PROGRAMMING - HISTORY - INTERVIEWS - REVIEWS - INSIGHTS

Future days are back

RETROMAGAZINE WORLD

ISSEU 25 - WINTER EDITION- WWW.RETROMAGAZINE.NET

ISSN 3103-697X



COMMODORE 64 HISTORY

GODZILLA BALLS

LITTLE MASTERPIECE ON MSX

We've reached 300... thousand!

As you know, we don't like to brag too much about RetroMagazine World's achievements, but this time we want to make an exception.
We've been keeping an eye on the number of downloads for a while, and finally, the moment has arrived.
RetroMagazine World issues in Italian and English have been downloaded more than 300,000 times in total (actually it is almost 400.000 now!).

Of course, if we look at the number of views on YouTube, where some of our videos have well over a billion views, this number may seem insignificant, but for us at RetroMagazine World, and for me personally, it is a remarkable achievement that deserves to be celebrated. When we started this adventure in 2017, few believed that we would have the perseverance to carry on with a project of this magnitude. Instead, these numbers confirm how much good work has been done so far and how much more there is to do. Yes, because even though 300,000 downloads may seem like a lot, they are only a milestone on our journey, and we hope to achieve many more goals.

Ah, and there's another consideration to add...
These are only the downloads we can count from our website, but since some sites mirror our numbers, the actual number of downloads is definitely higher. We don't know how much higher, so to be fair, we always refer only to this data.

Obviously, our intention is not to chase numbers, otherwise we would have to and could adopt different strategies, but to offer each issue a wealth of quality content and passion that I believe all our loyal readers have come to know and love. However, the comfort of knowing that our passion is not an end in itself gives us the strength to continue working hard and to continue this magnificent adventure.
See you at the 500,000 download milestone, which we hope to double soon.

Oh, I almost forgot, a huge THANK YOU to all of you readers, without whom RetroMagazine World would be nothing more than a beautiful exercise in style.

Francesco Fiorentini

Publication Note: this editorial and all the contents were first released in Italian in April 2024 - ref. issue 47-IT

TABLE OF CONTENTS

◊ Commodore 64 Pag. 3
◊ Cases of identity... - part two Pag. 13
◊ The "Mini-Lotto" game Pag. 18
◊ Guide to SGDK - part two Pag. 24
◊ PunyInform: installing tools and compiling a game Pag. 28
◊ Fortran on the Commodore 64 - the MEM command Pag. 32
◊ Invitation to ... "Personaggio" for C64 Pag. 35
◊ Rotor Rampage Pag. 36
◊ Interview with Richard/TND, a code maniac Pag. 40
◊ Japan 24th episode: Gacha...pon! Pag. 50
◊ Pang+ (Amstrad GX4000) Pag. 52
◊ Shadow Gangs (Dreamcast) Pag. 54
◊ Great Giana Sisters (Plus4) Pag. 56
◊ King of Demons (SNes) Pag. 58
◊ Go For It! Goemon (PS1) Pag. 60
◊ Alice Sisters (Atari Falcon) Pag. 62
◊ Tetris (Atari 8bit) Pag. 64
◊ Xiaomei and the Flame Dragon's Fist (MD) Pag. 66
◊ Harhargon (C64) Pag. 68
◊ Trucho (MSX) Pag. 56
◊ Good Knight (C64) Pag. 72
◊ Savage Princess 2 (ZX) Pag. 74
◊ Godzilla Balls (MSX) Pag. 76
◊ Mai Nurse (GB/PC Engine) Pag. 78
◊ Bomb Jack the Remake (SMS) Pag. 79
◊ Arkon Blazet (C64) Pag. 80
◊ Beach Head (C64) Pag. 82

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

- David La Monaca
- Carlo N. Del Mar Pirazzini
- Daniele Brahimi
- Francesco Fiorentini
- Francesco Donatini
- Roberto Del Mar Pirazzini
- Giampaolo Moraschi
- Leonardo Miliani
- Fredrik Ramsberg
- Ingrid Poggiali
- Eugenio Rapella
- Fredrik Ramsberg
- Marta Rossmann
- Gianluca Girelli
- Marco Pistorio
- Michele Ugolini

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

Original issue in Italian released in April 2024

Commodore 64

by Leonardo Miliani

In 1981, development began on a new system designed exclusively for gaming which, after various ups and downs, was converted into a fully-fledged computer. This computer was marketed to replace its successful predecessor: not only did it perform better, it became a best-seller in its own right, recording sensational sales figures and becoming the best-selling computer of all time. We are talking about the Commodore 64, commonly known as the C64 (fig. 1).



Fig. 1 - Commodore 64
(author: Evan-Amos – source: Wikimedia Commons)

Commodore before the C64

Jack Tramiel, a Polish survivor of the Nazi death camps who emigrated to the United States after World War II, founded Commodore Business Machines (CBM) in 1954, together with Manfred Kapp. The two met while working together at Ace Typewriter Repair Company, a typewriter repair shop. After a few years as employees, they decided to go into business for themselves, repairing and selling used or reconditioned typewriters. With Tramiel left alone at the helm of the company, after various ups and downs, Commodore entered the calculator market and then, in the mid-1970s, the computer market, creating one of the three members of what is known as the "trinity of 1977," the PET. This competed in the nascent personal computer market alongside the TRS-80 and the Apple II. Sales were good: the PET took the top spot thanks to the variety of models on offer, which satisfied not only home users but also small businesses, as some models had 80-column screens.

In 1979, however, something strange happened: VisiCalc, the first electronic spreadsheet for computers, was



Fig. 2 - The Commodore VIC-20
(author: Evan Amos - source: Wikimedia Commons)

released for the Apple II. The program became an instant success: all companies wanted it to simplify their accounting. Since the program was initially only available for the Apple II, people started buying this computer in order to use VisiCalc. The Apple II also had another advantage: color, which the PET lacked, generating only monochrome images. The TRS-80 was also monochrome, which is why Tandy-Radio Shark introduced a new computer in 1980 capable of generating color images, the TRS-80 Color Computer. Despite this computer's lackluster capabilities, it still sold reasonably well thanks to the fact that, being a Tandy product, it was available in its Radio Shark chain of stores, which had thousands of outlets across the United States. With the prices of computer hardware components falling, Tramiel decided to enter the home computer sector by creating a low-cost machine capable of producing color images and aimed at inexperienced users, which could therefore be sold not only in electronics stores but also in department stores and toy stores. In 1981, the VIC-20 (fig. 2) was launched, a computer that could be connected to a standard home television and was capable of generating color images and more complex sounds than the beeps emitted by PETs equipped with internal speakers. Its price of around $300 ensured its immediate success: the VIC-20 was the first computer in the history of computing to reach one million units sold.

A new chipset

Like any self-respecting company, once a product is on the market, the next one is immediately considered, and

so it was for Commodore. However, the company had to deal with a situation that was certainly not peaceful within the company: despite the success of the VIC-20, the PET was still generating excellent sales, and Tramiel did not seem willing to take it off the market. This created discontent: there was a dissident group of engineers and managers who did not approve of the company's focus on office machines such as the PET and were pushing to continue along the path opened up by the VIC-20, i.e., to produce machines intended primarily for home users. What's more, the creator of the PET, Chuck Peddle, recently left the company after clashing directly with Jack Tramiel over the VIC-20: Tramiel rejected the TOI (The Other Intellect) project that Peddle had presented to become the future VIC-20, favoring another project instead. With its creator gone, many believe that the PET has lost its point of reference and the person who could have continued its development.

Instead, the commercial sector is pushing to continue on the path of the VIC-20, which is recording record sales, so it has no desire to jump off this bandwagon at the moment. However, MOS Technology engineers are well aware that this success cannot last: the VIC-20 is objectively inferior to other offerings on the market, such as Atari's 8-bit computers, the 400 and 800. In agreement with Charles Winterble, head of MOS Technology, they decide to start developing its successor. Winterble begins discussing the idea with Al Charpentier, the designer of the VIC video chip that gave the VIC-20 its name, and together they lay the foundations for a new chipset. The heart of the new computer would be called the 6510 (fig. 3), basically a 6502 with an additional I/O port, while the CIA, an advanced version of the PIA, was designed to manage peripherals. The crucial point was the video department: the VIC-20 had a chip that generated a color image but with only 22 columns of text. However, there

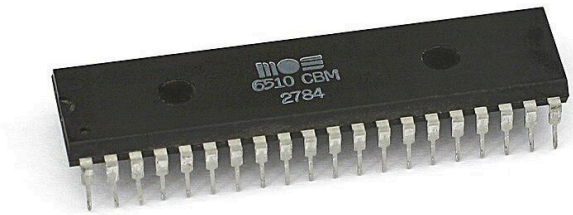


Fig. 3 - The MOS 6510 CPU (photo: Konstantin Lanzet - source: Wikimedia Commons)

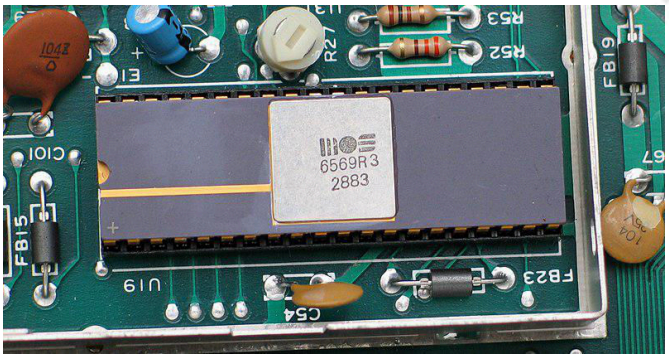


Fig. 4 - The MOS VIC-II, graphics chip of the C64 (photo: Christian Taube - src: Wikimedia Commons)

was a new video chip in the pipeline called VIC1.5, which was nothing more than the VIC upgraded to 40 columns. This chip was designed for a color version of the PET as well as an upgrade for the VIC-20 itself, intended to replace the VIC thanks to the fact that it maintains the pinout of the original chip: however, neither project ever saw the light of day, and the VIC1.5 remained in the drawer. Winterble requested a meeting with Tramiel and proposed using this graphics chip as the basis for a new project aimed primarily at games but which could also be converted into a computer. Tramiel liked the idea and gave Winterble the go-ahead, who then commissioned Charpentier and Bob Russel, another engineer who had worked on the VIC-20, to study the gaming systems on the market to get some ideas: in ten days, they examine the consoles and computers on the market, such as the Intellivision, Atari computers, and the TI99/4A, and draw up a list of the features the chip must have. Finally, they meet with Tramiel again to present the final project, which he approves definitively.

Charpentier set to work and designed an improved version of the VIC1.5, which was called VIC-II: the new chip was capable of generating a 40-column text image with a resolution of 320x200 pixels and 2 colors on screen from a palette of 16. However, he decided to also allow a multicolor mode with 4 different colors to make games more realistic, at the cost of halving the resolution to 160x200 pixels. It then added the management of 8 sprites in hardware with a whole series of advanced features such as collision control and the possibility of managing them in multiple colors: sprites are so important that 2/3 of the chip's surface area is dedicated to their management. Horizontal and vertical hardware scrolling was also maintained, useful for games with moving scenes. For sound, it was decided to create a separate audio chip,

thus abandoning the integrated system followed for the VIC (which generates image and audio) and following the path already taken by the competition. Bob Yannes, a lover of music and digital synthesizers, devoted himself to this project and created a chip with audio capabilities never before seen on a home system: the MOS SID (Sound Interface Device) was capable of generating sounds using four multi-octave waveforms. Since development took a long time and Yannes was unable to realize all the ideas he had in mind, instead of multiplexing the audio oscillator and creating multiple voices from a single internal source, he decided to replicate the circuit he had created three times: this is how the SID came to be equipped with three independent sound generators.

Towards the Commodore 64

Meanwhile, Tramiel decided that the computer project should be ambitious: the successor to the VIC-20 should not be just any computer, but the most powerful system in its class. Jack, confident that memory prices would fall between the start of design and the launch, decided that the new computer should be equipped with 64 KB of RAM. Development began on what was initially the VIC-40 project, a name chosen deliberately to indicate that this machine was the successor to the VIC-20. All of the above chips were delivered by MOS Technology in November 1981, and the team immediately began a real "tour de force," as Charpentier and Tramiel had decided to present the machine at the Consumer Electronic Show (CES) in January 1982. Despite skipping Thanksgiving and Christmas holidays and sometimes working 16-20 hours a day, the engineers managed to complete the prototype just a few days before CES, the set deadline.

Shortly before the official presentation, the new computer was renamed simply "C64," based on the amount of memory installed. The C64 was the real star of CES, and the Commodore stand was literally stormed by industry professionals eager to see the computer's presentation. Not only were its amazing technical features impressive, with sound and graphics never before seen on a home computer, but also, and above all, the fact that it cost only $595. After the event, the C64 returned to the engineers' benches: the prototype had to be converted into a computer that could be produced. There were still many problems to solve, some stemming from the video chip, which displayed graphic artifacts under certain conditions, others caused by read and write speed issues with the new disk drive; finally, there was the operating system, which had to be adapted to the new hardware. All this work takes a long time: the C64, which was supposed to debut in the spring, is finally released in August 1982.

The MAX (or MAX Machine)

In the meantime, Tramiel wanted to monetize the work done and decided to put into production a parallel project based on the same chipset as the C64 but intended to become a console, a project known within the company as MAX (fig. 5). The MAX was intended to replicate the success of the VIC-20 in Japan that it had enjoyed in the US, competing with Japanese products on their own turf: it was therefore a project for a game console to be sold for around $180. Since it used the same components as the C64, the game cartridges for these two systems were made interchangeable: to do this, the C64 was modified by creating a mode that made the computer's memory map compatible with that of the console so that, when the cartridge was inserted, the C64 'became' a MAX. Initial development took place in the US, but finalization was carried out in Japan at Commodore's local subsidiary by Yash Terakura.

Although it was created as a game console, the MAX was also designed to become a small computer, a solution that was very popular at the time and also adopted by other game system manufacturers, such as the ColecoVision, which could be converted into the Adam computer, the Spectravideo CompuMate kit for the Atari 2600, the Keyboard Component for the Mattel Intellivision, and



Fig. 5 - The Commodore MAX, or MAX Machine (photo: Thomas Conté - src: Wikimedia Commons)

others. But instead of selling the keyboard as an accessory, the new machine was placed directly inside a small case with a membrane keyboard similar to that of the PET. The transformation took place via a BASIC cartridge that provided, in addition to the interpreter, a screen editor similar to that of the C64 for writing small programs.

The console was released only in Japan in early 1982 as the Commodore MAX (also known as the MAX Machine). Despite its computer-like appearance, the MAX was a console in every sense of the word, as demonstrated by its mere 2 KB of RAM and the total absence of an operating system to launch if no cartridge was present at startup. Later, it was decided to export it outside Japan: it was therefore presented as Ultimax in Canada, again as MAX in Great Britain, and then as VC-10 in Germany. However, the MAX failed to establish itself due to the small price difference with the VIC-20, which, thanks to its greater expandability and flexibility, was preferred to the console. For this reason, it was withdrawn from the Japanese market a few months after its launch and after only a few dozen cartridges had been produced, and it was decided not to market it in the other markets where it had been presented because it was considered unsuitable to compete.

The debut of the C64 and the price war

As mentioned, the C64 went on sale in August at a price of $595. All the competition wondered how such a machine could cost so little. The answer was: "savings and recycling." Everything about the C64 was designed to be recycled from previous projects or developed from scratch with minimal investment. The chipset, the only thing that was almost completely new compared to computers already in production, was developed in-house by the subsidiary MOS Technology. The computer's KERNAL is that of the VIC-20, recycled exactly as it was and only adapted to the new hardware. Even the BASIC is the same: in fact, the version is 2.0 of its predecessor. Even the case is identical to that of the VIC-20, only in a different color, so the design cost is zero.

With this selling price, which included color graphics and sprites, audio capabilities equal to those of a digital synthesizer, a memory capacity normally found on much more expensive machines, and the ability to connect all kinds of peripherals and the convenience of using it on a standard home TV without having to purchase a special



Fig. 6 - The Texas Instruments TI-99/4A, one of the victims of the price war waged by Jack Tramiell (photo: Abraham San Pedro Salazar - source: Wikimedia Commons)

monitor, the C64 blew away the competition. From its debut at the end of that year, the C64 sold well and, after the Christmas holidays, recorded 360,000 units sold. But this was nothing compared to the numbers it would achieve later... In 1983, thanks to distribution not only in specialist stores but also in large retail chains and even toy stores, the C64 reached a much larger customer base than its competitors, which were only sold in specialist stores: people bought it because they were enticed by the trade press, which praised its excellent value for money. In that year, approximately 1.3 million units were sold: Atari's 8-bit computers, the 400 and 800 series, came in second place, followed by the Apple II. The latter was quickly outclassed due to its significantly higher price, almost triple that of the Commodore product. From there, a real price war began, with an aggressive marketing campaign by Tramiel. The reason was simple: to eliminate his rivals, especially Texas Instruments. Tramiel, still remembered when Commodore, in the early 1970s, as the undisputed leader in the calculator market, almost went bankrupt due to the entry into the sector of the Texas-based company, which, as a chip manufacturer, began selling complete calculators and dominated the market with very low prices thanks to the fact that the integrated circuits it sold to its competitors cost more than its own calculators. Before the end of the year, the VIC-20 was reduced to below $200, making it cheaper than its direct rival, the Texas TI-99/4A (fig. 6). In response, Texas offered a $100 discount on the purchase price of the computer, which was $300. At the same time, Atari lowered the price of its 800 to $500 to counter sales of the C64 during the

Christmas period. In response, in early 1983, Commodore lowered the price of the C64 from $595 to $399, without losing money: this was possible because the company produced the chipsets for its own computers.

In February, Texas Instruments lowered the price of the TI-99/4A to $150. In response, in April, Commodore cut the prices of its entire VIC-20 line, bringing the price below $100. Not only that, it also launched a promotional campaign that literally demolished the competition: any computer returned to Commodore would entitle the owner to a $100 rebate on the purchase of a C64. This promotion not only filled Commodore's warehouses with computers of all kinds but, absurdly, also increased, albeit temporarily, sales of Timex Sinclair microcomputers, which imported Sinclair products into the US: people bought cheap Timex-Sinclairs for less than $50 to send them to Commodore, get the $100 rebate, and pocket the difference.

In May, Atari lowered the price of the Atari 800 by $100, but at CES in June, Commodore announced that the C64 would cost only â,¬200, a third of the initial price! In October, Texas Instruments raised the white flag and surrendered to Commodore, withdrawing its TI-99/4A from the market: Tramiel's enemy was defeated. Atari, on the other hand, suffered an even heavier blow: compounded by the so-called video game crisis of 1983 that hit the North American video game market, some ill-advised marketing decisions made for its 2600 console proved to be huge commercial flops, and Atari was dragged into the abyss. The company recorded losses of more than $500 million: Atari nearly went under, and only a major corporate reorganization, with the division of the arcade and home systems departments into two separate entities and the sale of the latter, saved it from bankruptcy.

At this point, the rise of the C64 continued unchallenged. Even Apple, despite updates to its Apple II, was no longer a threat as the cost of this computer was much higher than that of the C64. In 1984, the latter recorded 2.6 million units sold, the highest peak ever achieved during the entire period in which it remained on sale: subsequently, sales continued at around 1.5 million units per year until 1989, when they fell to around 1 million units per year. The C64 also withstood "friendly fire": in 1985, Commodore introduced its successor, the C128. Despite more advanced features and almost complete compatibility with the C64, demand for the latter remained so strong that the C128 was withdrawn from the market after four years with around 2.5 million units sold, less than the C64 had recorded in its golden year of 1984. The Amiga was also introduced in 1985. Although it was a 16-bit computer with vastly superior features, the C64 held its own against this latest competitor, thanks in part to the fact that in 1985, the C64 could be purchased for $149, making it a viable, economical alternative for those who wanted a high-performance computer that wasn't too expensive.

In 1992, Commodore claimed to have sold approximately 12 million C64s from the start of its commercialization until December 31, 1991. In 1992, sales totaled 650,000 units. In 1993, sales were just over 200,000 units, while the figures for 1994 are uncertain, partly because this was the year Commodore went bankrupt, so it is reasonable to estimate that sales were low. We can say that throughout its commercial life, the C64 sold around 13 million units, although, according to other sources, the numbers could be higher. In any case, the C64 is the computer that holds the sales record, and it is unlikely to be surpassed in this regard.

Technical characteristics

The heart of the computer is the MOS 6510, a CPU derived from the 6502, with a reorganization of the pins that made it possible to add an 8-line I/O port (only 6 are used in the C64) used to control the computer's memory mapping and to directly control any connected cassette player. Apart from this external difference, the 6510 is basically a 6502, so the software can be adapted simply by modifying the parts that access the machine's hardware.

