past. Excellent!

SPANKY'S QUEST

Year: 1991
Editor/Developer: Natsume, Taito
Genre: Platform/Action
Platform: Game Boy

How wonderful is Spanky's, which I already knew about but hadn't played yet!
A title in the vein of "Bubble Bobble" and "Snow Bros." with the difference that it's an action game, even if it doesn't seem like it.
Our protagonist is a chubby little monkey whose weapon is a small bubble that bounces on his head and that we can explode at will in various sizes. The small one will deliver a blow and the large one will explode into many bubbles, killing the enemies...
During the game, we can also collect various letters that will complete the word "Spanky," which will eliminate all enemies (which are fruits) and take us to the bonus stage where we can accumulate points by bouncing a huge bubble.
The game consists of 5 stages: 1-16,

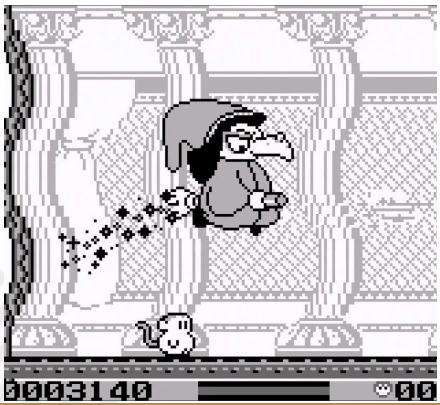


2-16, 3-16, 4-16, 5-05. Each level has a final boss, while in the 5th level we find all the bosses (they are also fruits, but giant ones). The only exception is the witch, the antagonist of the whole adventure, who, once defeated, shows us the ending.
This version is very similar to its NES



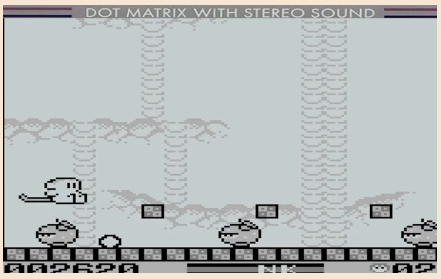
counterpart (with the necessary differences) and is set in a tower where our chubby protagonist is a prisoner of the witch.
With beautiful graphics and a good soundtrack, the game is a nice bit of entertainment and a great way to spend our (few) free hours.
Get it and play it in any form you can, even if it's on cartridge and Gb, it still looks great.
See you in the next post!
Kisses, Miss Game Boy

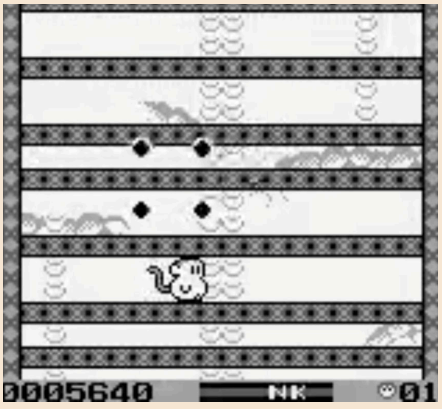
by **Barbara Murgida**











OUR FINAL SCORE

» Gameplay 85%
A classic platformer with lots of action and plenty to play. Simple and fun.

» Longevity 80%
Not impossible to finish, but definitely tough. The witch drives me crazy!

NEW GAME

THE CURSE OF LIES

Year: 2024
Editor/Developer: Rei-Vax-73
Genre: Puzzle Game
Platform: MSX2, MSXDOS2
Website: https://www.msxdev.org/2024/10/09/msxdev24-11-the-curse-of-lies/

French developer Rei-Vax-73 presents this interesting puzzle game in competition at MSXdev24.

The competition is one of the most popular and fiercely contested development "races" for the MSX standard and presents little gems every year.

The Curse of Lies sees us busy assembling the correct combination of mystical flowers to discover the Flower of Truth, which will help two sorcerers, a man and a woman, to remove a terrible curse.

As poor apprentices, the task will not be easy. The combinations will be suggested to us by the two mystics at the top of the screen, and we will have to find the right color of flower in order to progress through the levels. Each mistake will cost us a life until the dreaded game over.