Compared to the VIC-20, where a single coprocessor handles video and audio signal generation, on the C64 these tasks are divided between two dedicated chips: the VIC-II handles the image while the SID handles the sound. The VIC-II is a chip capable of generating a color image in both graphics and text modes: in the former, the screen consists of 320x200 pixels, while in the latter it consists of 40x25 characters, and in both modes 16 colors are possible. In graphics mode, you can operate in either high or medium resolution, the latter also known as multicolor: in the former, the image is divided into blocks of 8x8 pixels, and each block allows for a background color, which is the same for the entire image, and a different primary color. The multicolor mode was designed by the

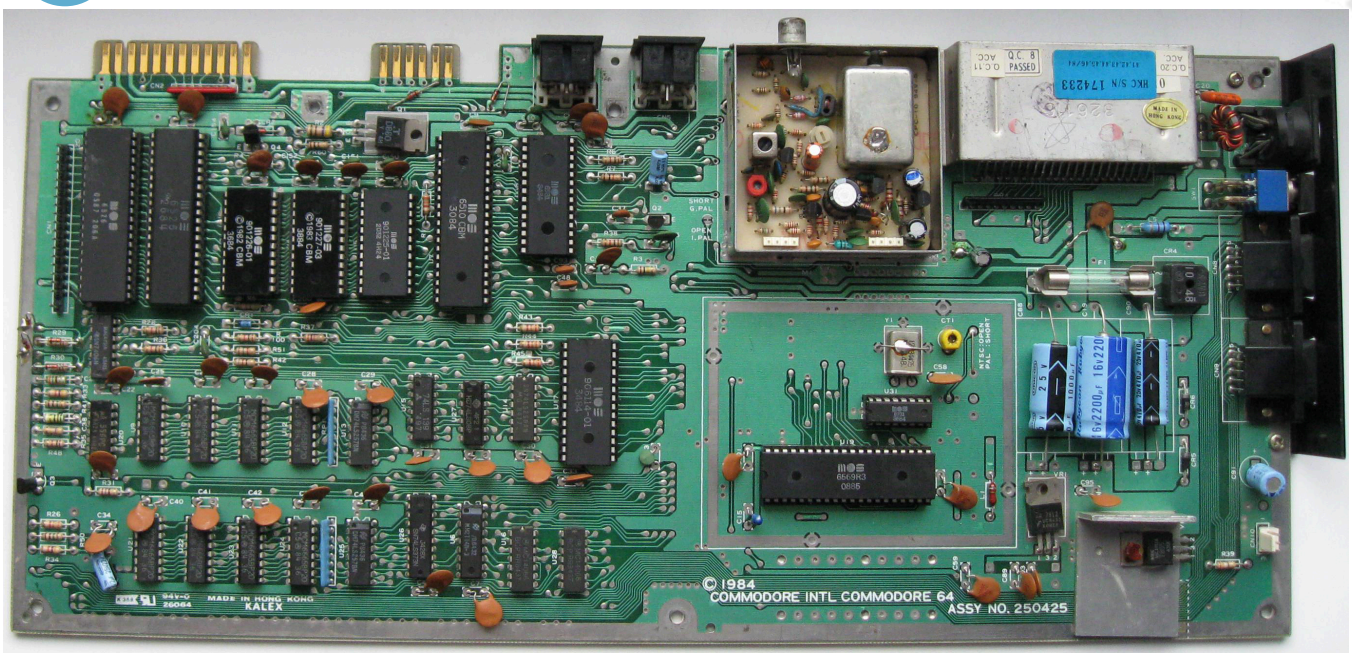


Fig. 7 - A C64 motherboard from 1984: the MOS SID can be seen at the top center and, to its left, the 6510 CPU. At the bottom, the chip placed horizontally is the VIC-II (photo: Gona.eu - src: Wikimedia Commons)

chip designers to help game programmers: in this mode, instead of using 1 bit for each color pixel, 2 bits are used. In this way, 4 different colors can be addressed per block, but at the cost of halving the horizontal resolution, which drops to 160 pixels, and, consequently, the blocks are 4x8 pixels. There are 3 text modes: the classic mode, with 1000 cells (40x25) of 8x8 pixels, each capable of displaying characters in 2 colors; the multicolor mode, in which the resolution is 4x8 pixels per cell but with the advantage of being able to address 4 colors per cell; extended multicolor mode, which allows 5 colors per cell, but at the cost of reducing the character table from 256 to 64. There are limitations in multicolor modes: in order not to take up too much memory, a series of colors common to the entire screen and others specific to each individual pixel block are set. Some color data is stored in a 1024 nibble (1024 4-bit cells) memory bank called Color RAM, which only the VIC-II can access.

In addition, the chip manages 8 monochrome sprites of 24x21 pixels or multicolor sprites of 12x21 pixels with 3 colors, with the limitations already mentioned above (only 1 color per sprite and 2 common to all). The chip also manages collisions, so you can tell when a sprite collides with another sprite or with the background. Sprites can also be enlarged to twice their size (the resolution does not change, as each pixel simply becomes a 2x2 pixel block on the screen). Finally, the VIC-II handles both vertical and horizontal scrolling, as well as an interrupt linked to the video brush, which allows you to change the graphic attributes when a certain line of the screen is reached, for example to change the color of the border or image.

The sound department is no less impressive. Created by a real musician and electronic music enthusiast, the SID is a true digital synthesizer capable of generating 4 different waveforms: sawtooth, triangular, square, and white noise. It has 3 separate voices, each capable of mixing multiple waveforms to create new ones, as well as applying filters or modulating waves to create numerous effects. In addition, each of the 3 oscillators can manage ADSR, which stands for Attach-Decay-Sustain-Release, i.e., volume management in the 4 phases of sound, respectively the time it takes to initially reach maximum volume and the duration of the first decrease, the maintenance time during the main sound generation, and finally the time it takes to reach zero volume once the penultimate phase is interrupted.

The machine's operating system occupies 20 KB: 8 KB contain BASIC 2.0, the same as its predecessor, the VIC-20, while another 8 KB are packed with KERNAL, i.e., the routines for managing the machine's basic functions. Finally, 4 KB are allocated to the character map, consisting of two different tables: the first contains the classic ASCII characters as well as a series of semi-graphic characters inherited from the PET, while the second is the "shifted" version of the first, where the classic capital letters are

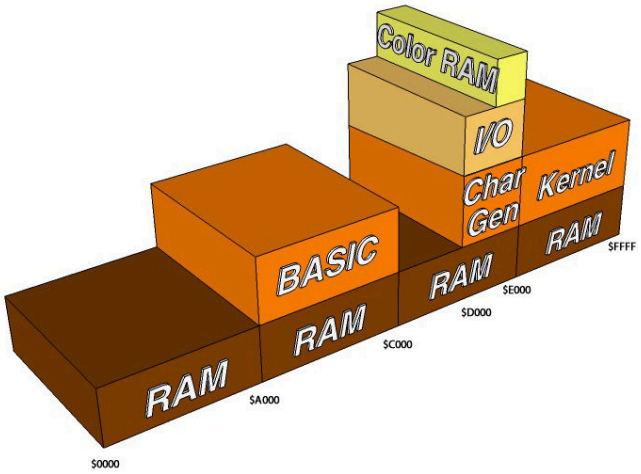


Fig. 8 - The memory configuration of the C64. Normally, the RAM that shares the same addresses as the ROM and system memory is disabled (author: Actraiser - source: Dustlayer.com)

replaced by their lowercase versions. In addition to the ROM, there is 64 KB of RAM: since the 6510 processor cannot address more than 64 KB of memory, some RAM banks are deactivated and replaced by the aforementioned ROMs so that, normally, the BASIC interpreter is active and the computer can manage user input via the screen editor. The active banks are managed by a sophisticated memory exchange system ("bank switching") controlled by a special integrated circuit called a PLA. The result of all this is approximately 38 KB of RAM available to the user for BASIC programs: if you want to have all the available memory, you can deactivate the operating system ROM and activate the underlying RAM, which occupies the same addresses, at the cost of giving up BASIC and KERNAL. (fig. 8).

In terms of interfacing with the outside world, the C64 offers a wide variety of possibilities. Behind the computer is a 44-pin connector connected directly to the CPU's data and address buses, used to connect ROM game cartridges or RAM memory expansions. There is also another 24-pin connector, called the user port, to which third-party peripherals such as printers or other devices can be connected. Next to these two primary connectors is a classic RF output connector for connecting a standard TV set, an 8-pin DIN connector for connecting monitors compatible with composite video signals, a proprietary serial connector to which printers and disk drives can be connected, another comb connector for connecting the Datassette (Commodore's tape drive), and finally a pair of 9-pin ports for connecting joysticks, mice, and optical pens.

Versions and expansions

During its long commercial life (a 12-year career!), the Commodore 64 underwent numerous revisions and changes, not only aesthetic ones. Starting with the technical side, it should be noted that Commodore, with a view to optimizing the production of the machine and pursuing a perpetual reduction in costs, continuously updated the computer's circuit board (fig. 7). This is reflected in the size and shape of the board housed inside the computer, but also in the integrated circuits themselves: for example, over time, the number of RAM chips, the layout of the integrated circuits, and the type of chipset installed have changed (for example, when new versions with an updated production process were released).

Apart from the electronics, there were also versions that sometimes affected the aesthetics or capabilities of the C64. The first special version was the MAX console, or Ultimax as it was known within the company, which went on sale in early 1982, even before the C64 itself, but lasted only a very short time. The Educator 64 was an attempt by Commodore in 1983 to erode the Apple II's market share in schools. Released with a case similar to that of the PET and with an integrated green phosphor monitor, the computer was unsuccessful, having to compete with the extensive software library of the former. Also in 1983, the Commodore SX-64 (fig. 9) was introduced, a



Fig. 9 - The Commodore SX-64, a "portable" version of the C64 (photo: Hedning - source: Wikimedia Commons)



Fig. 10 - The Commodore 64C, which echoes the style of the C128 (photo: Kausalkette - source: Wikimedia Commons)

C64 that echoed the style of the Osborne 1 and Kaypro II, i.e., a "portable" computer the size of a small suitcase. Compared to these systems, it was the first to offer a color monitor.

In 1986, following the style of the C128 introduced the year before, the C64C was introduced, a version with a lower, sleeker case in a light beige color (fig. 10). Functionally identical to the previous model, some peripherals were no longer compatible due to the different dimensions, such as the accelerator cards to be mounted internally or some external accessories such as the musical keyboards to be attached to the machine's keys. Apart from this, internally there was a switch to a new version of the chipset, which brought with it a problem with the SID: to correct a 'flaw' in the initial design, the SID lost its initial power and 'roughness' as well as some sound effects that, over the years, several musicians had learned to bring out, turning a flaw into an asset. While some prefer the cleaner sound, lovers of the old C64 are up in arms, as they can no longer hear certain sounds that the original SID was able to reproduce compared to the new one. Towards the end of the 1980s, Commodore produced a couple of special versions for the German market (the "ALDI" and the "G") for which the old case was dusted off: these were "hybrids" with the shape of the first C64 and the lighter plastic colors of the C64C.

In 1990, in a (misguided) attempt to (poorly) retrace the (erroneous) path of the MAX, the Commodore 64 Game System, abbreviated to 64 GS, was released, a C64 stripped of its keyboard and everything except the cartridge port and joystick ports. The C64 GS is basically the "console" version of the computer and can run its game cartridges as long as they do not require keyboard input, since there is no keyboard... Needless to say, this was yet another commercial flop.

Finally, we must mention the legendary Commodore 65. Created in 1991, it is a version halfway between the 8-bit C64 and the 16-bit Amiga: the project was then canceled, probably because the market was also moving away from 16-bit to the more powerful 32-bit. In 1994, after Commodore's bankruptcy, a few dozen C65 prototypes left the company and were put up for sale, becoming very rare collector's items over the years.

Games and programs

Needless to say, the computer's graphics and sound capabilities, as well as its enormous popularity, have always attracted software developers, so much so that in the mid-1980s, for any program to be successful, it only had to be released for the IBM PC and the Commodore 64. Many utility programs were also converted for the C64, such as Microsoft's MultiPlan spreadsheet or KoalaPainter, the drawing program offered in conjunction with the KoalaPad graphics tablet. A major boost to the spread of 'serious' programs came after the introduction of GEOS, a graphical operating system released by Berkley Software in 1986: originally developed specifically for the C64, GEOS offers a graphical environment very similar to that of the first Apple Macintosh and, being a true operating system, allows the execution of applications that can not only access all the machine's peripherals but also take advantage of the windowing environment. In addition to the integrated version, there were also more advanced versions of BASIC that made better use of the machine's capabilities, such as Simon's BASIC, which offered new commands for easier access to disks and for drawing in graphics mode, managing sprites, or generating music more easily. Remaining on the subject of programming languages, there was also a wide range of interpreters and compilers for the most popular languages of the time, such as assembly, Forth, C, Pascal, and FORTRAN.

However, it was games that best demonstrated the audio and graphics capabilities of the C64: in the second half of the 1980s, more than 60% of the software produced for this platform was games, so it is impossible to list them all, as there are thousands. According to some



Fig. 11 - Impossible Mission
(source: Mobygames.com)



Fig. 12 - Winter Games
(source: Mobygames.com)

estimates, there were around 6,000, and the total number of programs released, including application software, reached 10,000. It is also necessary to distinguish between games published in the US and those published in Europe: while the former were mainly released on diskette, as this was the most popular format in North America, in Europe games were mainly released on cassette, a cheaper format that was more widespread on the Old Continent. Therefore, many titles developed to use random data loading during gameplay were not released on tape, while not all titles released on cassette, mainly by European developers, made it to America. Apart from this, there are still some titles that are distinctive to the computer: "Impossible Mission" from 1984 (fig. 11), where you play the role of a special agent, is one of the first widely distributed games, often cited as one of the "must-have" games for the C64. Then there are the multi-event sports games published by Epyx, such as "Summer Games," "Winter Games" (fig. 12), "California Games," and "World Games." "Pitstop II" is worth mentioning because it is the first 2-player car racing title with split screen.

In the early 1990s, when the platform was nearing the end of its commercial life and the market was dominated by 16-bit systems, programmers, armed with the experience and knowledge of the machine accumulated in previous years, released some real gems that made the most of the C64's capabilities. Turrican I (fig. 13) and II were games released on the C64 in 1990 and 1991 respectively, which, given their quality, were so successful that they were converted for other platforms. Mayhem in Monsterland (fig. 14), from 1993, is universally known as the ultimate 8-bit game: the programmers exploited the C64 so thoroughly that the end result, a Super Mario-style platform game, looks like a game for a 16-bit system.

The audio capabilities of computers have also given rise to a new profession: digital composer of music for games. Among the musical artists who started out on the C64, we must mention Martin Galway, the first to use sampled music in his 1987 conversion of "Arkanoid," and Matt Gray, known for the famous musical themes of "The Last Ninja" series, games that, in addition to their music, are



Fig. 13 - Turrican
(source: Mobygames.com)



Fig. 14 - Mayhem in Monsterland
(source: Mobygames.com)



Fig. 15 - The Last Ninja 2 is remembered not only for its isometric view but also for its musical theme (source: Mobygames.com)

memorable for their isometric scenes (fig. 15). Rob Hubbard wrote many pieces of music for famous games, including "International Karate," several chapters of the "Monty" series, and "Commando," to name a few.

Since many games were released with protection systems, several hackers or groups of hackers emerged at that time who, to boast about having broken the protection of this or that software, inserted their signatures at the beginning of the loading process. These signatures became increasingly complex to show what they were capable of: from simple texts, they soon moved on to graphic animations accompanied by music. Over time, the animations became so elaborate and exploited the machine's hardware so thoroughly that they reached unbelievable levels of quality: from "crack-intros," the scenes at the beginning of cracked games, to the phenomenon of 'demoscene', i.e. crack-intros that become independent programs and do nothing more than show off the hacker's ability to push the machines they run on to their limits. This phenomenon remained active for years and, in recent times, thanks to renewed interest in our 8- and 16-bit systems, has regained ground and is back in vogue among enthusiasts of the time but also among newcomers (fig. 16).



Fig. 16 - "We Come in Peace," a demo by Censor Design & Fairlight from 2018 (source: Youtube.com)

Conclusions

We could go on for a long time, citing the myriad of peripherals created to enable the Commodore 64 to perform any type of task. I myself remember that, in the mid-1980s, I went to visit a friend whose parents had a computer connected to a modem and, using a program, received images from Meteosat satellites, processed them, and then displayed the weather situation in Italy in real time on the screen. Countless young people built their IT careers by learning the basics of programming on this computer, and those who didn't program still learned how to use computers thanks to the C64. Newsstands were flooded with magazines featuring programs to type into your computer, as well as collections of games to try out at home that new arcade game you saw at the bar. Yes, because despite the limitations of its BASIC, when you switched to assembly language and squeezed the machine, it produced things that other contemporary computers could only imagine. We all know how the story ended, with Commodore never managing to make the most of all the opportunities it had in its hands, from the Amiga to the C64 itself, achieving a fraction of what these systems could have potentially brought in economic terms if only they had thought to develop and evolve them intelligently instead of focusing on taking the chicken today and giving up on tomorrow's egg.

Cases of identity... - part two

by Marco Pistorio

Welcome back, dear readers of RetroMagazine World!

In the first part of this article, published in issue 20-EN (42-IT) of RetroMagazine World, I told you about my experience developing some assembly language programs for our beloved C64. I focused on the problem of how to accurately determine whether an executable program matches exactly, byte for byte, with one created later, even a few years later.
I talked about the possibility of determining the MD5 signature of the file in order to calculate its exact correspondence with another file.

This is a path that is usually easy to follow, or at least should be in most cases.

I also suggest that you always document your work, perhaps using notes within the source code itself, always indicating the exact date of compilation, the compiler you used, specifying its version number, any tools used (specifying their version numbers as well), and finally the operating system you used.

If you find yourself recompiling the source code for these projects, it will be easier to put all the 'pieces' together to obtain the same precise object file, and you can compare its MD5 signature with that of the object file created previously.

Unfortunately, despite these precautions, the result is not always guaranteed.

I told you about a situation that arose when I found myself recompiling an old intro that I published in 2016 on CSDB and, despite several attempts, I was unable to obtain the same .prg file as the one previously published.
It is likely that, in this particular circumstance, the reason is due to the fact that, at the time, I used a version of EXOMIZER that ran on LINUX systems.
I believe that the same data compression tool that runs on Windows behaves slightly differently internally, resulting in a slightly different .prg file and therefore a mismatched MD5 signature.
So, at the end of the previous part of this article, I left you with a question: "How can we tell if two executables are traceable to the same source code when the respective MD5 signatures of the two executable programs do not match?"

Some might think, "If the two executable programs behave in the same way... they must be the same!"
Is this an acceptable approximation for you? I was thinking more of an almost 'forensic' level analysis :D

How could this be done? First of all, it should be noted that you would need to have the exact source code available. You would need to recompile it and then analyze it appropriately.

This analysis should be done under the same conditions. Let me explain:

If the original program was initially decompressed at startup, its code should be examined immediately after decompression.

Why? Because we start from the idea that the MD5 signatures of the two files are different and that there is therefore a difference between the two files.
To observe them, we must start from the same initial conditions (the first decompressed, the second also decompressed).
We must try to understand WHERE they differ, in which areas of memory and to what extent.

Here, the discussion becomes more technical, and I would like to remind you that I am referring explicitly to work done for the Commodore 64.
The technique could also be used in other areas, but the examples and tools I will show you refer to the Commodore 64.

It is necessary to obtain a dump of the C64 memory immediately after the program has been launched and decompressed.

How? For this purpose, you could use this excellent (and free) tool: c64 65XE NES DEBUGGER (see figure 1)

https://sourceforge.net/projects/c64-debugger/files/

The procedure for obtaining a dump is as follows:
1) Launch the program
2) Press: CTRL+B "BREAKPOINTS"
3) Set a specific address where you want to block the execution of the program, in the "CPU PC" box, i.e., the Program Counter register (although you can use other options).
PLEASE NOTE: HERE, the program's source code may be useful in understanding WHERE it is best to set the breakpoint from which to save a complete image of the C64's memory.
4) Press CTRL+B again.