Easy, right? Well, the color combinations can be misleading, which is why the game can be downloaded with a manual and instructions to help you understand it better.

A very intriguing game in terms of mechanics, which really gains a lot in terms of playability and longevity when played in pairs.

Graphically, it is well done and minimalist, and the sound is pleasant and unobtrusive.

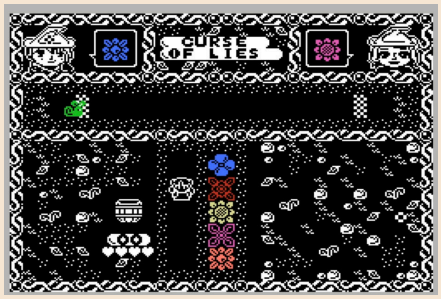
The mechanics of the combinations can be confusing without reading the manual carefully, but once you've learned them, you'll hardly ever make a mistake in the game.

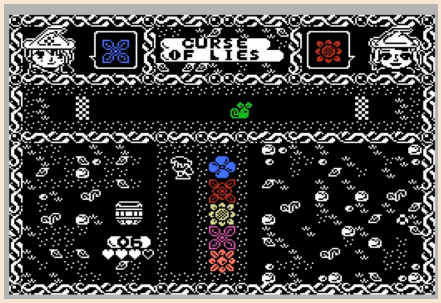
When played solo, it suffers greatly from repetitiveness, but when played in pairs, the competition makes the game much more enjoyable and fun. It may not be the best entry in this edition of MSXdev, but it's a good product.

by **Giampaolo Moraschi**











OUR FINAL SCORE

» Gameplay 75%
The game mechanics are unusual but fun.

» Longevity 70%
It gets a little boring playing alone. Find a friend!

NEW GAME

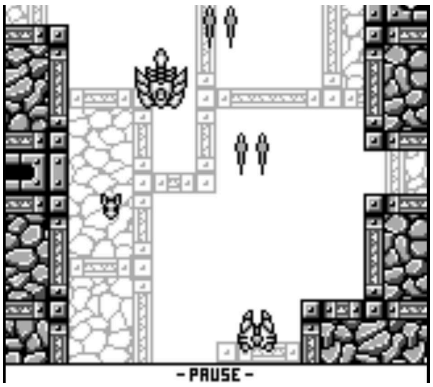
LUNAR LANCER

Year: 2024
Editor/Developer: Jonas Fischbach
Genre: Shoot em up
Platform: Game Boy
Website: https://jonas-fischbach.itch.io/lunar-lancer

This issue is packed with titles for Game Boy and Game Boy Color. The scene for this handheld console, which has just turned 35, is still very active and lively.

Lunar Lancer is a simple, well-developed vertical scrolling shoot 'em up, released by its author as a birthday "gift" for Game Boy.

It's a very linear and easy-to-learn title: there are 5 long levels full of free-falling spaceships that want to kill us, there are the classic boss fights with 5 rather tough bosses, there's our spaceship that can increase its firepower by collecting power-ups throughout the levels... in short, everything is classic in its genre, except that the power-ups have a limited number of ammunition and there is... the Konami code! The "cheat" code found in many titles from the Japanese company, which becomes essential in order to continue the adventure.



Lunar Lancer is a good game that doesn't bring innovation to the genre but finds its "home" within the walls of the Nintendo handheld console.

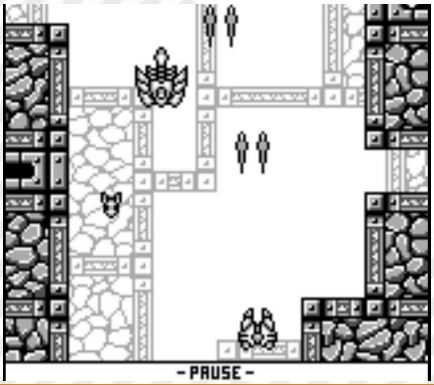
It seems tailor-made to be played between bus/subway/train stops.

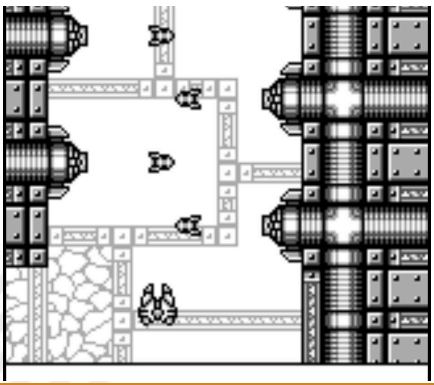
It also has good visuals that allow you to clearly distinguish everything that moves on the screen, and a decent execution speed. There are also some levels with rather impressive parallax backgrounds, and musically it makes good use of the console's audio channels.

The developer has included a password system that allows you to continue the adventure where you left off, as well as some very special "Easter eggs."

I really enjoyed this game and recommend you check it out and maybe download it to your portable device. It's worth it.

by **Giampaolo Moraschi**











OUR FINAL SCORE

» Gameplay 85%
An easy game to get into and learn, with several interesting ideas that "expand" the gameplay.

» Longevity 80%
Without the Konami code, it becomes "difficult" at certain points. The difficulty curve points upwards, but we like it that way.

NEW GAME

TIBURONCY

Year: 2024
Editor/Developer: Patagonia
Genre: Action game
Platform: Commodore 64
Website: https://patagonia.itch.io/tiburoncy-c64











Tiburoncy is a simple little game. It's a mix between a running game and a classic action game where we help a funny little shark with a life jacket rescue his friends who have been trapped by fishermen.

It's a game suitable for younger players thanks to its colorful graphics, dynamics, and setting.

Tiburoncy was created by the Patagonia collective for Flashparty 2024 in Argentina, an event dedicated to demoscene and video games for all computers in South America and Spanish-speaking countries.

The gaming and demo scene in Hispanic countries is quite solid and the developers are very talented. If you want to see more, here is the link to the event: https://flashparty.rebelion.digital/index.php/es/.

Returning to Tiburoncy, you can't expect a game with extreme depth. Here, you jump on the heads of enemies running towards you, collect keys, open levers, and move on to the next level.

In the eyes of a small child, it is certainly fun and colorful, with beautiful cartoon-like graphics that are easy to appreciate.

The soundtrack reflects the collective's experience in the demo scene. It's pumped up, perhaps detached from the action, but definitely well done.

This is a simple title, even from a learning perspective. Use the fire button to make Tiburoncy swim and press down to hit the enemy, crushing them.

Don't expect hours and hours of gameplay. It's a simple title, and that's fine.

If you have a small child, it's perfect.

by **Giampaolo Moraschi**

OUR FINAL SCORE

» Gameplay 70%
One button to swim and your butt to crush enemies.

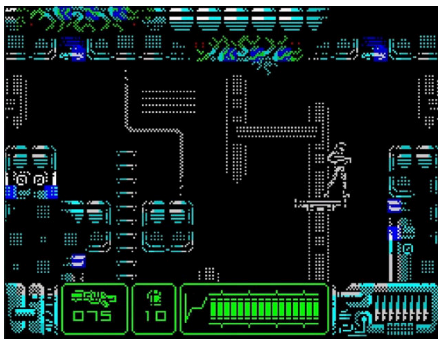
» Longevity 65%
Easy to complete, but fun to play with your kids.

NEW GAME

ALIENS NEOPLASMA 2

Year: 2024
Editor/Developer: Sinc LAIR
Genre: Action/Platform
Platform: ZX Spectrum 128k, ZX Spectrum NEXT
Website: https://sinc-lair.itch.io/aliens-neoplasma-2

The first Alien Neoplasma was a great game, but this game is a damn masterpiece!
Set in the Aliens universe and a direct sequel to the aforementioned first chapter, it sees Lieutenant Smith's escape from the secret Weyland-Yutani laboratory.