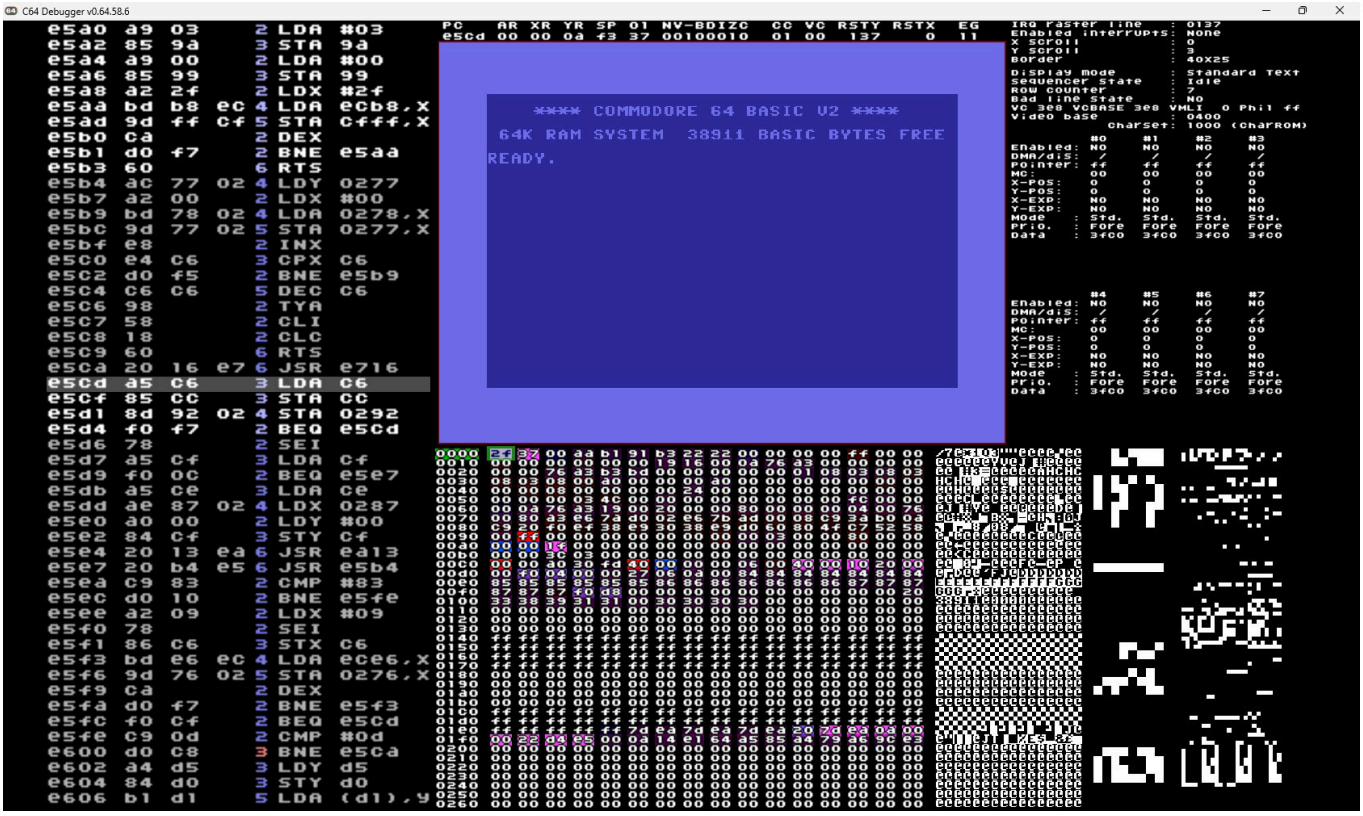


Fig. 1 - C64 Debugger just launched

5) Press CTRL+O and choose the .prg file to launch. If you have done everything correctly, the program will stop executing EXACTLY at the location you specified in step 3.
6) Press CTRL+U to save a complete dump of the C64 memory to a file on your PC.

Obviously, if you are comparing this first .prg file with a second one that you created later, you will have to perform the same operations a second time, this time specifying the new .prg file from which you want to obtain a dump that you will then compare.

At this point, our problem is reduced to comparing these two dump files. If they are very different, it means that they cannot be traced back to the same source code. If, on the other hand, the differences we find are minimal, the situation could be different.

At this stage, it is also important to examine the sources to determine which areas of memory are used, for example, to contain data and which contain assembly instructions.
If we do not know this information, we may not be able to determine, when finding a difference between the two dumps, whether that difference can be ignored or not.

The content of each dump file is not immediately readable. It is best to use a decompiler to make it more readable.

I often use my own personal decompiler, but there are several others available.

Among the many available, I can recommend this one:
Infiltrator Disassembler V1.0 (see Figure 2)
https://csdb.dk/release/?id=100129

Once you have processed the two .prg files and decompiled them, you can compare the two text files generated to determine the differences.

I recommend this excellent online tool (ref. figure 3):
https://www.diffchecker.com/text-compare/

Finally, we can observe the differences and try to evaluate them.
I did this work on the dump of my intro "The Awakening of the Force," which I published on CSDB in 2016 (see Figure 4), and compared it with the dump of a .prg file that I now obtain by compiling the same source code.

The two MD5 signatures, as already mentioned, are different. This implies that there are differences in memory.

This is a complete list of all the differences I get when comparing the two dumps using my tools:

```
$0023/$0023 :ERR|$0023: A3        |$0023: 08
$0049/$0049 :ERR|$0049: 04        |$0049: 08
$00a1/$00a1 :ERR|$00a1: 12        |$00a1: 13
```

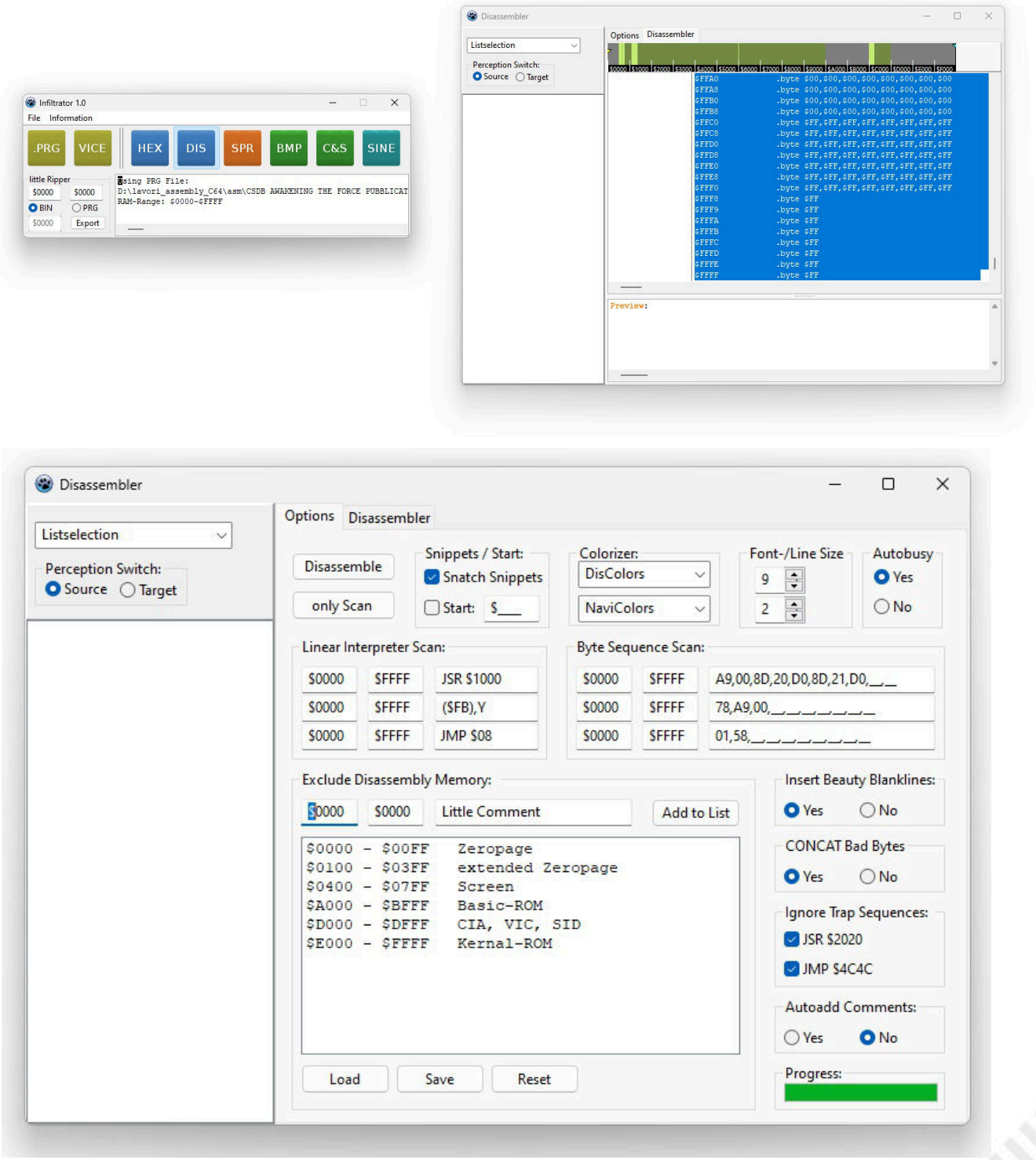


Fig. 2 - Infiltrator Disassembler in action!

```
$00a2/$00a2 :ERR|$00a2: C8       |$00a2: 3D 40 C0
$00a3/$XXXX :ERR|$00a3: 40       |UNAVAILABLE
$00a4/$XXXX :ERR|$00a4: C0 00    |UNAVAILABLE
$00c9/$00c9 :ERR|$00c9: 07       |$00c9: 05 03
$00ca/$XXXX :ERR|$00ca: 03       |UNAVAILABLE
$00cc/$00cc :ERR|$00cc: 01 02    |$00cc: 01 11
$00d1/$00d1 :ERR|$00d1: 80       |$00d1: E0 05
$00d2/$XXXX :ERR|$00d2: 06 00    |UNAVAILABLE
$00d6/$00d6 :ERR|$00d6: 10 0D    |$00d6: 0C
$00d8/$XXXX :ERR|$00d8: 00       |UNAVAILABLE
$00d9/$XXXX :ERR|$00d9: 84 84    |UNAVAILABLE
$00db/$XXXX :ERR|$00db: 84 84    |UNAVAILABLE
$00dd/$XXXX :ERR|$00dd: 84 84    |UNAVAILABLE
$00df/$XXXX :ERR|$00df: 84 85    |UNAVAILABLE
$00e1/$XXXX :ERR|$00e1: 85 85    |UNAVAILABLE
$00e3/$XXXX :ERR|$00e3: 85 85    |UNAVAILABLE
$00e5/$XXXX :ERR|$00e5: 85 86    |UNAVAILABLE
$00e7/$XXXX :ERR|$00e7: 86 86    |UNAVAILABLE
$00e9/$XXXX :ERR|$00e9: 86 86    |UNAVAILABLE
$00eb/$XXXX :ERR|$00eb: 86 86    |UNAVAILABLE
$00ed/$XXXX :ERR|$00ed: 87       |UNAVAILABLE
$00f3/$00f3 :ERR|$00f3: 80       |$00f3: E0 D9
$00f4/$XXXX :ERR|$00f4: DA       |UNAVAILABLE
$00f5/$00f5 :ERR|$00f5: 81 EB    |$00f5: C2
$01f7/$01f7 :ERR|$01f7: 00       |$01f7: FF
$0202/$0202 :ERR|$0202: 4E 00 00 |$0202: 4E 00 2C
$0286/$0286 :ERR|$0286: 0E 0E 04 |$0286: 0E 6E 04
$028c/$028c :ERR|$028c: 10 00    |$028c: 10 01
```

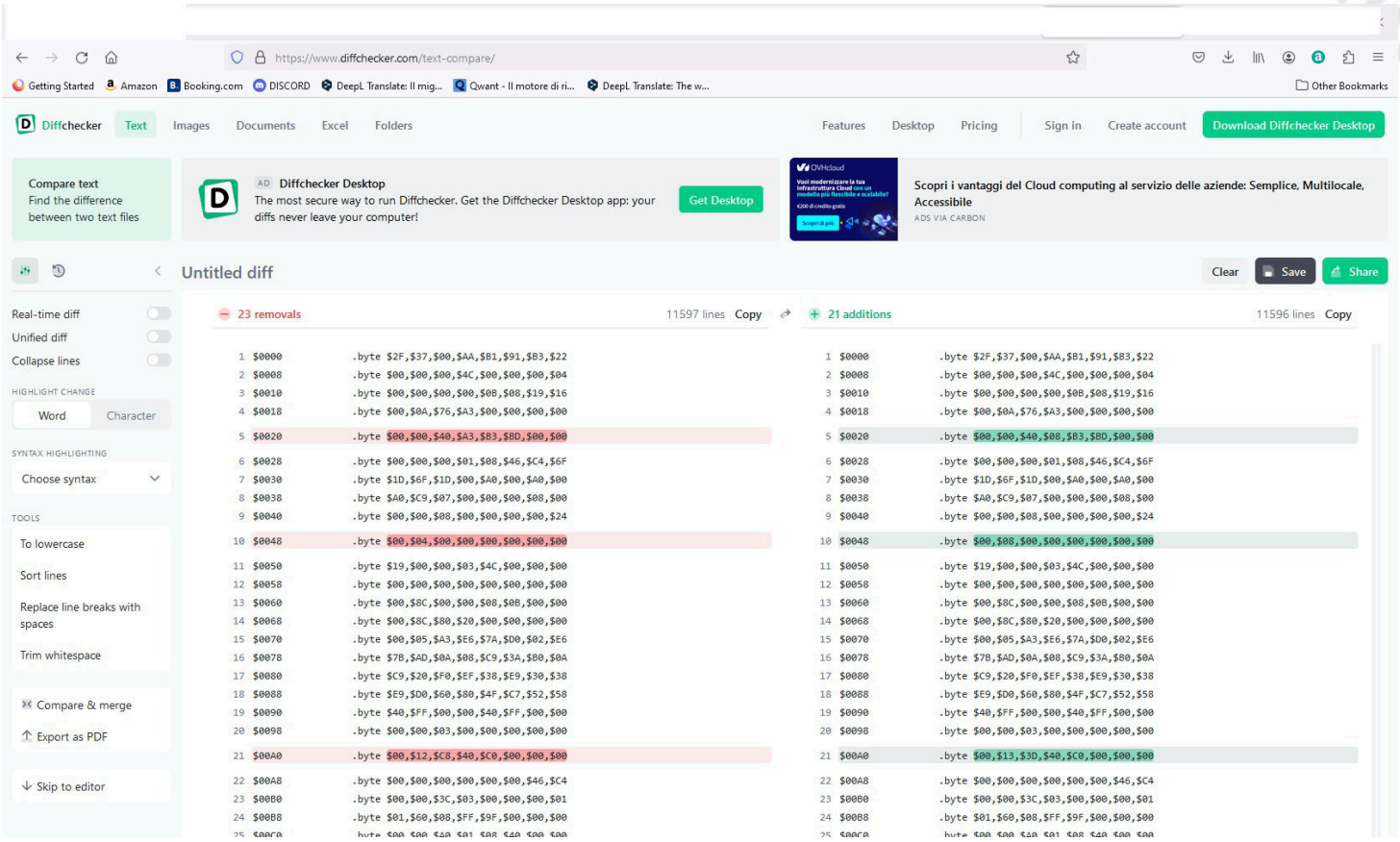


Fig. 3 - Result of the comparison between the two decompiled dump files

```
$028e/$028e :ERR|$028e: 00       |$028e: 01 48
$028f/$XXXX :ERR|$028f: 48       |UNAVAILABLE
$05b8/$05b8 :ERR|$05b8: 0C       |$05b8: 12
$05b9/$05b9 :ERR|$05b9: 09 13    |$05b9: 15 0E
$05bb/$05bb :ERR|$05bb: 14       |$05bb: 20 20 20
$05bc/$XXXX :ERR|$05bc: 20 20 20 |UNAVAILABLE
$05bf/$XXXX :ERR|$05bf: 20 20 20 |UNAVAILABLE
$05c2/$XXXX :ERR|$05c2: 20 20 20 |UNAVAILABLE
$05c5/$XXXX :ERR|$05c5: 20 20 20 |UNAVAILABLE
$05c8/$XXXX :ERR|$05c8: 20 20 20 |UNAVAILABLE
$05cb/$XXXX :ERR|$05cb: 20 20 20 |UNAVAILABLE
$05ce/$XXXX :ERR|$05ce: 20 20 20 |UNAVAILABLE
$05d1/$XXXX :ERR|$05d1: 20 20 20 |UNAVAILABLE
$05d4/$XXXX :ERR|$05d4: 20 20 20 |UNAVAILABLE
$05d7/$XXXX :ERR|$05d7: 20 20 20 |UNAVAILABLE
$05da/$XXXX :ERR|$05da: 20 20 20 |UNAVAILABLE
$05dd/$XXXX :ERR|$05dd: 20 20 20 |UNAVAILABLE
$05e0/$XXXX :ERR|$05e0: 20 20 20 |UNAVAILABLE
$05e3/$XXXX :ERR|$05e3: 20 20 20 |UNAVAILABLE
$05e6/$XXXX :ERR|$05e6: 20 20 20 |UNAVAILABLE
$05e9/$XXXX :ERR|$05e9: 20 20 20 |UNAVAILABLE
$05ec/$XXXX :ERR|$05ec: 20 20 20 |UNAVAILABLE
$05ef/$XXXX :ERR|$05ef: 20 20 20 |UNAVAILABLE
$05f2/$XXXX :ERR|$05f2: 20 20 20 |UNAVAILABLE
$05f5/$XXXX :ERR|$05f5: 20 20 20 |UNAVAILABLE
$05f8/$XXXX :ERR|$05f8: 20 20 20 |UNAVAILABLE
$05fb/$XXXX :ERR|$05fb: 20 20 20 |UNAVAILABLE
$05fe/$XXXX :ERR|$05fe: 20 20 20 |UNAVAILABLE
$0601/$XXXX :ERR|$0601: 20 20 20 |UNAVAILABLE
$0604/$XXXX :ERR|$0604: 20 20 20 |UNAVAILABLE
$0607/$XXXX :ERR|$0607: 20 31 39 |UNAVAILABLE
$060a/$XXXX :ERR|$060a: 39 33 20 |UNAVAILABLE
$060d/$XXXX :ERR|$060d: 13       |UNAVAILABLE
$060e/$XXXX :ERR|$060e: 19 13 32 |UNAVAILABLE
$0611/$XXXX :ERR|$0611: 30 35    |UNAVAILABLE
$0613/$XXXX :ERR|$0613: 39 20 20 |UNAVAILABLE
$0616/$XXXX :ERR|$0616: 20 20 20 |UNAVAILABLE
$0619/$XXXX :ERR|$0619: 20 20 20 |UNAVAILABLE
$061c/$XXXX :ERR|$061c: 20 20 20 |UNAVAILABLE
$061f/$XXXX :ERR|$061f: 20 20 20 |UNAVAILABLE
$0622/$XXXX :ERR|$0622: 20 20 20 |UNAVAILABLE
$0625/$XXXX :ERR|$0625: 20 20 20 |UNAVAILABLE
$0628/$XXXX :ERR|$0628: 20 20 20 |UNAVAILABLE
$062b/$XXXX :ERR|$062b: 20 20 20 |UNAVAILABLE
$062e/$XXXX :ERR|$062e: 20 20 12 |UNAVAILABLE
$0631/$XXXX :ERR|$0631: 05 01    |UNAVAILABLE
$0633/$0633 :ERR|$0633: 04       |$0633: 20 20 20
$0634/$XXXX :ERR|$0634: 19 2E 20 |UNAVAILABLE
$0637/$XXXX :ERR|$0637: 20 20 20 |UNAVAILABLE
$063a/$XXXX :ERR|$063a: 20 20 20 |UNAVAILABLE
$063d/$XXXX :ERR|$063d: 20 20 20 |UNAVAILABLE
$0640/$XXXX :ERR|$0640: 20 20 20 |UNAVAILABLE
$0643/$XXXX :ERR|$0643: 20 20 20 |UNAVAILABLE
$0646/$XXXX :ERR|$0646: 20 20 20 |UNAVAILABLE
$0649/$XXXX :ERR|$0649: 20 20 20 |UNAVAILABLE
$064c/$XXXX :ERR|$064c: 20 20 20 |UNAVAILABLE
$064f/$XXXX :ERR|$064f: 20 20 20 |UNAVAILABLE
$0652/$XXXX :ERR|$0652: 20 20 20 |UNAVAILABLE
$0655/$XXXX :ERR|$0655: 20 20 20 |UNAVAILABLE
$0658/$XXXX :ERR|$0658: 12       |UNAVAILABLE
$0659/$XXXX :ERR|$0659: 15 0E    |UNAVAILABLE
$065b/$XXXX :ERR|$065b: 20 20 20 |UNAVAILABLE
$065e/$XXXX :ERR|$065e: 20 20 20 |UNAVAILABLE
$0661/$XXXX :ERR|$0661: 20 20 20 |UNAVAILABLE
$0664/$XXXX :ERR|$0664: 20 20 20 |UNAVAILABLE
$0667/$XXXX :ERR|$0667: 20 20 20 |UNAVAILABLE
$066a/$XXXX :ERR|$066a: 20 20 20 |UNAVAILABLE
$066d/$XXXX :ERR|$066d: 20 20 20 |UNAVAILABLE
$0670/$XXXX :ERR|$0670: 20 20 20 |UNAVAILABLE
$0673/$XXXX :ERR|$0673: 20 20 20 |UNAVAILABLE
$0676/$XXXX :ERR|$0676: 20 20 20 |UNAVAILABLE
$0679/$XXXX :ERR|$0679: 20 20 20 |UNAVAILABLE
$067c/$XXXX :ERR|$067c: 20 20 20 |UNAVAILABLE
$067f/$XXXX :ERR|$067f: 20 20 20 |UNAVAILABLE
$0682/$XXXX :ERR|$0682: 20 20 20 |UNAVAILABLE
$0685/$XXXX :ERR|$0685: 20 20 20 |UNAVAILABLE
$0688/$XXXX :ERR|$0688: 20 20 20 |UNAVAILABLE
$068b/$XXXX :ERR|$068b: 20 20 20 |UNAVAILABLE
$068e/$XXXX :ERR|$068e: 20 20 20 |UNAVAILABLE
$0691/$XXXX :ERR|$0691: 20 20 20 |UNAVAILABLE
```

```
$0694/$XXXX :ERR|$0694: 20 20 20 |UNAVAILABLE
$0697/$XXXX :ERR|$0697: 20 20 20 |UNAVAILABLE
$069a/$XXXX :ERR|$069a: 20 20 20 |UNAVAILABLE
$069d/$XXXX :ERR|$069d: 20 20 20 |UNAVAILABLE
$06a0/$XXXX :ERR|$06a0: 20 20 20 |UNAVAILABLE
$06a3/$XXXX :ERR|$06a3: 20 20 20 |UNAVAILABLE
$06a6/$XXXX :ERR|$06a6: 20 20 20 |UNAVAILABLE
$06a9/$XXXX :ERR|$06a9: 20 20 20 |UNAVAILABLE
$06ac/$XXXX :ERR|$06ac: 20 20 20 |UNAVAILABLE
$06af/$XXXX :ERR|$06af: 20 20 20 |UNAVAILABLE
$06b2/$XXXX :ERR|$06b2: 20 20 20 |UNAVAILABLE
$06b5/$XXXX :ERR|$06b5: 20 20 20 |UNAVAILABLE
$06b8/$XXXX :ERR|$06b8: 20 20 20 |UNAVAILABLE
$06bb/$XXXX :ERR|$06bb: 20 20 20 |UNAVAILABLE
$06be/$XXXX :ERR|$06be: 20 20 20 |UNAVAILABLE
$06c1/$XXXX :ERR|$06c1: 20 20 20 |UNAVAILABLE
$06c4/$XXXX :ERR|$06c4: 20 20 20 |UNAVAILABLE
$06c7/$XXXX :ERR|$06c7: 20 20 20 |UNAVAILABLE
$06ca/$XXXX :ERR|$06ca: 20 20 20 |UNAVAILABLE
$06cd/$XXXX :ERR|$06cd: 20 20 20 |UNAVAILABLE
$06d0/$XXXX :ERR|$06d0: 20 20 20 |UNAVAILABLE
$06d3/$XXXX :ERR|$06d3: 20 20 20 |UNAVAILABLE
$06d6/$XXXX :ERR|$06d6: 20 20 20 |UNAVAILABLE
$06d9/$XXXX :ERR|$06d9: 20 20 20 |UNAVAILABLE
$06dc/$XXXX :ERR|$06dc: 20 20 20 |UNAVAILABLE
$06df/$XXXX :ERR|$06df: 20 20 20 |UNAVAILABLE
$06e2/$XXXX :ERR|$06e2: 20 20 20 |UNAVAILABLE
$06e5/$XXXX :ERR|$06e5: 20 20 20 |UNAVAILABLE
$06e8/$XXXX :ERR|$06e8: 20 20 20 |UNAVAILABLE
$06eb/$XXXX :ERR|$06eb: 20 20 20 |UNAVAILABLE
$06ee/$XXXX :ERR|$06ee: 20 20 20 |UNAVAILABLE
$06f1/$XXXX :ERR|$06f1: 20 20 20 |UNAVAILABLE
$06f4/$XXXX :ERR|$06f4: 20 20 20 |UNAVAILABLE
$06f7/$XXXX :ERR|$06f7: 20 20 20 |UNAVAILABLE
$06fa/$XXXX :ERR|$06fa: 20 20 20 |UNAVAILABLE
$06fd/$XXXX :ERR|$06fd: 20 20 20 |UNAVAILABLE
$0700/$XXXX :ERR|$0700: 20 20 20 |UNAVAILABLE
$0703/$XXXX :ERR|$0703: 20 20 20 |UNAVAILABLE
$0706/$XXXX :ERR|$0706: 20 20 20 |UNAVAILABLE
$0709/$XXXX :ERR|$0709: 20 20 20 |UNAVAILABLE
$070c/$XXXX :ERR|$070c: 20 20 20 |UNAVAILABLE
$070f/$XXXX :ERR|$070f: 20 20 20 |UNAVAILABLE
$0712/$XXXX :ERR|$0712: 20 20 20 |UNAVAILABLE
$0715/$XXXX :ERR|$0715: 20 20 20 |UNAVAILABLE
$0718/$XXXX :ERR|$0718: 20 20 20 |UNAVAILABLE
$071b/$XXXX :ERR|$071b: 20 20 20 |UNAVAILABLE
$071e/$XXXX :ERR|$071e: 20 20 54 |UNAVAILABLE
$cffa/$cffa :ERR|$cffa: 09 59    |$cffa: 0D 59 14
$cffc/$XXXX :ERR|$cffc: 14       |UNAVAILABLE
```

From my examination, I found that the differences are essentially contained in memory areas not directly connected to the intro but, probably, to the exomization process used, i.e., in some locations in RAM from page 0 to page 7.

I also detected a few different bytes in the area at addresses $CFFA-$CFFC which, from examining the source code, I determined to be designated to contain some variables that I used in my project.

In conclusion, in this specific case, I can safely say that the two programs can be traced back to the same original source code.

I am aware that this article is "tough," very technical, and therefore...not for everyone.

However, I hope I have managed to clearly illustrate all the necessary information and useful "tools of the trade" to readers interested in these issues.

It is clear that an analysis of more complex procedures requires more time and attention. Furthermore, some techniques, such as overwriting opcodes in memory during program execution, would not help those who wish to use this analysis method.

Anyone wishing to learn more is welcome to write to the RetroMagazine World editorial staff at the following email address: retromagazine.redazione@gmail.com
I will be happy to respond, of course.

Best regards to all, and see you next time!



Fig. 4 - The intro that was the subject of my analysis, which was the first intro I published on CSDB, "The Awakening of the Force," from 2016.

https://csdb.dk/release/?id=150842

The "Mini-Lotto" game (for C64)

by Eugenio Rapella

In this article, I propose a variation of a program that I presented in a book many years ago. It is little more than an excuse to work a little with our C64 in both Basic and Assembly.
The conclusions we will reach through our simplified "Lotto Game" remain valid for the normal game (freeing us from combinatorial complications).

Here is the story:
many citizens of Bingotown play the Mini-Lotto every week. From Monday to Friday, it is possible to bet on one of the ten numbers 0,1,2,..,9. On Saturday, a draw is held and those who have played the number drawn win. The amount for a bet is fixed and rather high; normally, a person cannot play more than one number per week.
Three friends, Alfio, Bruno, and Carlo, are among the most avid players: every week they go to the ticket office.
Alfio argues that the best strategy is to always bet on the same number because, sooner or later, it is bound to come up. Since he considers four to be his "lucky number" (Alfio was born on April 4), he always bets on that number and no other.
Bruno keeps an up-to-date "delay table" where he associates each number with the number of weeks since it last came up. When he goes to the lottery office, he plays the number with the longest delay: if a number hasn't come up for a while, well, its turn is coming!
Carlo argues that this is all nonsense and that playing one number or another makes no difference. With the help of a well-shuffled deck of cards, he randomly chooses the number to bet on each week.
What can we say? Which strategy is the most promising? We'll talk about that a little later. For now, let's put our C64 to work: it will simulate a number of draws and keep track of the three friends' winnings.
I will propose three programs: the first (in Basic) shows the situation in detail as the draws take place. It is intended for those who want to follow how things are going in real time. With all its PRINTS, however, it is quite... slow.
The second (also in Basic) is identical to the first but without the intermediate PRINT statements. It only gives us the final results but is, of course, faster than the first.
The third is identical to the second, but it is in Assembly and is... incredibly fast. I am not an expert in Assembly, so my program could probably be improved, but my goal is, let's say, educational. A little experiment: a code (full of comments) that I wrote based almost exclusively on a "Vocabulary of the 6510 Chip" that I found on the Internet.

Here is the listing for the first program:

```
100 poke 53280,6:poke 53281,6:print chr$(5):x=rnd(-ti):dim
r(9),f(9):pa=4
110 ra$=".. perso...": v$=chr$(18)+" vinto !! "+chr$(146)
115 for nl=1to39:s$=s$+chr$(32):next
120 print chr$(147):input"quante simulazioni ";ns:print chr$(147)
130 for k=1 to ns: m=0:x=int(rnd(1)*10):f(x)=f(x)+1:w$=""
140 print chr$(19):print"simul n.";k;"   su";ns
150 print"numero uscito:";x:print
160 print"-----------------------------------"
170 print"alfio ha puntato sul numero ";pa:rr$=ra$
180 if x=pa then rr$=v$:ta=ta+1
190 print"alfio ha";rr$:print"tot alfio";ta:print"% alfio";100*ta/k
200 print:print"-----------------------------------":m=0
210 print
220 print s$
230 for j=0 to 9:w$=w$+str$(r(j)):if r(j)>m then m=r(j):pb=j
240 next j:print chr$(145)+w$
250 print:print"bruno ha puntato sul numero ";pb:rr$=ra$
260 if x=pb then rr$=v$:tb=tb+1
270 print"bruno ha";rr$:print"tot bruno";tb:print"% bruno";100*tb/k
280 for j=0 to 9:r(j)=r(j)+1:next j
290 r(x)=0
300 pc=int(rnd(1)*10)
310 print:print"-----------------------------------"
320 print"carlo ha puntato sul numero ";pc:rr$=ra$
330 if x=pc then rr$=v$:tc=tc+1
340 print"carlo ha";rr$:print"tot carlo";tc:print"% carlo";100*tc/k
350 next k
360 print chr$(147):print"sono state eseguite ";ns;"simulazioni"
370 print:print"totale di alfio:";ta
380 print"% alfio:";100*ta/ns
390 print:print"totale di bruno:";tb
400 print"% bruno:";100*tb/ns
410 print:print"totale di carlo:";tc
420 print"% carlo:";100*tc/ns
430 print:for k=0 to 9:print"freq. di";k;">";f(k);" in %:";100*f(k)/
ns:next
```

At 100, the color combination (border, background, font) is chosen, the random number generator is initialized, and the two vectors R and F are sized, both with a size of 10. For each I from 0 to 9, R(I) will contain the weeks of delay for number I (for Bruno's use) and F(I) will contain the frequency of number I's appearance. PA is the variable that contains Aldo's bet, which, as we know, is always number 4.
At 110, a couple of alphanumeric constants are

defined that we will need, and at 120, the number of simulations to be performed (NS) is requested. The screen is cleared and... let's go!
The group of instructions from 130 (FOR K...) to 350 (NEXT K) is executed for each simulation.
At 130, X is assigned the number drawn and the "frequency of X" is immediately updated.
Each individual simulation involves printing an entire screen. At 140, PRINT CHR$(19) returns the cursor to the upper left corner of the screen without clearing it. This means that as the simulations change (about one per second), only the values that actually change are modified on the screen. The text that does not need to change remains fixed, allowing the screen to be read (if you try replacing CHR$(19) with CHR$(147), you will see the difference).
The simulation number (K) and the number that came out (X) are then printed.

We now move on to defining the bets of the three friends and checking any winnings.

For Alfio, it's straightforward: if X=PA (PA=Alfio's bet=4), Alfio's total winnings (TA) are increased, the message ("won"/"lost") is updated, TA is printed, and Alfio's win percentage up to that point (100*TA/K) is printed.

To choose what to bet, Bruno must check which number has the longest delay. This is handled by instructions 230 and 240: at the end of the FOR-NEXT cycle, the variable PB will contain the number (from 0 to 9) associated with the maximum delay, and the variable M will contain the amount of that delay (the elements of R(..) are reviewed: as soon as R(J) exceeds the current M, M and PB are updated).
Within the FOR-NEXT cycle, the string W$ is also constructed, which contains the sequence of delays; W$ is then printed (CHR$(145) pushes the cursor up; in this way, the printout overlaps the string from the previous simulation, replacing it).
Bruno then bets on PB; if X=PB (inst. 260), Bruno's total winnings are increased. Instructions 260-270 (relating to Bruno) have the same effect as 180-190 (relating to Alfio).
At 280, the delay of all numbers is increased by one unit except for the delay of the drawn number, which is naturally reset to zero (inst. 290).

Now it's Carlo's turn. We know that Carlo chooses the number to play at random, so the C64 generates it at 300. Instructions 310-340 are similar to those for the other two friends.
Instruction 360 is only reached when all simulations have been performed. For each of the friends, the number of winning plays and the relative percentages are printed.
Finally, the frequencies (absolute and percentage) of the various numbers drawn are printed.

Fig. 1 shows a screenshot during processing (with the delay table used by Bruno); Fig. 2 shows the final screenshot.

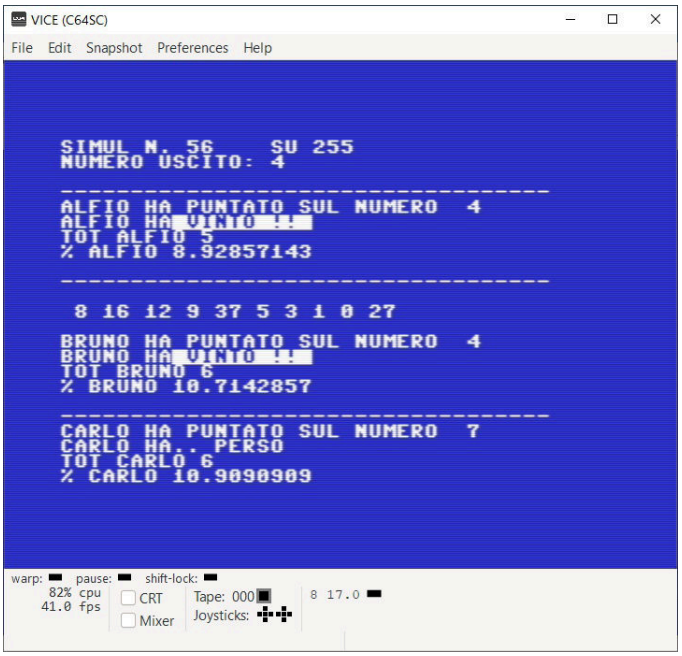


Fig. 1

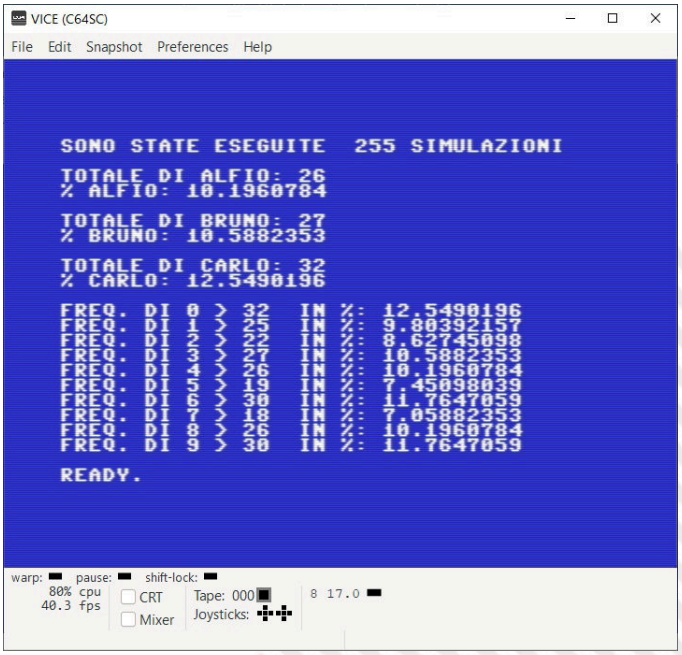


Fig. 2

The simulations confirm that none of the strategies adopted by the friends is preferable to the others.

Since the probability of guessing the exact number is 10%, the percentage frequencies will also approach this value.

It is true that if you always play the same number, or the number with the longest delay, sooner or later it will come up, but this is simply due to the fact that you are playing (any number): it is unlikely that you will always get it wrong, and sooner or later you will win.

Several newspapers report the numbers drawn in the "Gioco del Lotto"; a few years ago, the numbers with the longest delay were also reported. Only recently, at the newsstand, did I discover the existence of "Settimana Lotto": 48 pages (large format) full of statistics, frequencies, delays, historical data ... of dubious usefulness! (See Fig. 3 and Fig. 4 for a couple of examples).

```
100 poke 53280,2:poke 53281,2:print chr$(5):x=rnd(-ti):dim
r(9),f(9):pa=4
110 print chr$(147):input"quante simulazioni ";ns:print"o.k.
procedo"
120 for k=1 to ns: m=0:x=int(rnd(1)*10):f(x)=f(x)+1
130 if x=pa then ta=ta+1
140 m=0
150 for j=0 to 9:if r(j)>m then m=r(j):pb=j
160 next j
170 if x=pb then tb=tb+1
180 for j=0 to 9:r(j)=r(j)+1:next j
190 r(x)=0
200 pc=int(rnd(1)*10)
210 if x=pc then tc=tc+1
220 next k
230 print chr$(147):print"sono state eseguite ";ns;"simulazioni"
240 print:print"totale di alfio:";ta
250 print"% alfio:";100*ta/ns
260 print:print"totale di bruno:";tb
270 print"% bruno:";100*tb/ns
280 print:print"totale di carlo:";tc
290 print"% carlo:";100*tc/ns
300 print:for k=0 to 9:print"freq. di";k;">";f(k);" in
%:";100*f(k)/ns:next
```

We know that when working in Basic, our Commodore is not a "high-speed train": it takes about five and a half minutes to run 255 simulations!

If we are only interested in the final results, here is the "cleaned up" program that runs 255 simulations in 1 minute and 44 seconds (a time saving of 68%).

In writing these programs, I used CBM Prg Studio (v. 4.2.1) with the VICE emulator (v. 3.8): the times I reported refer to this situation.

I tried compiling this program with "Blitz Compiler": the compiled version runs 255 simulations in 34 seconds.

However, if we want "Formula One" performance, we have to resort to Assembly.

Let's translate the contents of FOR-NEXT (instructions from 120 to 220) of the previous program into Assembly; the final printout will remain in Basic.



Fig. 3



Fig. 4

The memory locations starting from $C000 can host the code; for the variables, I will use locations $033C - $03FB (828-1019), which correspond to the "cassette

recorder buffer" and are therefore available. I decided to leave the number of simulations fixed at 255; here's how I organized it:

* The Y register is used exclusively as a simulation counter. It is initially loaded with the value 255 (ldy #$ff) and is decremented (dey) at the end of each simulation. When it reaches zero, the program ends.
* The X register is also used as a counter in several places: to reset certain memory locations, to scroll through the "frequency vector," the "delay vector," etc.
* The ten locations from 828 ($033C) to 837 ($345) will contain the absolute frequencies of each of the numbers from 0 to 9. If, at a given moment, location 828 contains the number 12, it means that, up to that point, the number 0 has been drawn 12 times. Note that, since there are 255 simulations, here, as in all other cases, the numbers to be stored do not exceed this value and one byte is sufficient to contain them.
* The ten memory locations from 838 ($346) to 847 ($34F) are reserved for the "delay vector."
* Locations 848 ($350), 849 ($351), and 850 ($352) contain TA, TB, and TC, respectively: Alfio's total winnings, etc.
* Location 851 ($353) contains the number played by Bruno; 852 ($354) contains the maximum value of the "delay vector."
* Location 853 ($355) is reserved for the number played by Carlo; 854 ($356) contains the number drawn.

To generate a random number, I used an Assembly code that I found on the Internet; I had to resort to Marco Pistorio's expertise to unravel its mysteries.
Marco provided me with a detailed explanation (which deserves an article of its own) that I will try to summarize.
We know that our C64 is equipped with the wonderful SID (Sound Interface Device), thanks to which many games are accompanied by fantastic soundtracks (go and listen to the one for "Monty On The Run"). The SID can also generate a "noise" obtained thanks to a register (LFSR = Linear Feedback Shift Register) whose signal, when manipulated appropriately, generates a pseudo-random sequence of bits. To do this, you need to set the frequency of "voice 3" in locations $D40E and $D40F:

```
lda #$ff; set voice #3 frequency
sta $d40e
sta $d40f
```

and set bit #7 of the register that controls voice #3 to 1, setting the other bits of this register to zero:

```
lda #$80        ; set voice #3 control register
sta $d412       ; $80 = %10000000 (binary).
```

At this point, every time you read the contents of location $D41B (oscillator voice 3), you will obtain a pseudo-random value in the range 0, 255.
Here is the code:

```
*=$c000             ; sys 49152
       lda #$ff     ; ********************
       sta $d40e    ; * inizializzazione
       sta $d40f    ; * del generatore di
       lda #$80     ; * di numeri casuali
       sta $d412    ; ********************

       ldx #$1b     ; le locazioni da
       lda #$0      ; 828 a 855 che
alfa1  sta $33c,x   ; verranno utilizzate
       dex          ; per le variabili
       bne alfa1    ; vengono preventivamente
       sta $33c     ; azzerate

       ldy #$ff     ; si eseguono 255 simulazioni
beta
       jsr teta     ; vai al sorteggio...
       sta $356     ; qui va il numero uscito
       tax          ; che va anche nel registro X
       inc $033c,x  ; aggiorna tot. vett. NUMERI USCITI
       cmp #$4      ; è uscito il 4?
       bne istr0
       inc $350     ; se sì, aggiorna il tot. di Alfio

; Nelle istruzioni che seguono viene determinato qual è
; il massimo valore nel VETTORE DEI RITARDI e a quale
; indice X corrisponde(sarà il numero che giocherà Bruno)
istr0  ldx #$09
gamma  lda $354     ; qui va l'attuale MASSIMO (M)
       cmp $346,x   ; confr con l'elemento del vett
RITARDI
       bcs istr1    ; se R(X)<=M vai a istr1
       lda $346,x   ; ... altrimenti
       sta $354     ; metti R(X) in M
       txa
       sta $353     ; e X in $353
istr1  dex
       bpl gamma    ; se non abbiamo finito si torna a
gamma

; Nelle istruzioni che seguono, il numero scelto da Bruno
; (quello con maggior ritardo), in $353, viene confrontato
; col numero uscito, in $356. Se sono uguali, il totale
delle
```

```
; vittorie di Bruno, in $351, viene incrementato.
        lda $353
        cmp $356
        bne istr2
        inc $351

; Nelle istruzioni che seguono viene aggiornato il
VETTORE DEI
; RITARDI: Si addiziona 1 a tutte le componenti, poi
viene
; azzerato il ritardo del numero uscito.
istr2   ldx #$09
istr3   inc $346,x
        dex
        bpl istr3
        ldx $356
        lda #$0
        sta $346,x

; Nelle istruzioni che seguono viene gestita la
situazione di
; Carlo: il numero che gioca è scelto a caso (jsr teta)
e, se
; coincide con il numero uscito, il suo totale (in $352)
viene
; incrementato.
        jsr teta
        cmp $356
        bne istr4
        inc $352

istr4   dey
        bne beta     ; se non hai finito, un'altra
simulazione.
        rts

;** subroutine teta: genera in A
;** una cifra casuale tra 0 e 9
teta
alfa    lda $d41b    ; in A un numero tra 0 e 255
        and #$0f     ; ora in A un numero tra 0 e 15
        cmp #$0a     ; è maggiore o uguale a 10 ?
        bcs alfa     ; se sì, torna ad alfa
        rts          ; in A c'è una cifra tra 0 e 9
```

The Assembly program performs the 255 simulations, but does not display the results.