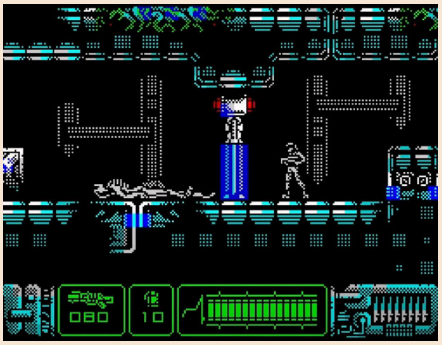


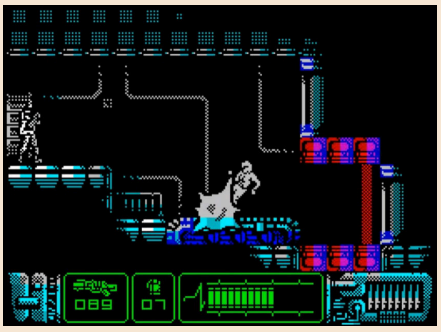
Initially armed with only a simple screwdriver, our heroine must approach the adventure intelligently and astutely in order to escape alive from the workers and discover what is really going on.
A masterfully crafted product in both versions.
The NEXT version supports a screen refresh rate of 60 Hz and showcases impressive technical prowess with masterful use of color.
The ZX 128k version is perhaps the highest peak for a title on this home computer. It is well-designed in terms of animation and appearance and wonderful to listen to.
The first title shone in terms of playability, and this second product perfects it even further. The control system is wonderful and allows you to control every action of the protagonist. Interaction with the environment and the need to think strategically greatly increase the longevity curve.
In some places, it is still 'pixel perfect' and takes a little practice, but it is undoubtedly one of the best modern titles for ZX 128 and one of the best games for NEXT in 2024.
Don't miss it.

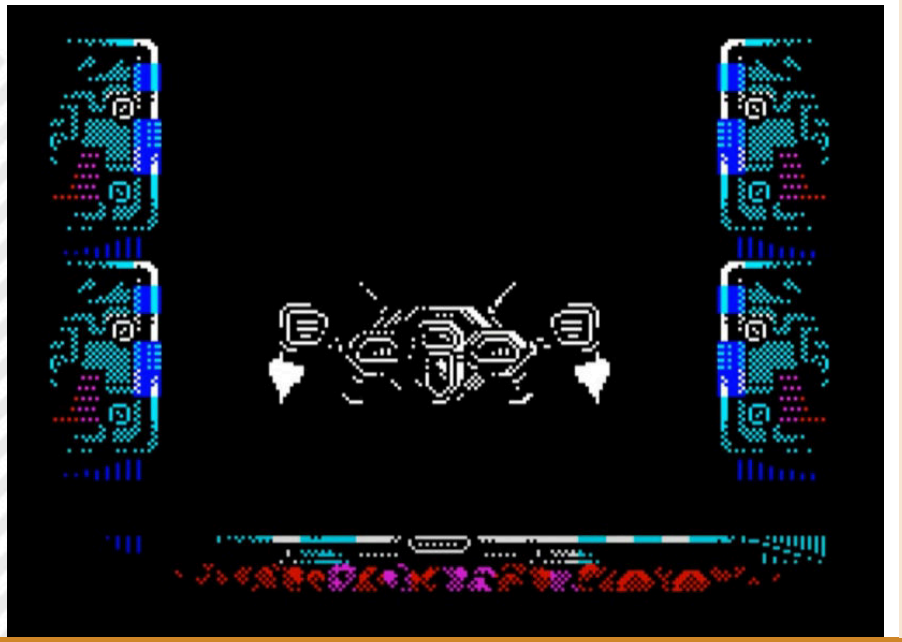
One last thing: the NEXT version is unstable in emulation.

by **Roberto Del Mar Pirazzini**









OUR FINAL SCORE

» Gameplay 95%
Immediate, dynamic, and with a great game structure. Enjoyable from start to finish.

» Longevity 95%
You'll be glued to the screen until the end, and if that's not enough, there are three additional difficulty levels.

NEW GAME

TWIST OF SOULS

Year: 2024
Editor/Developer: LarsImNetz
Genre: Platform
Platform: Atari 8bit
Website: https://forums.atariage.com/topic/374525-twist-of-souls/

Pixel Perfect! This Twist of Souls is definitely Pixel Perfect. A port of Fran Tovar's title seen on PC in 2018, it is an exploratory platform game where we guide a maiden inside an ancient and mysterious temple that holds a curse.

The technical transposition is practically perfect and adapts to the graphic characteristics of the machine. Not that it was complicated to reproduce, given the minimalist component of the PC title.





The gameplay is interesting. In order to tackle the game, you need to "sacrifice" yourself and overturn the classic vision. In short, at certain points, you can tackle the level by turning it upside down. This is done by pressing the space bar once you have passed through a large red door. The upside-down world allows you to tackle the levels in a completely different way and reach places that would otherwise be impossible in the "canonical method."
The title came third in the recent ABBUC contest 2024 and requires an Atari with at least 64Kb of RAM.

Minimalist sound accompanies all of us players in a hell of almost impossible jumps and a good dose of frustration. The game is well developed but definitely difficult even for the most "savvy" of players.
If you like impossible challenges, this is the title for you!

by **Giampaolo Moraschi**











OUR FINAL SCORE

» Gameplay 75%
Interesting game mechanics.

» Longevity 75%
Difficult and heavily Pixel Perfect.

NEW GAME

CHIMNEY MAN

Year: 2024
Editor/Developer: Gamecast
Genre: Action/Platform
Platform: ZMSX, 32KB
Website: https://www.msxdev.org/wp-content/uploads/2024/06/MSXdev24_ChimneyMan_v1.5.zip

We received a kind request from GameCast to review their game CHIMNEY MAN for MSX, developed to participate in MSXDEV24.
Of course, we didn't need to be asked twice. :-)



Chimney Man is a fun platform game where you play as a chimney sweep who must retrieve his tools to clean the chimneys of all the buildings in the city.

Unfortunately, the tools are locked in a storage room, and the key is on top of a building. Oh well, what can you do? We're chimney sweeps, and climbing buildings is our job. Too bad the ladder to the roof is crossed by lethal electric shocks...

So arm yourself with timing and patience and go get that key!

The concept is definitely cute, but... that's all there is to it. After completing the first and second levels, which differ only in the number of chimneys, 2 in the first and 4 in the second, the other levels repeat in the same way. It's a shame, because the graphics are very cute, colorful in just the right way, and pleasing to the eye, in full retro style. The sound isn't bad either. Probably the tight deadlines of MSXDEV did not allow for the addition of other features to the program, which we strongly recommend doing.

The basis for a good game is there, it just needs a little variety to become a title that could effectively become a nice commercial product.

You can also play it online:
https://www.file-hunter.com/MSXdev/?id=chimneyman.

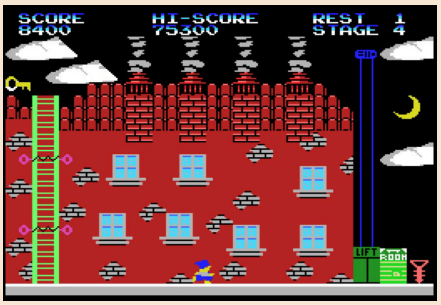
by **Francesco Fiorentini**











OUR FINAL SCORE

» Gameplay 70%
It doesn't take long to figure out what to do and to complete the levels.

» Longevity 50%
But once you discover that there are only 2 levels, you're unlikely to play it again.