One of the CBM Prg Studio tools is "Generate BASIC loader": the Assembly program is transformed into a series of DATA that the generated Basic program will read and insert into locations starting from the SA (Starting Address) corresponding to *=$C000 in the Assembly code. The SYS SA instruction will send everything to be executed.

Now we can complete the generated Basic program with the print instructions; in fact, we can do more.

Since the 255 simulations are performed very quickly, we can afford the luxury of having the C64 execute more "blocks" of 255 simulations each and print the overall totals.

Here is the code:

```
1000 REM GENERATED ML LOADER
1040 poke 53280,4:poke 53281,4:print
chr$(5),chr$(147):dim f(9)
1120 SA = 49152
1160 FOR N = 0 TO 119
1200 READ A%:POKE SA+N,A%: NEXT N
1240 input"quanti gruppi di 255 simulazioni > ";g
1280 for g1=1 to g
1360 SYS SA
1400 ta=ta+peek(848)
1440 tb=tb+peek(849)
1480 tc=tc+peek(850)
1520 for k=0 to 9:f(k)=f(k)+peek(828+k):next
1560 next g1
1570 tgg=255*g
1600 print chr$(147):print"sono state
eseguite";tgg;"simulazioni"
1640 print:print"totale di alfio:";ta
1680 print"% alfio:";100*ta/tgg
1720 print:print"totale di bruno:";tb
1760 print"% bruno:";100*tb/tgg
1800 print:print"totale di carlo:";tc
1840 print"% carlo:";100*tc/tgg:print
1920 for k=0 to 9:print"freq. di";k;">";f(k);" in
%:";100*f(k)/tgg:next
1960 END
3060 DATA 169,255,141,14,212,141,15,212
3070 DATA 169,128,141,18,212,162,27,169
3080 DATA 0,157,60,3,202,208,250,141
3090 DATA 60,3,160,255,32,110,192,141
3100 DATA 86,3,170,254,60,3,201,4
3110 DATA 208,3,238,80,3,162,9,173
3120 DATA 84,3,221,70,3,176,10,189
3130 DATA 70,3,141,84,3,138,141,83
3140 DATA 3,202,16,235,173,83,3,205
3150 DATA 86,3,208,3,238,81,3,162
3160 DATA 9,254,70,3,202,16,250,174
3170 DATA 86,3,169,0,157,70,3,32
3180 DATA 110,192,205,86,3,208,3,238
3190 DATA 82,3,136,208,175,96,173,27
3200 DATA 212,41,15,201,10,176,247,96
```

The FOR-NEXT (1160-1200) loads the Assembly code into memory; at 1240, it asks how many "blocks" are to be executed. For each block (FOR-NEXT, 1280-1560), the Assembly code (SYS SA) is executed and the totals of the three friends' winnings are updated by taking the values from their reserved locations (1440-1480), the same happens for the absolute frequencies at 1520.
At the end, the results are printed; the total number of simulations performed is, of course, equal to 255 for the number of blocks requested (TGG inst. 1570).
If, given the RUN, you choose to run only one group, you perform 255 simulations and, as you will see, the result is almost immediate (Fig. 5).



Fig. 5

So, to recap, to run the 255 simulations, the first program took 5 minutes and 30 seconds (but there were a lot of PRINTS and the aim was not speed, but to show how the situation evolved); the second took 1 minute and 44 seconds; the second compiled took 34 seconds and the last one... 1 second!

The Assembly program allows us to... exaggerate: if, at RUN, we enter the request for 67 groups, we perform 17,085 plays and, after about 35 seconds, we obtain (Fig. 6):

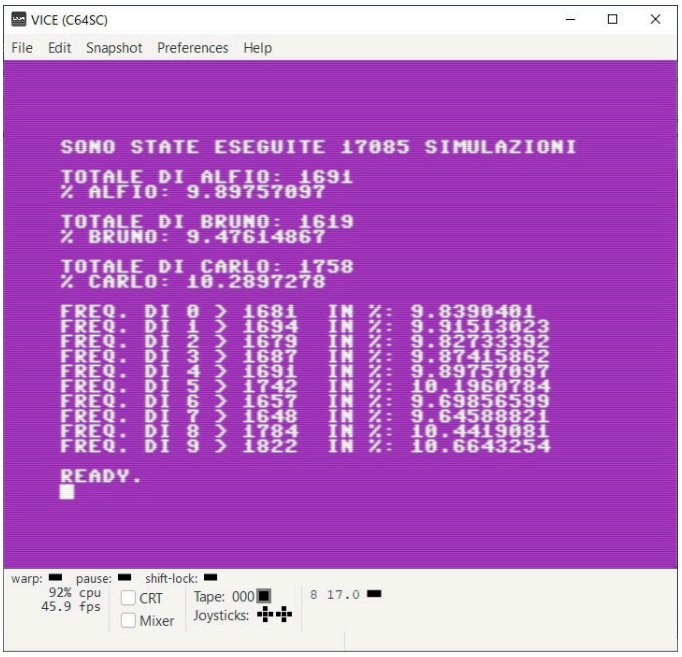


Fig. 6

... as the number of trials increases, the frequencies tend to settle around 10%, confirming once again that the strategies of the three friends are equivalent.

Running 17,085 simulations with the other programs is not recommended: the "cleaned up" one compiled with Blitz Compiler would take about 78 minutes, the "cleaned up" one not compiled would take an hour and 56 minutes, and the first program ... over six hours!

Guide to the Sega Genesis Development Kit (SGDK) - part two

by Francesco Donatini

In the previous article, we prepared the setup and obtained the first ROM with the classic Hello World screen print. Now we continue our exploration of the framework dedicated to game development for the Sega Megadrive with another classic of the genre, the display of an animated sprite.



Fig. 1 - So much effort to stun a scorpion... You weren't expecting Sonic, were you?

Before we begin, let's review some hardware features related to sprites:

-The minimum size of a sprite corresponds to the smallest unit that the character-based video display processor (VDP) can handle (8x8px), while the maximum size corresponds to 4x4 tiles (32x32px).

-for standard resolutions with 320px horizontally (320x224px in NTSC mode, 320x240px in PAL mode), up to a maximum of 80 sprites can be displayed (no more than 20 per scanline);

-Sprites are displayed on a dedicated plane/playfield (i.e., independent of the background, for example) measuring 512x512px, so that sprites that are not to be displayed can be hidden from the game display, as the game screen starts at coordinates 128,128 and, being smaller, occupies only a portion of the sprite plane.

-within the RAM dedicated to graphics (VRAM), sprite data is stored in a table (sprite table), which we will look at in more detail shortly;

-a sprite can be associated with a palette consisting of sixteen colors (the first of which is transparent) out of a possible 512 (we will discuss palettes in more detail in a future article).

LET'S PREPARE THE SCORPION

As an example sprite, we will use a cute scorpion taken from a game I am developing, for which animations have been created for movement and for when it is stunned. When preparing the PNG file, it is essential to configure the colors using the indexed color mode and to use a palette consisting of a maximum of 16 colors. In this case, purple has been used, which is the color generally adopted as transparent by convention (RGB values 255,0,255).

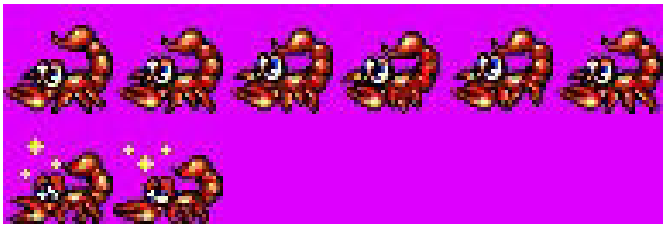


Fig. 2 - The spritesheet with animations

Once the image has been prepared, in the "res" folder, we create the "resources.res" file, which is identified by the SGDK as the configuration file for the resources used in the game (images, audio, etc.), and we insert the reference to our scorpion image, defining it with these attributes:

1. SPRITE is the type of resource that will be a sprite (if it were a static image, we would have used IMAGE);
2. spr_scorpion is the name of the "SpriteDefinition" type constant that we will have available in the code to manage the sprite;
3. "sprites/scorpion.png" is the path of the PNG file

relative to the location of the "res" folder;

4. the two numbers "4" indicate the number of horizontal and vertical tiles that make up a single frame of the sprite, in this case 4 and 4, the dimensions of the single frame being 32x32px.

The Sprite resource configuration line:

```
SPRITE spr_scorpion "sprites/scorpion.png" 4 4
```

There are additional parameters that can be specified that are not of interest to us at this time. However, it is important to know that during compilation, the configuration file is processed by a Java program that will perform a series of optimizations on the multimedia resources provided, transforming them into an assembly format that can be read directly by the framework. You can find all the details in the "rescomp.txt" text file included in the framework, which I highly recommend reading.

LET'S DISPLAY THE SCORPION

To display the scorpion on the screen, we can reuse the code from the previous article, delete the command to print the sentence, and (see the code in Fig. 4):

-First, we compile. If everything has been done correctly, a "resources.h" file will be generated in the "res" folder, in which we find our "SpriteDefinition" constant for the scorpion defined as extern.

-(A) we add the include of the header of the generated resource file so that it is available to the program;

-(B) we initialize the VDP and set the screen, for example, to standard NTSC resolution 320x224px;

-(C) we initialize the framework's sprite engine;

-(D) We assign a palette (in this case, the first one, which is PAL0) with the palette taken directly from the scorpion image; if you try to load the program before and after inserting this instruction, you will see that the background color will change from black to purple. This is because, by default, the screen background color is the first one in the first palette;

```
#include <genesis.h>
// (A)
#include <resources.h>

int main()
{
    // (B)
    VDP_init();
    VDP_setScreenWidth320();
    VDP_setScreenHeight224();

    // (C)
    SPR_init();

    // (D)
    PAL_setPalette(PAL0, spr_scorpion.palette->data, DMA);

    // (E)
    Sprite* scorpionSprite = SPR_addSpriteSafe(&spr_scorpion, 100, 100, TILE_ATTR(PAL0, FALSE, FALSE, FALSE));

    while (TRUE)
    {
        // (F)
        SPR_setAnimAndFrame(scorpionSprite, 0, 0);

        // (G)
        SPR_update();

        SYS_doVBlankProcess();
    }

}
```

Fig. 4 - The code to display the scorpion on screen

-(E) We initialize our sprite using the definition from the resources, placing it at position 100,100 (the position refers to the top left corner of the frame) and assigning it PAL0 (the rest of the parameters concern priority and flip, which we will leave out for now);

-(F) We set the scorpion's animation, for example the first one, i.e. the one with index zero, and the starting frame, for example the first one, i.e. the one with index zero;

-(G) We call the sprite engine update so that the sprite table data can be transferred to the VDP when the vertical interrupt occurs.

As a result, we have our scorpion, which is beautifully still but still present and correctly displayed on the game screen.

A LOOK AT THE SPRITE TABLE

If we run the program using the Gens_KMod emulator (which I mentioned in the previous article as a debugging tool), we can view the sprite table by accessing CPU â†' Debug â†' Genesis â†' VDP - Sprites. Each entry consists of 8 bytes representing:

-"Num" is the sprite number; ours is the first, so it has the number 0, and as already mentioned, with a horizontal resolution of 320px, we have a maximum of 80 sprites available;

-"YPos" and "Xpos" are the vertical and horizontal positions of the sprite. The value is 228 because the plane dedicated to sprites starts at position 128,128 relative to the screen, so the position on the screen 100,100 corresponds to 228,228 on the sprite plane;

-"Size" is the size of the sprite, i.e., 32x32px;

-"Link" is an internal value that indicates a link to the next sprite in the table's linked list;

-"Pal" is the sprite's palette, so the value is 0;

-"Tile" is the initial number of the tile used by the sprite;

-"Flags" are values 0/1 relating to high/low priority and possible horizontal/vertical flip;

You will see that reading data from the sprite table will become more and more common as your games become

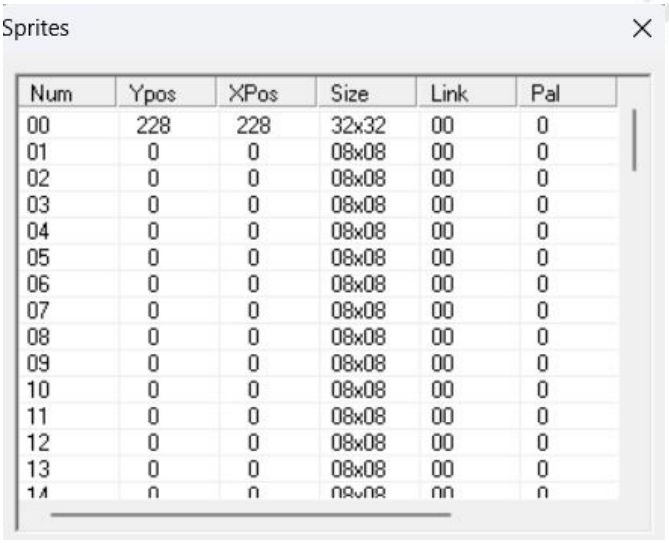


Fig. 5 - The sprite table of the Gens_KMod emulator

richer in graphics and interactions. We will return to this topic with some examples taken from real situations.

LET'S ANIMATE THE SCORPION

It's time to pick up the scorpion we left behind and animate it, stunning it, poor thing, with the stun animation (see the code in Fig. 6):

-(A) we introduce two variables, "frame" to store the current frame to be displayed and "nextFrameTimer" to adjust the frame change speed (otherwise the animation would be too fast);

-(B) in the game loop, we arbitrarily set the value that the timer must reach to trigger the next animation frame to 10, and we adjust the frame value so that it is never different from 0 and 1, as the stun animation consists of only two animation frames;

-(C) we set the animation with the stun and the current frame and advance the frame change timer;

-(D) the sprite engine update is moved to the game loop because the sprite table must reflect the changes in the animated sprite.

This is a simple example of how you can create a constant speed animation. Then, by examining how the Sprite and SpriteDefinition structures of the framework are made, you can build your own engine to, for example, move the sprite, set the speed and direction of the animation sequences, and achieve many other wonders: in general,

```c
int main()
{
    // (A)
    s16 frame = 0;
    u8 nextFrameTimer = 0;

    VDP_init();
    VDP_setScreenWidth320();
    VDP_setScreenHeight224();

    SPR_init();

    PAL_setPalette(PAL0, spr_scorpion.palette->data, DMA);

    Sprite* scorpionSprite = SPR_addSpriteSafe(&spr_scorpion, 100, 100, TILE_ATTR(PAL0, FALSE, FALSE, FALSE));

    while (TRUE)
    {
        // (B)
        if (nextFrameTimer >= 10) {
            frame++;
            nextFrameTimer = 0;
        }
        if (frame == 2) {
            frame = 0;
        }

        // (C)
        SPR_setAnimAndFrame(scorpionSprite, 1, frame);

        nextFrameTimer++;

        // (D)
        SPR_update();

        SYS_doVBlankProcess();
    }

}
```

Fig. 6 - The code for animating the scorpion

the framework provides the building blocks without offering a "ready-made solution," so it is up to us to take on the burden and honor of building the house as we want it.

USEFUL RESOURCES

https://github.com/Stephane-D/SGDK

https://plutiedev.com/

https://danibus.wordpress.com/

https://www.copetti.org/writings/consoles/mega-drive-genesis/

https://segaretro.org/Sega_Mega_Drive/Technical_specifications

PunyInform: installing tools and compiling a game

by Fredrik Ramsberg - adapted for RetroMagazine World by Gianluca Girelli

In the previous tutorial, I gave a brief introduction to PunyInform, a new library for writing text adventures for 8-bit computers (and others). Now let's take a look at what you need to download and install to start writing a game using PunyInform.

For this article, I will be using Windows, but you can also use PunyInform on Mac or Linux.

Step 1: Install the PunyInform library

Go to https://github.com/johanberntsson/PunyInform/releases and download the latest version of PunyInform (the version that appears first on the page) by clicking on "Source Code (zip)" under Resources. Once the download is complete, click on the Zip archive in your browser window to open the archive in Windows Explorer, click on "Extract All" at the top of the window, just below the menu bar. Choose a folder where you want to place the PunyInform files. Windows Explorer will probably suggest something like "C:\Users\MyUsername\Downloads\PunyInform-1_2\", which should work fine. You can always rename and/or move the folder later.

Step 2: Get the compiler

PunyInform is a library for the Inform 6 programming language. To use it, you will need to download the compiler, called "inform" or "inform6". PunyInform requires at least version 6.34 of the compiler. If you are using Windows, it is available as a binary executable, so you can simply download it and start using it. Go to http://www.ifarchive.org/indexes/ifarchiveXinfocomXcompilersXinform6Xexecutables.html and download inform634_win32.zip. Open the Zip archive and extract the file (there is only one) to the PunyInform base folder (the folder containing the "documentation" and "lib" subfolders).