BACK TO THE FUTURE II

Year: 1990

Editor/Developer: Imageworks

Genre: Tie-in

Platform: Commodore 64

I remember it like it was yesterday, that Christmas Eve when my father brought me home a double gift, namely Golden Axe! I rushed to the store with the receipt to exchange it and so I got the game Back to the Future 2: on Commodore 64 screens and many other platforms with the original title Back to the Future Part II by Images Software. I've always been a bit skeptical about tie-ins, but with a title like this, which included several mixed levels with different gameplay, well, that was the beauty of it for me, not to mention the original music that accompanied us throughout the loading, bringing back the Christmas atmosphere of the 90s with first screenings and games! Let's start with the first level, in which we play Marty McFly aboard a hoverboard in the year 2015 (has anyone seen one?), where we have to reach the town hall on roads and intersections with horizontal and vertical scrolling. We are disturbed by enemies, also on hoverboards, an elderly Biff with his cane reminiscent of a courtyard doorman, and various obstacles such as puddles. Various bonuses, such as additional energy and various incentives, will come to our aid. The second level sees us inside a house with a top-down view, where we have to get Jennifer out without her encountering the other occupants; this requires a minimum of strategy but is not impossible. The third level, on the other hand, is a horizontal scrolling beat 'em up set in Hill Valley. Now at the mercy of Biff's thuggish henchmen, we have to fight our way through with fists and weapons to reach the other side of the level, where Doc is waiting for us with his DeLorean. The fourth level is a puzzle in every sense of the word. The aim of the game? Everyone knows it, but if you find it tough, don't worry, once the time is up, you can start again from there, and the same goes for all the others. The fifth and final level is identical to the first, except that now we also have enemies on foot, a big black car driven by Biff who will try to run us over. At the end, we get to see the funniest animated sequence in the film and get our hands on the diary. I would vote the graphics and sound as the best of the year, and even the gameplay deserves the red carpet treatment, as I was eager to play it again. The longevity is dictated by the fourth level with that damn puzzle that makes us sweat our brains out. It must be said that, in addition to the legendary soundtrack of the title, the loading times are not long, and at the beginning, you can watch the scene in which the DeLorean takes off at full speed in all its glory. Many of you, or at least those who have seen the film, will already know that the ending could be disappointing... But instead, it is simply a clear invitation to grab the entire trilogy, both in film and video game form, to relive those golden years.

With Christmas just around the corner, the lights already on in some towns and the gift list almost complete, we can enjoy this Christmas issue under the tree and, without being too repetitive, go back to playing this and other titles!

by **Daniele Brahimi**







OUR FINAL SCORE

» Gameplay 80%
Five different levels, which play very well.

» Longevity 80%
Simple in some parts, but you'll want to play it again just to listen to the music...

[... continued from page 2]

Preservation of software and languages: Many programs and games were distributed with listings and source code published in magazines or manuals. Their digitization is essential to preserve vintage software and programming practices from those years.

Cultural memory: Magazines from the 70s, 80s, and 90s offer a unique insight into the lifestyles and aspirations of the time, especially in relation to the evolution of computing and gaming.

Emulation support: Scans contain useful documentation for emulator developers, allowing for greater accuracy in the reproduction of vintage consoles and computers.

Recovery of lost content: many games and applications from the past were never digitally distributed and, without scans of manuals and magazines that mention them, would risk being forgotten or irrecoverable.

Preserving the memories of a generation: for many enthusiasts, digital scans represent a gateway to their childhood or adolescence, materials that fuel memories and nostalgia, significant aspects of retrogaming.

Educational and teaching resource: digitized materials can be used for courses on the history of computing, digital culture, and even to teach the basics of programming through historical examples, making learning more engaging.

Preventing the risk of physical loss: paper deteriorates over time. Digitization ensures that this information is preserved beyond the physical life of the original documents, protecting it from accidental destruction or deterioration.

For all these reasons, digital scanning is an extremely valuable activity for safeguarding the technological and cultural memory linked to computers and consoles of the past, making available a heritage that would otherwise be at risk of disappearing. So I invite you too to think about it and, when you pick up your tablet late at night to read a manual, book, or magazine from the past, think about how you can repay all those groups that spend time and resources fighting the battle of time to give us pieces of the history of our beloved retrocomputers. Especially those who do so in a totally selfless manner and with the sole purpose of sharing and universal dissemination.

David La Monaca

Disclaimer

RetroMagazine World as an aperiodic magazine entirely ad-free is a non-profit project and falls off any commercial circuit. All the published material is produced by the respective authors and published thanks to their authorization.

RetroMagazine World is licensed under the terms of: Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) https://creativecommons.org/licenses/by-nc-sa/4.0/

This is a human-readable summary of (and not a substitute for) the license. You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material

The licensor cannot revoke these freedoms as long as you follow the license terms. Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.



RetroMagazine World
Year 4 - Issue 27 - WINTER ED. 2025

Editor in Chief
Francesco Fiorentini
Editing/Deputy Manager
David La Monaca
Publishing Manager
Marco Pistorio
Website/Social Media Managers
Giorgio Balestrieri/Carlo Nithaiah Del Mar Pirazzini

AI Disclaimer - No content included in this digital magazine (editorials, articles, reviews, any texts or images) has been produced or generated by AI tools