This is how your PunyInform folder should look now [Fig. 1]:

Step 3: Install a Z-code interpreter

Although the real strength of PunyInform is that it allows you to write games fast enough and small enough to be played on 8-bit machines, the games produced will still be playable on virtually any computer, and you will soon discover that you can run most of your tests on your

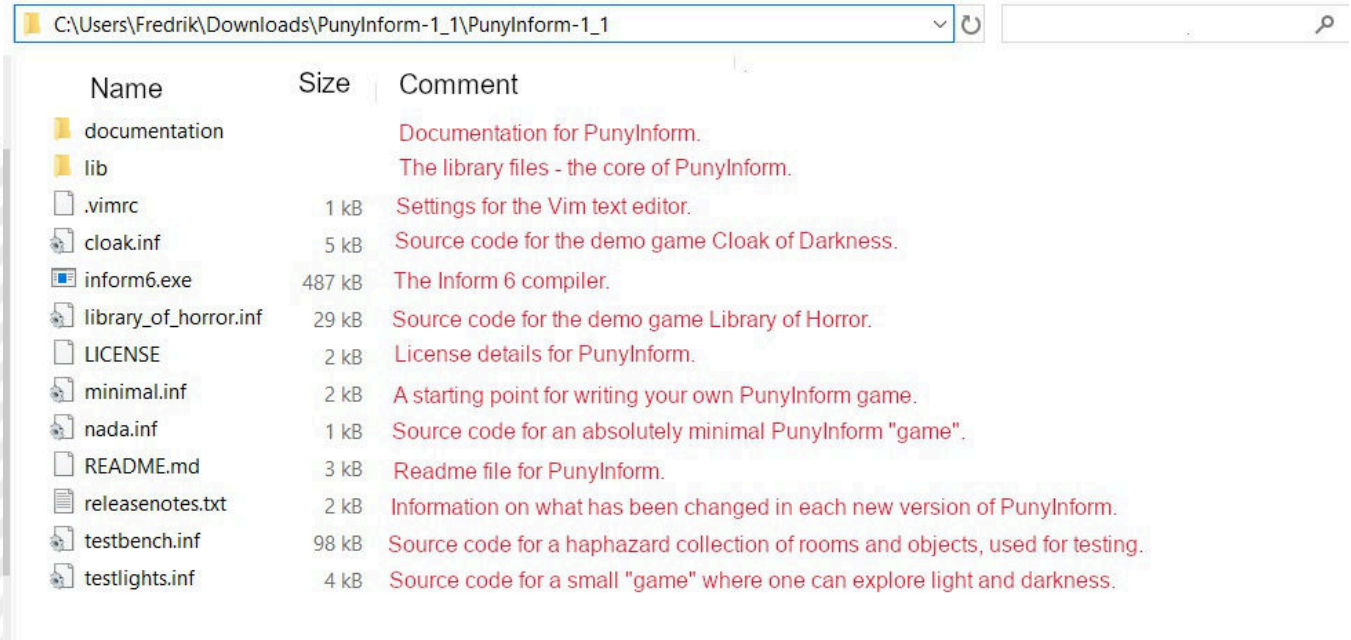
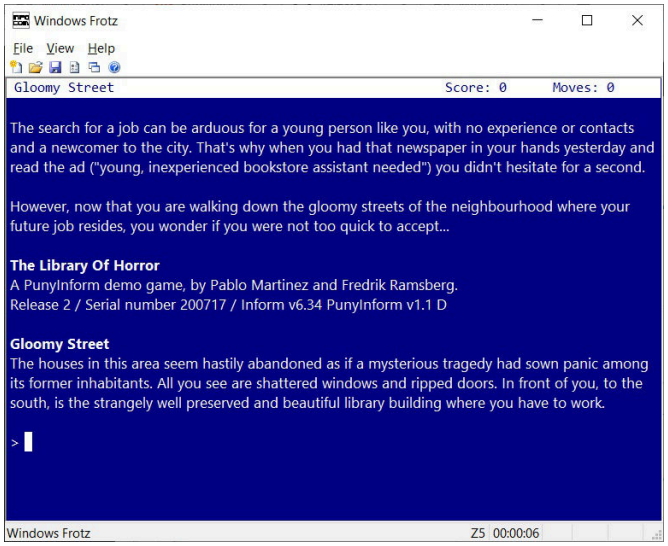


Fig. 1 - The PunyInform folder

modern PC. To do this, you will need a Z-code interpreter. PunyInform compiles Inform 6 source code into Z-code, a game format invented by the wizards of Infocom text adventures. A Z-code file is in fact a memory image for a virtual/imaginary computer called a Z-machine. Although the Z-machine does not exist, it can be emulated on a wide range of platforms, including 8-bit computers from the 1980s, but also 16-bit computers such as the Commodore Amiga and Atari ST, as well as computers running MS-DOS, any version of Windows, Mac OS, Unix, Linux, Android: whatever you choose. All you need is a Z-code interpreter (an emulator for the Z-machine) for that platform. There is a very popular interpreter called Frotz, which has been ported to many different platforms. For Windows, I recommend getting Windows Frotz.



Go to http://www.davidkinder.co.uk/frotz.html, download Windows Frotz, and run the installer. Then launch Windows Frotz once to make sure it has set up the file type associations correctly.

Step 4: Start writing your first game

In Windows Explorer, create a copy of the minimal.inf file and name it mygame.inf. Then open mygame.inf in Notepad (or your preferred text editorâ€"if you want something really nice, you can get the Atom text editor and install the "language-inform6" package in File -> Settings). The mygame.inf file should now look like this [Fig. 3].

Now change the text in the line that starts with Title constant (line 9 from the top). Change A sample game to This is my first game or something else you find amusing. Save the file.

Step 5: Compile the game

Start a command prompt (in Windows 10, click the Windows Start icon on the Windows menu bar and type cmd, then click Command Prompt). Now use the CD command to go to the PunyInform base folder. If it is on a different device, you can type the device letter and ":" to go to that device first, such as D:, and then CD to go to the right folder. CD .. goes up one level, and CD \ (which is a backslash character) goes to the root folder of the current device. If you start typing a folder name, you can press the TAB key several times to display all files and folders with a matching name. Use DIR to list the files in the current folder [Fig. 4].

Now, to compile the source code, type:

inform6 -v3 +lib mygame

This tells the Inform 6 compiler to read the source code in mygame.inf and compile it into a Z-code version 3 file (a so-called "story file") called mygame.z3, and to look for library files in the lib folder. There are many other command line options available for the Inform 6 compiler. To list them all, type inform6 -h2. Most of the time when I use PunyInform, I find myself using the command-line options -e (to use string abbreviations, making the story file smaller) and -s (to print some statistics about the story file produced).

```
12  ! $ZCODE_LESS_DICT_DATA=1 removes an empty data byte for every dictionary word.
13
14  Constant Story      "Minimal";
15  Constant Headline   "^A sample game which uses PunyInform.^";
16
17  ! Uncomment ONE of the two following lines, to show either time or score/turns
```

Fig. 3 - The mygame.inf file

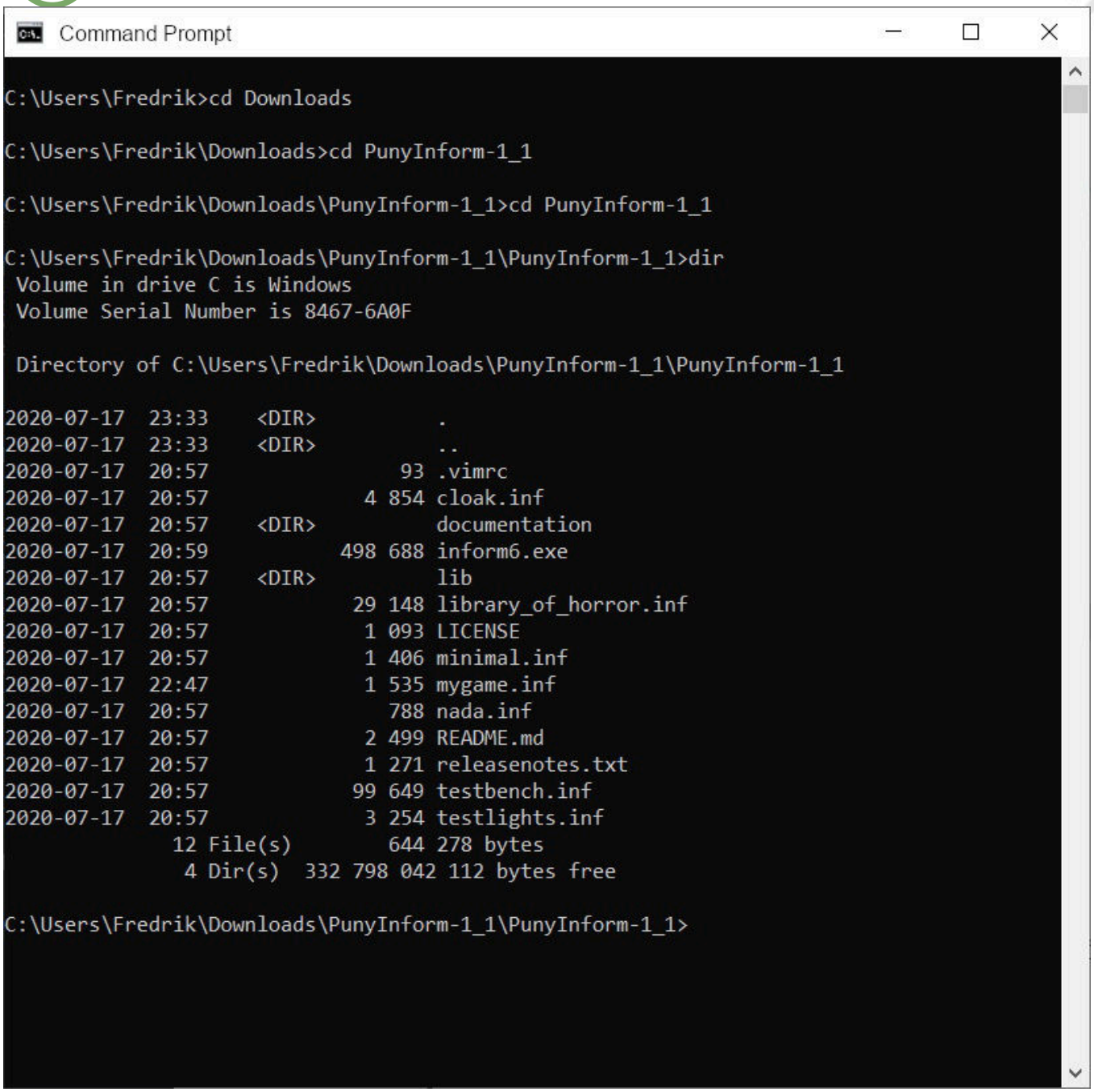


Fig. 4 - The files in the PunyInform folder

Step 6: Launch the game in Windows Frotz

Now use Windows Explorer to go to the PunyInform base folder. You should see mygame.z3 there. Double-click on it. The game should now start in Windows Frotz [Fig. 5].

Step 7: Turn your game into a C64 disk image

Now let's create a Commodore 64 version of your game. To do this, we will bundle the story file with a Z-code interpreter called Ozmoo. Go to

https://ozmoo.online/

In the "Upload Z-code file" section, select the mygame.z3 file and click "Upload." Then, in Build Game, make sure the mygame.z3 story file is selected, then scroll to the bottom of the page and click "Build." If all goes well, you will receive a message that the game has been successfully created, and under Files you can see mygame.d64. Click on the file name to download the file.

Step 8: Test your game on a C64 (emulated)

If you don't already have a Commodore 64 emulator installed, this should help. Download WinVICE from:
https://viceemu.sourceforge.io/index.html#download

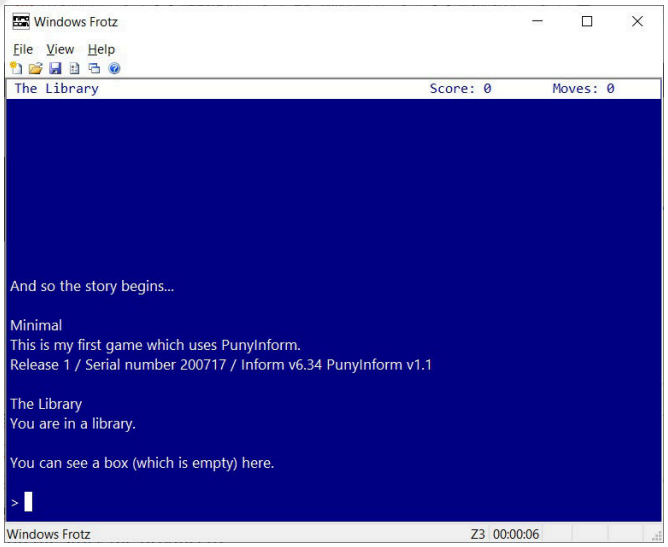


Fig. 5 - The game running on Windows Frotz

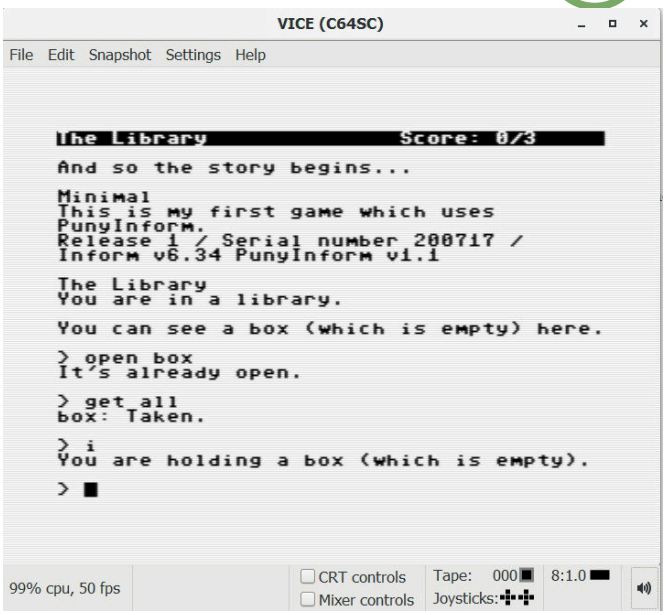


Fig. 6 - The same game running on the Commodore 64

and unzip it into a folder of your choice. You may need to install 7-Zip to unzip WinVICE.

Then go to this folder in Windows Explorer and double-click x64sc.exe. Click File -> Attach Disk Image -> Drive #8 and point to mygame.d64. Then type loadâ€ *â€ ,8 and press Enter. If you get tired of the (realistic) loading time, you can press Alt-W to activate warp mode (maximum speed).

Once the game has finished loading, make sure Warp mode is disabled, type run, and press Enter.
The game will now decompress and then start [Fig. 6].

Useful links

Main project page per PunyInform:
https://github.com/johanberntsson/PunyInformPunyInform

Releases:
https://github.com/johanberntsson/PunyInform/releasesPunyInform

Manual:
https://github.com/johanberntsson/PunyInform/wiki/ManualInform

Designer's Manual, 4th edition:
https://www.inform-fiction.org/manual/html/

A friendly forum where you can ask for help:
https://intfiction.org/

A Discord server for support and discussions about PunyInform:
https://discord.gg/zyYvb8JjE6Ozmoo

A Z-code interpreter that can be used to make PunyInform games playable on Commodore 64/128/Plus 4 and MEGA 65:
https://ozmoo.online/

PunyJam#4 Official Page:
https://itch.io/jam/punyjam-4

Fortran on the Commodore 64 - the MEM command

by Francesco Fiorentini

We continue our experiments with Fortran using the Abacus Fortran-64 compiler on the Commodore 64.
In previous issues, we saw how to write, compile, and run increasingly complex programs with this tool; this time, however, we will try to go a little further.
Those who program in Basic on the Commodore 64 know how extremely limited this language is on this machine and therefore need to use two special commands to read and write directly to memory, unlocking some functions that would otherwise be unusable.
I am referring, of course, to the PEEK (to read) and POKE (to write) commands.
Abacus Fortran-64 behaves in exactly the same way: the basic command set is limited, but fortunately we have been given the ability to access memory directly via the MEM command, which emulates exactly what PEEK and POKE do in Basic.

The MEM command

The Abacus Fortran-64 manual describes the MEM command as follows:

PURPOSE: to examine or modify individual memory locations

MEM (varnam|constant) = integer expression
varnam = MEM (varnam |constant)

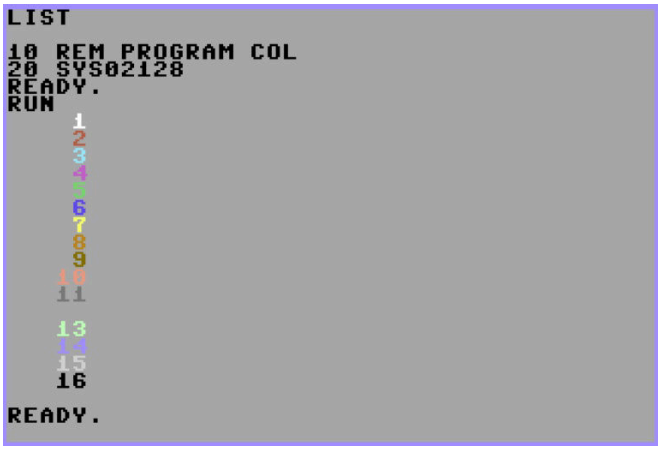
where: the argument between () is an address between 1 and 65535 and varnam is an integer variable

NOTES:
1) Using MEM on the left side of an equation is equivalent to POKE in BASIC. Using MEM on the right side of an equation is equivalent to PEEK in BASIC.

2) Memory locations can only contain values between 0 and 255. Only the lower byte of the integer is transferred to memory. When a value is obtained from MEM, a 0 is stored in the upper byte of the integer variable.
Let's look at a practical example of how to use the MEM command in our Fortran programs:

```
10 : program col
15 :* this program writes
16 :* a list of numbers
17 :* from 1 to 16
18 :* with different colors
20 : integer a,b
30 : open 3,3
40 : do 10 i = 1, 16, 1
45 : mem (646) = i
50 : write (3,100) i
60 : 10 continue
65 : close 3
70 : stop
80 : 100 format(i5)
90 : end
```

This simple program uses the MEM command to alter the value of memory address 646.
Address 646 (hexadecimal: $0286) contains the color code for the current cursor color: the Kernal applies the color code stored here to the text color RAM for each visible character printed on the screen.



Location 53270

Taken from the book "Peek Poke Commodore 64 Manual" by Hannes RÃ¼gheimer and Christian Spanik - CHIP Special No. 9, October 1985.

This register contains many important functions: horizontal fine scrolling, which basically works like vertical scrolling, and MULTICOLOR selection, in which up to 4 colors can be used for each point on the screen.
Bits 0 to 2 define the horizontal movement of the video. The value that these 3 bits can encode ranges from a minimum of 0 to a maximum of 7.
The value normally read is 0.

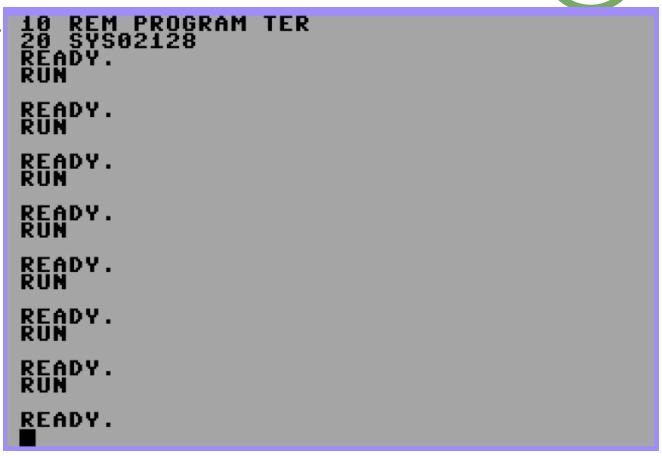
To modify these bits, you can write:
POKE 53270,(PEEK<53270)AND243)OR(num.)
The example below, in Basic V2, is based on a small program published in the book. Using the horizontal fine scroll, we can simulate an earthquake on the screen.

```
5 for i =1 to 50
10 x=x+1: if x=7 then x=-x
20 poke53270,(peek(53270) and 240) or abs(x)
30 next i
```

This is an excellent example of how to use the PEEK and POKE commands. If we wanted to replicate the program in Abacus Fortran-64, we would have to write:

```
10 : program ter
15 :* this program simulates
17 :* an earthquake on the screen
20 : integer x,y
40 : do 10 i = 1, 50, 1
45 : x=x+1
46 : if (x .eq. 7) then
47 : x=-x
48 : endif
50 : y=(mem (53270) .and. 240) .or. abs(x)
55 : mem (53270)=y
60 : 10 continue
70 : stop
90 : end
```

In this program, we have introduced some new Fortran commands.
The IF (logical expression) THEN ... ENDIF construct, which is more effective than its Basic counterpart, as it allows multiple lines to be used within it.
Logical expressions can use the following operators:



Output of the TER program in Abacus Fortran-64. Note the distance of the text from the left side of the screen..

EQ. - equal
.NE. - not equal
.LT. - less than
.GT. - greater than
.GE. - greater than or equal
The ABS() function, which corresponds to that of Basic. And finally, the AND and OR operands, which in Abacus Fortran-64 are written as .AND. and .OR. respectively.

The Helic program

At this point, the question arises... Can we use the MEM command to draw and animate sprites in Fortran? The answer is obviously yes.
The program below shows how it is possible to draw a sprite, in this specific case a helicopter, and move it around the screen using the joystick in port 2. The example is taken from a program attached to the Abacus Fortran-64 disk.

```
1000 : *program sprite display
1010 : *use joystick in port 2
1020 : *to move sprite
1030 :     program helic
1040 :     character a
1050 :     dimension ia(189)
1060 : *sprite data facing left
1070 :     data ia/189*0/
1080 :     data ia/0,255,255,0,0,128,96/
1090 :     data ia(8)/0,128,144,1,240/
1100 :     data ia(13)/159,255,200,103,255/
1110 :     data ia(18)/254,0,14,127,0/
1120 :     data ia(23)/6,127,0,3,254,0/
```

```
1130 :     data ia(29)/0,32,0,15,255/
1131 : *sprite data facing right
1132 :     data ia(65)/255,255,0,1,0,0,1/
1133 :     data ia(72)/0,6,15,128,9,19/
1134 :     data ia(78)/255,249,127,255/
1135 :     data ia(82)/230,254,112,0,254/
1136 :     data ia(87)/96,0,127,192,0,4/
1137 :     data ia(93)/0,0,255,240,0/
1140 : *clear the screen
1150 :     do 15 j=1,1000
1160 :       i=1023+j
1170 :       mem(i)=32
1180 : 15  continue
1190 : *enter sprite data into memory
1200 :     do 10 j=1,189
1210 :       i=12287+j
1220 :       mem(i)=ia(j)
1230 : 10  continue
1240 : *define sprite location
1250 :     mem(2040)=192
1260 : *define the color
1270 :     mem(53287)=3
1280 : *define x-position
1290 :     mem(53248)=100
1300 : *define y-position
1310 :     mem(53249)=100
1320 : *reset x-position msb
1330 :     mem(53264)=0
1340 : *enable sprite
1350 :     mem(53269)=1
1360 : *read joystick
1370 : 20  ib=mem(56320)
1380 :     if (ib-127) 30,20
1390 : 30  if (ib-125) 40,50,60
1400 : 40  if (ib-119) 70,80,100
1410 :     goto 20
1420 : *down (y-pos)
1430 : 50  id=mem(53249)
1440 :     if (id-250) 55,20
1450 : 55  mem(53249)=id+1
1460 :     goto20
1470 : *up (y-pos)
1480 : 60  id=mem(53249)
1490 :     if (id-0) 20,20,65
1500 : 65  mem(53249)=id-1
1510 :     goto20
1520 : *button=quit
1530 : 70  mem(53269)=0
```



Output of the HELIC program

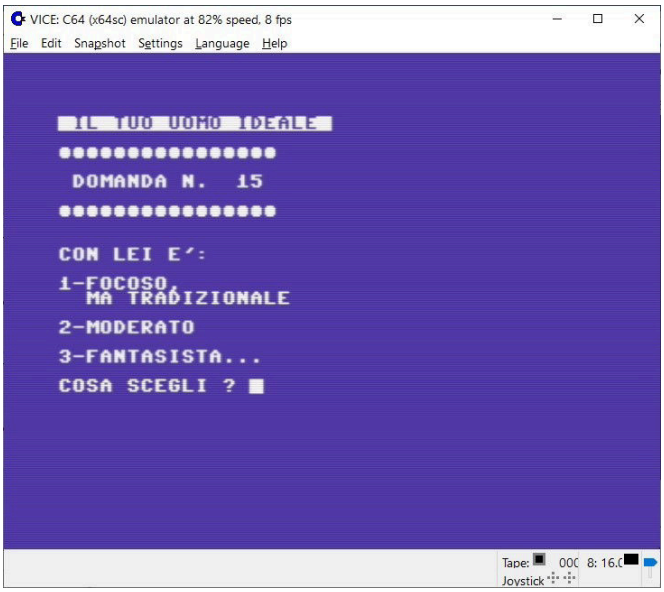
```
1540 :     stop
1550 : *right (x-pos)
1555 : 80  mem(2040)=194
1557 :     mem(2040)=192
1560 :     ic=mem(53248)
1570 :     ie=mem(53264)
1580 :     if (ie-0) 82,82,83
1590 : 82  if (ic-255) 90,85
1600 : 83  if (ic-89) 90,20
1610 : 85  mem(53264)=1
1620 :     mem(53248)=0
1630 :     goto20
1640 : 90  mem(53248)=ic+1
1650 :     goto20
1660 : *left (x-pos)
1665 : 100 mem(2040)=194
1667 :     mem(2040)=193
1670 :     ic=mem(53248)
1680 :     ie=mem(53264)
1690 :     if (ie-0) 101,101,102
1700 : 101 if (ic-0) 20,20,105
1710 : 102 if (ic-0) 103,103,105
1720 : 103 mem(53264)=0
1730 :     mem(53248)=255
1740 :     goto20
1750 : 105 mem(53248)=ic-1
1760 :     goto20
1770 :     stop
1780 : *end of program
1790 :     end
```

For the next article, I decided to take inspiration from this program to port the C-64 game Road Racer, published on RetroLiPS, to Fortran:
https://www.retromagazine.net/c-64-road-racer-commodore-64-basic-v2/

Invitation to... “Personaggio” for C64

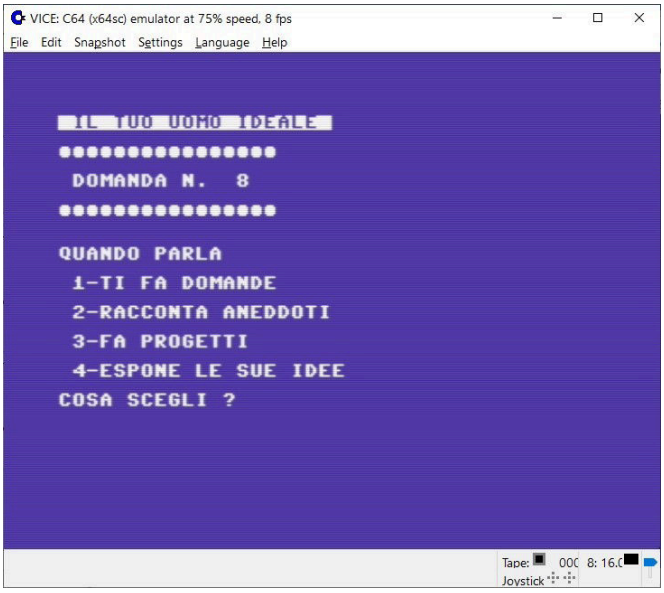
by Eugenio Rapella

"Personaggio" is an old program for Vic-20 that was published in "Super Commodore" back in 1985.
I recently dusted it off and adapted it to the C64; maybe some RMW readers who want to have a laugh (at least that would be the goal) will be interested in it.
The Commodore asks 20 multiple-choice questions (see screenshots); at the end, a woman will have found her ideal man, and a man will find the type of person he would like to identify with (he can choose to answer "as I am" or "as I would like to be"). You can also play "the opposite": by answering with the least desirable alternative, you will get the individual least suited to your personality.



To download the program, go to the website "ready64.org"; in the search box at the top right, type "personaggio" and proceed with the download.
Avoid looking at the list: the result of the test should be a surprise!
The C64 compares your series of answers with those associated with the various characters (in a completely personal way) and suggests the character with the most matches.
Since the program dates back to 1985, some characters are "dated" and, of course, there are no famous characters from the last... forty years!

There is nothing to stop you updating the program:



1. Choose a group of 'characters' (famous people, acquaintances, friends, relatives), at least twenty (the more the better) and list their names on a sheet of paper.
2. Press RUN and, for each person on the list, write the answer you associate with that person next to their name.

Continue like this until question no. 20. At the end, you will have something like this:
GRANDMA OLGA 11121123121111121112
UNCLE MARCO 321...
3. Add (or replace, or vary) the DATES according to the following format:
888DATANONNA OLGA 11121123121111121112
(the order in which these instructions are written is irrelevant).
4. Change the value of P in line 50 to match the total number of characters.
5. Save the modified program and... play.

The characters currently present are only male (at the time, there was no talk of gender quotas); of course, you can proceed differently.

ROTOR RAMPAGE

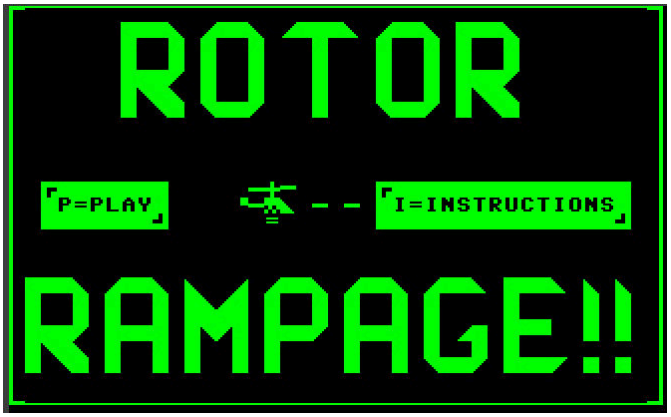
Game review and interview with the developers
by Gianluca Girelli

We have a mission here at Retromagazine World: to preserve the past and bring it into the future. The past we are interested in is, of course, that which concerns retro systems in every possible form, from the study, repair, and/or improvement of hardware to the recovery, preservation, and/or development of new software.

Usually, my fellow editors and I cover (and write about) a handful of systems, but this gap is only due to the fact that we sometimes lack in-depth knowledge of a specific machine. For example, my "weapon of choice" is the Commodore C128, my first computer and faithful companion throughout high school. Since then, I have gotten my hands on many retro systems (Sinclair, Acorn, MSX, and so on), but obviously, it is the C128 that I know best and therefore write about. Yet we love them all (even the more "obscure" ones) and, in order to preserve the past, we encourage and support anyone who can help us expand those boundaries, especially when it comes to talking about less used or lesser known machines.



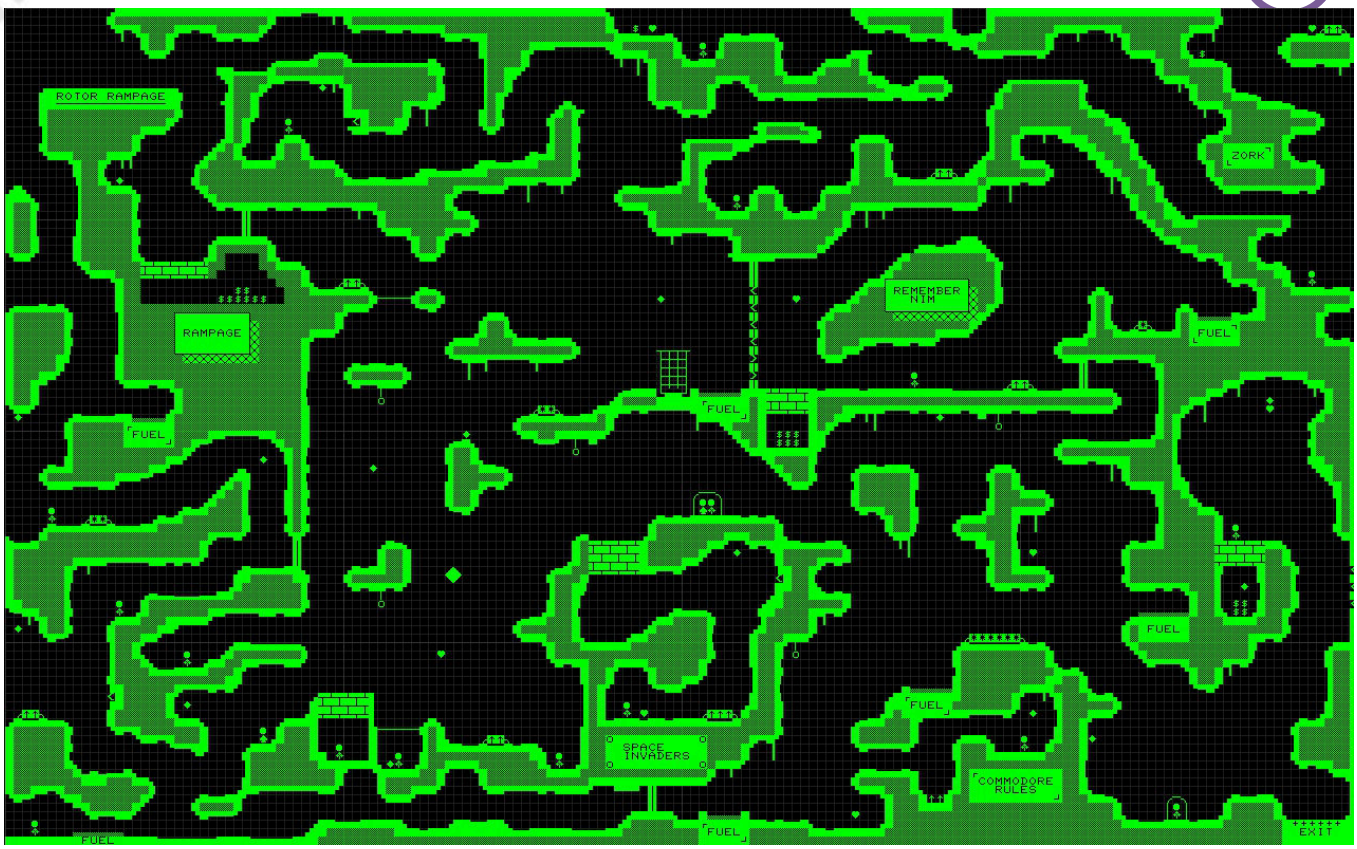
We are therefore delighted to present to our readers a brand new game, Rotor Rampage, developed specifically for the wonderful Commodore PET by Dave Duke and Ken McIlveen.



"Rotor Rampage" is a fresh and exciting video game developed specifically for the Commodore PET, the classic computer that made its debut over 40 years ago. The game is an extraordinary blend of nostalgia and innovation, offering a new gaming experience for vintage computer enthusiasts. Personally, I think this is a great achievement, even more so when you consider that the PET was never designed to be a gaming machine but rather a machine for office productivity.



I had a nice chat with Dave a while back about what these guys do and how the game was conceived and developed. Ken also sent me some more details later. Here's what



they told me.

RM: Hi Dave, can you tell our readers a little bit about yourself and what you do?

Dave: I've always been passionate about computers, especially the PET, a marvel that people started throwing away when new machines came out, but which I continue to collect and restore.

RM: When did this all start?

Dave: I started restoring machines about twenty years ago, and since then, I think I've found and repaired about 50 PETs, which I've ended up shipping all over the world, including Australia and China [Editor's note: Dave lives in the UK]. I also run a website called www.RetroMedics.com, where you can find useful information about PETs, including their schematics.

RM: Tell us how you started developing the game.

Dave: I wanted to create a modern game for the PET, but its limited memory meant that the program would have limited functionality, like most games of the time. I decided that it had to be loaded from disk to be the best it could be. Due to memory and execution speed issues, it was clear that the code had to be in machine language, and this proved to be a significant problem. Disk access routines on a Commodore PET in machine language were impossible to find, so the project remained frozen for several years until I met Ken McIlveen, through a random message on an online forum from another 8-bit enthusiast.

It turned out that Ken had some routines from the early 1980s that could access floppy disk data from machine language on a PET.

Bingo, we had the tools to build what I had been dreaming of for decades.





RM: What language is the game programmed in?

Dave: Initially, I used XCBASIC3 as a rapid prototyping tool, adding Ken's machine language routines. It quickly became clear that to make a game with all the features I envisioned, it would need to be written entirely in assembler. Ken is a wizard with 6502 assembler, and it was clear that everything had to be ported to ML to get the speed and functionality we needed.

For a period of five months, Ken and I worked on the features and gameplay with the goal of fitting as many features as possible into 32k of RAM. I think that when Ken finished writing code in assembler and adding the many, many extra features that I kept adding to the project, he only had a handful of bytes of memory left. There is even a hidden message in one of the maps that refers to what is known in jargon as "feature creep"!

Ken: The machine language parts (and ultimately the rest of the game) were coded using the CA65 assembler package in unison with VICE. These two pieces of software probably cut the development time in half. The hundreds of hours of coding and testing that we would have spent developing on a real C64 were greatly reduced, but keep in mind that it still took several months to complete. During development, we also kept in mind that much of the code could be reused for future games, so we kept things somewhat generic (internally).



RM: That's really impressive, and as a developer, I can see how the game and the whole process behind it evolved over time. Can you tell us more about that evolution?

Dave: "Rotor Rampage" was made by two people with a passion for keeping retro games alive [Editor's note: I love it!], and in the game, we're dealing with a helicopter that goes through different levels, shooting its way through. A rudimentary physics engine renders the falling debris, adding an extra layer of fun. Each level consists of over 16,000 blocks that can be loaded from disk or SD card. The code preloads 16KB banks, then loads another when needed. The game features huge levels to traverse, each containing many references to the past. They also all have different themes, such as mines, Egyptian tombs, internal PET hardware maps, and so on. Finally, you can create and share your own maps using the built-in map editor (coming soon).



Ken: The driving factor for me, as Dave said, was a passion for retro computing. Working with these systems for many years instilled in me an appreciation for the technology of the time. Now, many years later, it's about getting back

to the things in life that I enjoy doing (now that I'm retired). That's why we distributed the PET version for free. Originally, this was just a project to pass the winter months, and we never intended it to be anything more. We enjoyed the process so much that things evolved, and from there we decided to make a C64 version.

RM: What are the requirements to run the game?

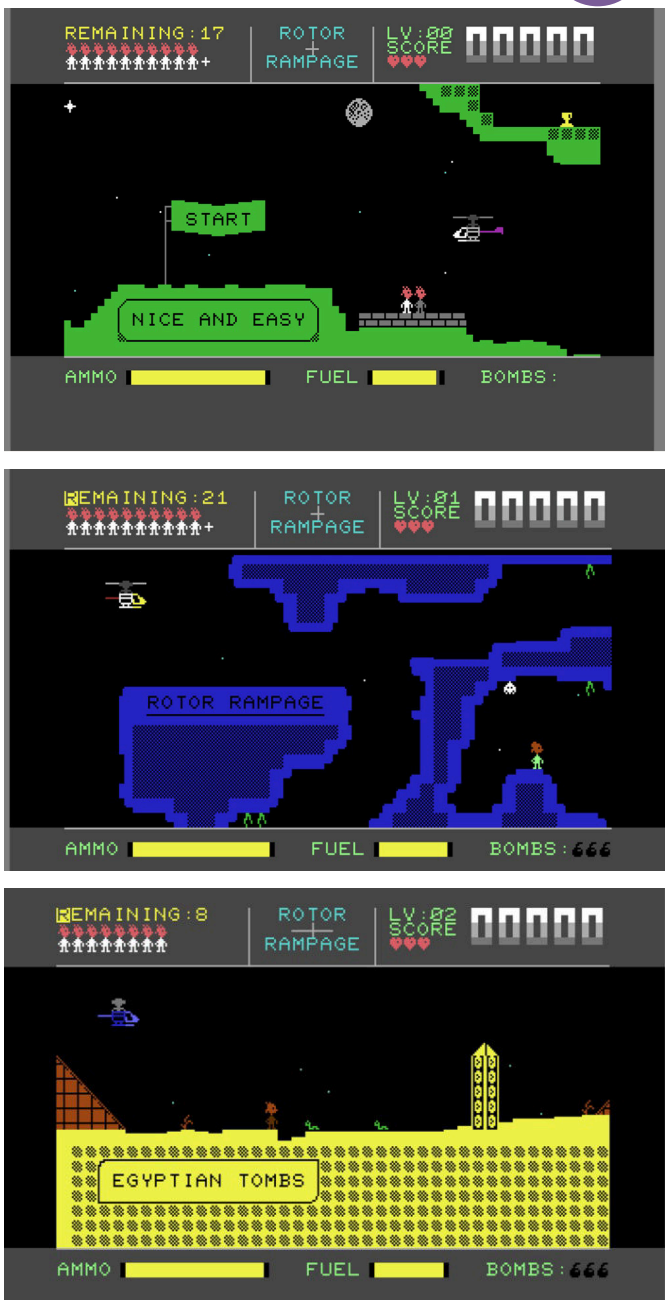
Dave: The main requirement is to have at least 32k of memory. PETs from 2001-n to 4032 are supported, and finally, due to popular demand, there is now a native 80-column version for the PET 8000+. It also now supports joysticks and audio. The initial version requires SD/Floppy disk support, so make sure your PETs have ROM 2 or later.

RM: What about your plans for Rotor Rampage?

Dave: Starting with the original PET version, a completely reworked C64 version is now also available. With the extra memory, we've added better graphics, better sound, and level compression (which means we can fit more levels on a single floppy disk).

RM: Now that you've completed the PET and C64 versions of Rotor Rampage, what will you be working on next?

Dave: We are now working on a multi-user dungeon game that supports 3 different clients. You can play it using TELNET over the Internet (this also means any PET with a terminal client), a web browser, but most importantly, a native C64 app with full-screen AI-generated images. Each dungeon can have 1,000 simultaneous users playing. There may also be interactive AI characters in the game.



Images from the Commodore 64 version

RM: Thank you very much for your time and for this game. Retro Magazine World will continue to follow your projects and help spread the word. Keep the passion alive!

Useful Links

https://www.retromedics.com/

https://xc-basic.net/doku.php?id=v3:start

https://www.rotorrampage.com/

Interview with Richard/TND, a code maniac

di David La Monaca

The RMW staff is delighted to have the opportunity to meet Richard from the TND (The New Dimension) group, coder, programmer, developer, musician, graphic designer, diskmag editor, webmaster, blogger, etc. If you can think of a role in the retrocomputing scene, and in particular in relation to the Commodore 64, well, Richard has certainly played it and continues to do so. A true fan of the world's most famous 8-bit computer and a great supporter and promoter of the scene, let's find out more about Richard's activities.

DLM: Hi Richard, and thank you very much for accepting my invitation for an interview. All the readers and editors at RMW are very happy to have the opportunity to ask you some questions about your experience in the underground world of the C64 computer scene.

RTND: Hello. It's a real pleasure to participate in the interview. I'm happy to answer your questions and try to provide as much information as possible about my history with the C64.

DLM: Can you briefly introduce yourself and tell us something about your life (e.g., where you are from, your education, your upbringing and inclinations, your personal interests outside of work, etc.)?

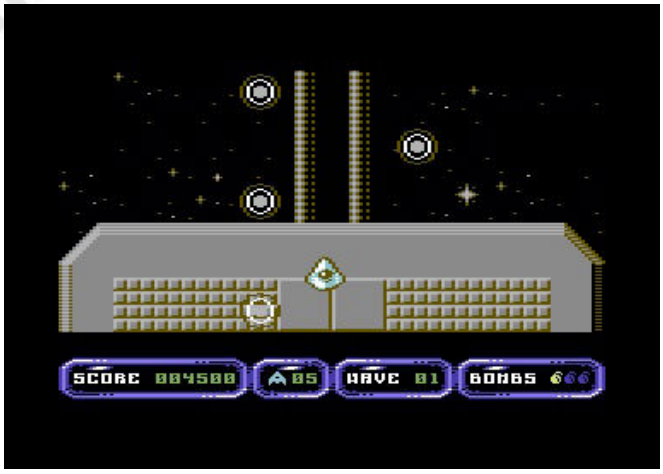


DMC Album - An extensive music selection containing Richard's fantastic tracks

RTND: Hi, my name is Richard and I'm from the UK. I don't really want to give too much information about myself, my education, etc. Like any good Englishman, I'm a very private person by nature! My personal interests outside of work include video games/retrocomputing, playing pool, darts, and going for long walks. I don't have much interest in television today because of the poor quality of programming I endured around the 2000s.

DLM: I imagine you have always been a computer enthusiast and user since you were a child. What got you into computing and what was your first experience with a computer? Was the Commodore 64 your first microcomputer?

RTND: You're absolutely right, and no, the Commodore 64 wasn't my first micro! My first home computer was actually a second-hand Commodore VIC20 equipped with a C2N cassette player and also a 3K RAM expansion cartridge to play some "Bubble Bus" titles. Most of the cassettes I got were for the non-expanded VIC20. I was only 9 or 10 years old when I got this computer. I fondly remember how much fun I had playing different games on the VIC20. I have to say that the games I liked the most were the ones for the VIC20. Titles such as VIC Avenger, a Space Invaders-style game. I also remember Triad, a variant of Galaxian. I also liked Space Rescue, Duck Shoot, and Super Breakout.



CruiserX79 - Fast vertical scrolling shooter not made with SEUCK

In 1990, it was extremely difficult to buy VIC20 games in stores, markets, etc. So I didn't have many, probably 15-20 in total. I went to some electronics stores and really liked the look of the Commodore 64 games on display. So I wanted a Commodore 64 as a birthday present. And in December 1990, I opened my birthday present and it was indeed a Commodore 64 accompanied by a Datassette and some games. Over the next two years, I received some accessories (disk drive, Action Replay MKVI cartridge, Star LC100 color printer, and mouse with various supporting software utilities).

During that time, I also had the opportunity to play with an Amstrad CPC, as I would occasionally visit one of my uncles who had an Amstrad CPC computer. Anyway, going back to the VIC20, I remember really loving that machine at the time, but then, in the late 80s/early 90s, it was really difficult to find and buy new games for the VIC20.

During my childhood, I also remember using the BBC Micro Computer in elementary and middle school, playing educational titles such as POD (not to be confused with the C64 game by Mastertronic) and Symetry. I remember playing Chuckie Egg at school, around the time of the seasonal school holidays. It was great fun to play, so years later, in the 90s, I bought Chuckie Egg for my Commodore 64.

DLM: How did you start exploring your C64, besides playing games? Were you immediately interested in programming and discovering how the machine worked?

RTND: I started working on the C64 in addition to playing it using the Shoot'Em Up Construction Kit (SEUCK), even before programming. SEUCK came with my Commodore 64. I created lots of games using that editor. I really enjoyed developing games with SEUCK. My graphics and design were pretty poor for a teenager. However, it was a lot of fun to create games with this tool.

My introduction to programming began in 1993 when I visited a friend after school. He showed me some programs he had written in BASIC. They were very simple, but in 1994 I managed to program my first game in BASIC, which I called "Cidcom." It was a game where numbers flashed on the screen and you had to press the space bar to prevent the numbers from changing. I didn't publish it because I wasn't aware of the Public Domain software scene until 1995.

1999 was the year I started learning assembly language code through two print fanzines for the C64, Commodore Scene and Commodore Zone. Both magazines were viable alternatives to Commodore Format and also came with software supplements. Some of these included a Turbo Assembler on tape, along with some source code routines with explanations. I also remember learning something from magazines such as Coder's World and perhaps others between the late 1990s and early 2000s. I had to rely on limited resources until I got regular access to the Internet in the early 2000s thanks to broadband services :).

DLM: What was the configuration of your first computer (including peripherals, mass storage, expansions, and accessories)?

RTND: The configuration of my first computer (the VIC20) was very simple. I had a black and white TV (later replaced with a small Saisho CRT color TV), a Datassette, the classic ATARI joystick, and a 3K RAM expansion unit along with a few games. Among these, I remember: Awari, Jupiter Lander, Mars Mission, Alien Plague, Duck Shoot, The Wizard and the Princess, and a couple of VIC Tape Computing cassettes, which contained some games such as "Pennies from Heaven."

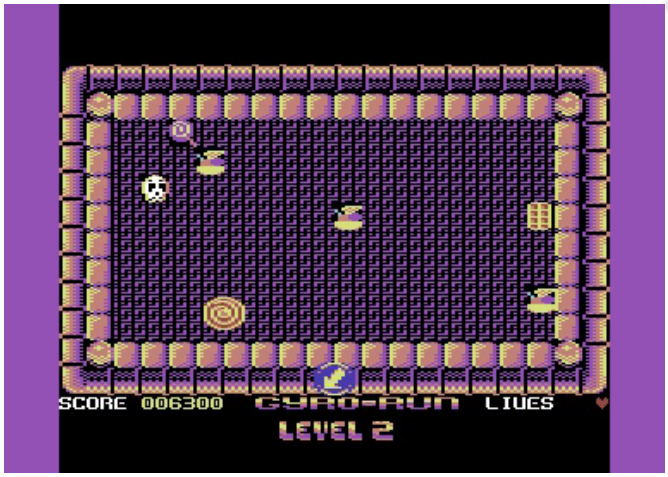
DLM: What prompted you to get involved with computer graphics, music, and sound effects and decide to try your hand at designing your own works?

RTND: I played many impressive productions that I was very satisfied with, especially the games for readers attached to magazines, published on some Commodore Format, Commodore Force, and Zzap!64 cassettes. These games somehow inspired me to try to develop something of my own for the C64.

As for music, between 1994 and 1995 and the early 2000s, I collected many public domain utility disks and tapes. Eventually, I had several music editors made by various members of the Demo Scene groups. So I started experimenting with these editors/trackers before creating my own software tools. The first music editor I remember came with my C64 and was pretty basic. It was Firebird's "The Music System." It was still a fun music editor, though. PD editors/trackers such as DMC, Music Assembler, Music Mixer, etc. had free sounds built into the editor, and with these you could create really impressive soundtracks for the time.

DLM: How did you learn to program, and what was your first program, if you remember?

RTND: My first program for the C64 is hard to remember,



Game Gyro Run - A very strange spinning top game that will make your head spin

and unfortunately, I don't have it anymore. I learned to program in 1998/1999, when in 1995 I was shown a fanzine called "Commodore Zone" (after Commodore Format had closed its pages for good in September 1995). The A5-sized fanzine came with an attachment (optional in disk or cassette format). In subsequent issues, there was an article called "The Hex Files." It started with the basics of displaying objects on the screen, creating scrolling text, making simple image demos, and eventually creating complete games. With the help of the Internet, I also contacted other people in the scene who sometimes helped me when I was trying to learn how to create new things.

DLM: What was the entire system you used to develop and test the first drafts of your productions (games, demos, SID tracks, graphic images) like?

RTND: Before discovering cross-assembler development tools, I used a C64C with a 1541-II disk drive and some public domain disks that I bought in PD libraries in the UK for less than Â£2. Most of them contained programming tools and utilities. I mainly used Turbo Assembler V5 for coding and an Action Replay cartridge to use the machine code monitor. I also used music editors such as Music Assembler V1.0 from the Dutch USA Team, DMC from Graffity, and some other software for composing. Sprites were drawn with Sprite Editor V1.3 from Faces, while

characters were drawn with Dunex Char Editor V2.0. When I designed the game screens, I basically loaded the charset and edited the screen manually on the BASIC screen. It was quite a laborious task. After finishing the game, I used a packer and a cruncher (these operations took a long time).

DLM: When did you start getting involved in the C64 scene? How did you come up with the idea? How did you start collaborating with other friends and sceners? How did you communicate with each other at that time?

RTND: I first entered the C64 scene in 1995, when I sent some material to Binary Zone PD (Colored, Nyaaaah!, and Sector 5). I didn't realize I was taking a step towards something new. That is, until the Internet boom arrived. The first group I joined was Civitas, at the specific invitation of Zeitgeist (who unfortunately disappeared mysteriously after our last contact in 2006). I helped with demo code, intros, music creation, and even a few games. When the Internet became even bigger, I was contacted by many people in the scene and collaborated on several projects, which I still collaborate on today.

DLM: Did you take any computer science courses to start your career in electronics and computer science, and if so, what were they and how much time did you invest? Or, like many designers/programmers in the early 1980s, were you self-taught in coding?



Mix Box Music Disk - A good mix of different songs for your refined ears

RTND: Between 1995 and 2000, I took some computer science courses. At first, it was a first national diploma and a higher national diploma in computer science. Some parts of the course included programming. I used GWBasic/Q-Basic, Visual Basic, and C++. Although C++ was the most "unpleasant" programming language, I still completed the courses and passed them with flying colors. So in real life, I invested five years at university in computer science courses. My favorite subject was programming. I remember that once, before Christmas 1995, we were given a kind of gift, namely a couple of complete GWBASIC listings consisting of games to type in. The games were Concentration and Artillery Duel.

As for programming on the C64, I was kind of self-taught. However, sometimes I had to rely on the help of members of the Scene for some specific problems I encounteredâ€"or that I had never seen before. Fortunately, there were always some very helpful people who helped me with the pitfalls I encountered and that I tried to overcome while learning a lot at the same time.

DLM: How did your process of developing a new project or production unfold at the time? For example, did you sketch drawings and diagrams to follow an initial concept, talk to a team about the plot, draw inspiration for the soundtrack, images, etc. from other titles?

RTND: None of the above. I didn't want to waste paper and ink designing and developing a new game. Instead, I would create a structure using raw graphics directly on the computer. Then, at a later stage, I would improve things a little.

When it comes to making music, I experiment with music editors and trackers until I get the right tracks that I think are suitable for my productions.

DLM: Technology and many development tools have certainly changed over the years, but when it comes to the creative side of your work, do you still approach the process in the same way you did for your most recent productions?

RTND: I agree with you that development tools have changed over the years. Although most of the time, I use cross-development tools, such as C64Studio, CBMPRGStudio for programming, Cuneiform, Char Pad, Multipaint, and Sprite Pad for graphics, and Exomizer for compression. I also use the VICE emulator for testing. This makes the process of developing, compressing, and testing the game shorter than it used to be.

I used a freezer cartridge like Action Replay, public domain utility disks to create and develop graphics and music, and Turbo Assembler to program the software. I had a lot of fun with these tools. In fact, I still use them from time to time :-).

DLM: What was the biggest technical hurdle you overcame while designing/programming/testing your productions for 8-bit machines (C64)?

Definitely raster timing and bug fixing are topics that gave me a hard time. I really hated this phase of code development. Sometimes it's something very simple, other times it's a real challenge. For example, when you move around to try to eliminate flickering, you move on to the next raster line. The more code you add, the more the timing changes. I find that the best thing to do is to leave raster timing for a demo/intro or something like that as the last part before the actual release.

A real challenge for me, which I've never really tackled, is double-buffering in games. Mainly because most of the time I don't have enough memory to do it. The RAM is usually taken up by other data. That's why most of my games only use three colors if they use vertical or horizontal scrolling of the game map. Amazon Tales, for example, uses more colors, but doesn't use double buffering.

DLM: What was/is your favorite game for the C64? Do you still find time to play video games?

RTND: As far as favorite games go, my all-time favorite C64 game is Mayhem in Monsterland. I remember trying out the playable demo of the game in 1993, when Commodore Format gave it away on the double cassette Power Pack tape with issue 37. I eventually bought the full version on disk. I was very impressed with the overall presentation, graphics, and concept.

There is also another favorite C64 game of mine that is a real challenge to play. That is, of course, Armalyte. For me, it is perhaps one of the most difficult C64 shoot 'em ups of all time. I can't even get past the first level! Despite this, I have replayed it many, many times to test myself.

I usually manage to find some time to play video games on weekdays, when TV is dominated by depressing news and current affairs programs, soap operas, and dramatic films. Other times, I surf the Internet or do things for the



Rocket & Roll - Experience the most entertaining journey of your life in this arcade adventure created with SEUCK



Starfysh - Since when do taxi lanes cross the galaxy?

C64/Scene such as programming, composing music, working for the disk magazine Scene World, etc.

DLM: What was/is your favorite C64 soundtrack/chiptune? What are your top 3 original SID tracks that have never been released? Do you still find time to follow the latest productions?

RTND: My favorite soundtrack is Cybernoid 2 by Jeroen Tel. I don't really have a top 3 of favorite SID tracks. From time to time, I find time to follow the latest productions thanks to the Internet.

DLM: Speaking of SID tracks, who is your favorite C64 artist/musician/composer of all time?

RTND: Definitely Jeroen Tel because his melodies in C64 games are catchy and memorable.

DLM: Now a question that everyone would like to ask you as an expert C64 coder: what tools do you currently use to create games, demos, applications, etc., and what is your setup (do you use a Mac or a PC)?

RTND: The tools I regularly use to create graphics are SpritePad V2.0, CharPad V2.7.6 (both free editions), GangEd, PixCen, Multipaint, Cuneiform, and Ultrafont PC. For programming/development, I use CBMPRGStudio, C64Studio, sometimes just Notepad++, KickAssembler, ACME cross-assembler, and TMPx. For compression, I regularly use Exomizer V3.1.2 or TSCrunch.

DLM: Which editor and tools do you use when composing songs for old machines such as C64, Amiga, or consoles? Do you still like using old hardware?

RTND: There are a few composers I use to make music on the C64. The most common music editor I use is on the C64, and that is DMC V4.0 or DMC V5.0. Recently, I have been using EMS V7.03. In fact, I'm using it on a C64 game that I developed during the winter and hope to release in April 2024 (if the game is completed). On the PC, I regularly use Goat Tracker V2, but occasionally I might also use CheeseCutter and SidFactory 2.

DLM: I imagine you have a collection of stock C64s (i.e., all versions: C64 "breadbin," C64c, C64g, C128) for testing purposes or just for nostalgia. Am I wrong? Are you also a collector of retrocomputers, not just Commodore brand?

RTND: I'm not really a big collector of retro computers. However, I have collected a fair amount of C64 games on cassette. These are mainly cheap games and tapes that came with magazine covers (Commodore Format Power Pack, Zzap! 64 Mega Tape, Commodore Force Reel Action). I've always been attached to the C64 and have owned five in the past, three of which are now defective and one was destroyed by a child visiting my home. They were all C64Cs.

The motherboard of my longest-lived C64C was replaced by an Ultimate64 in the old C64C case due to power supply issues. I remember that this C64C displayed a faint, large black line running across the screen and the character map was garbled. So, as a last resort, the

Ultimate 64 was a great replacement for me.

Among other retro technology I own are a C64 Mini computer and a C64 Maxi (on which I can also play memorable VIC20 games). I also have a mini NES Classic console, a first edition Evercade VS for retro games not related to the C64. I also have a custom arcade machine that my brother made for my 40th birthday a few years ago.

DLM: Did you initially work alone on the C64, in a team, or did you simply consult with other coders to achieve a particular result or to fix a bug in your code? Do you generally prefer working in a team or alone?

I usually tried to develop my productions on my own, but since the C64 Internet community has grown so large, I think it's better to work and collaborate as a team, because with a good team there can be much more quality in a production. I like to develop for the C64 for fun, not for money. It's always a good idea to collaborate with other people. They tend to test my material and report any bugs that need to be fixed, or they advise me on improvements that can be made to a C64 production based on our shared experience.

DLM: I'm sure you've worked very hard on your projects over the last few years, but I'm also sure you've had a lot of fun doing it. What is the funniest/strangest moment or story you've experienced while working on your productions?

RTND: Developing productions for the C64 has been really fun. It can sometimes be a headache when dealing with last-minute bug fixes, but the overall result is usually pretty good (or bad).

Speaking of the funniest moment in developing a new game for the C64: the funniest (and perhaps most hilarious) moment happened in 2022. I saw a poll on Twitter posted by the Commodore Format Archive. It said:

Vote in this important poll:
- Dick Tracey
- Cisco Heat
- Tasered in the Crotch

After the poll closed, the results were revealed. For a laugh, I decided to create a little game called "Tasered in the Crotch," a sort of shooting gallery. In the game, you control a team of police officers who have to taser robbers (they are dressed in striped sweaters and carry bags with loot on their shoulders) and avoid hitting ordinary citizens. After the game was released, the feedback on the production was amazing. It's funny how something made for laughs turned into an entertaining masterpiece. When I told my family about this game while





Star Hawx - These evil space birds want to conquer the Galaxy. Go feed them a laser pie.



Snake vs Bomb 2 - This snake is hungry for fruit, but could it meet an explosive end?

I was away from home, we all laughed about it.

DLM: Looking back at the starting point, is there anything you regret about a single production (music or program/game/tool/application) or any other detail? Would you do anything differently if you could?

RTND: Sub Hunter was a great game, and it took over 4 years to develop. Unfortunately for some in the C64 cracking scene, it was poorly programmed. The game wasn't that technical, but I remember seeing the title cracked and modified with trainers by a fake cracking group, with a message about the production addressed to me saying to learn how to develop code properly. They classified the code as "spaghetti code" (as stated in the intro they put in front of the game). Clearly, I wasn't very happy about that.

However, if I could go back in time to that game project, I would perhaps have modified the code to use self-modifying data, instead of constantly repeating the same code and interrupts for each level. This was probably one of the biggest game projects I've ever worked on. At one point I got lost, but in the end I managed to complete the project.

DLM: When did the idea for TND come about, both as a production group and as a website that you built to publish all the releases and also some programming and technology tutorials?

RTND: The TND website (if I remember correctly) may have been created in 2000 or 2001. It was initially created to showcase my productions. It later turned into a kind of community that invites people to share their content and contribute to the website. It was really nice to be contacted with C64 games to share on my site. When the SEUCK games arrived, I didn't want people to have to wait long for the disk to load. So I loaded the game, added some music (after notifying the author), and then compressed the games into executables to be loaded in C64 BASIC. The authors were quite happy with this.

Later, I expanded the TND website to make it something slightly different. The programming and technology tutorials were fun to prepare, including the SEUCK School. I don't remember when I started running the programming tutorials. I only know that the "Assemble It" column started as a small development tutorial for creating intros, small demos, and small games while I was still learning to program in assembly on the C64. The examples were originally based on Turbo Assembler.

In 2018, the mini replica of the C64 (the "theC64") was launched. I tried the mini console in April 2018, before pre-ordering an Ultimate 64 in June 2018 (when my Commodore 64 started having problems). With the retro community gearing up to create new games for this console, I decided to expand "Assemble It" and make it even better than before. This time, "Assemble It" was based on modern IDEs rather than Turbo Assembler. Since I was using ACME Assembler a lot at the time, I decided to use C64 Studio as my main tool for writing code, as well as cross-development tools such as Char Pad and Sprite Pad for graphics, Goat Tracker for music, and

Exomizer for compression.

The new "Assemble It" section is not just about creating and developing a new game for the C64. There is some introductory programming, demo creation, burning disks, tapes, and cartridges, and finally a section with my previously released games complete with source code. The SEUCK School section is based on improving games created with Shoot'Em Up Construction Kit. This ranges from adding music to front-ends and gameplay to creating complete SEUCK games, such as creating a new front-end, adding power-ups, and much more. Some SEUCK enthusiasts have even contributed suggestions for improving SEUCK. I have also included a special feature on the SEUCK Redux engine.

DLM: Among your many publications are many SEUCK games. When did you first encounter this development tool, and why did you like it so much that you explored all the possibilities available for creating games?

RTND: I first encountered SEUCK in 1990 because it was included in the "Light Fantastic" package for my Commodore 64C. I really liked it because I didn't have to do any programming at the time. Creating a game using the kit was pretty straightforward. SEUCK was part of my youth. It also helped me become a game designer for the C64, even though I spend most of my time programming.

DLM: You worked on at least two different diskmags, "GameOverview" and "Scene World," the latter probably the most famous C64 diskmag along with Vandalism News. Were you a reader of diskmags before joining the staff of a diskmag and becoming an editor?

RTND: I wasn't actively involved with GameOverview, but I contributed music to the releases. Scene World was the only diskmag for which I mainly wrote material. I was



Scene World 32 DiskMag - A PAL/NTSC disk magazine, spontaneously created by long-time C64 / retro computing enthusiasts



Sub Hunter - A ton of action in this superb shooter littered with enemies under the sea

also a reader of some diskmags on the scene before joining Scene World as an editor. I especially liked diskmags that contained bonus software to load and run or to learn code from. Scene World was a small diskmag before it became something big on the Internet.

This last question concludes our interview with Richard/TND. Once again, we thank him for taking the time to answer our questions. We hope readers enjoyed reading it and can draw inspiration from Richard's many activities and the passion he puts into all his work.

References

TND - website
- https://tnd64.unikat.sk/

SWO - official website
- https://sceneworld.org/

RCB (Richard's CSDB profile page)
- https://csdb.dk/scener/?id=1201

Richard's GitHub page
- https://github.com/RichardTND

Japan Episode 24: Gacha...pon!

by Michele Ugolini

How many times as children did we exasperate our parents during those famous shopping trips dedicated to buying clothes?
How many times as children were we exasperated by our parents during countless, seemingly useless purchases in shopping malls?
How many times as children did we complain about going to barber X, store Y, pharmacy Z, etc.? etc., etc.?

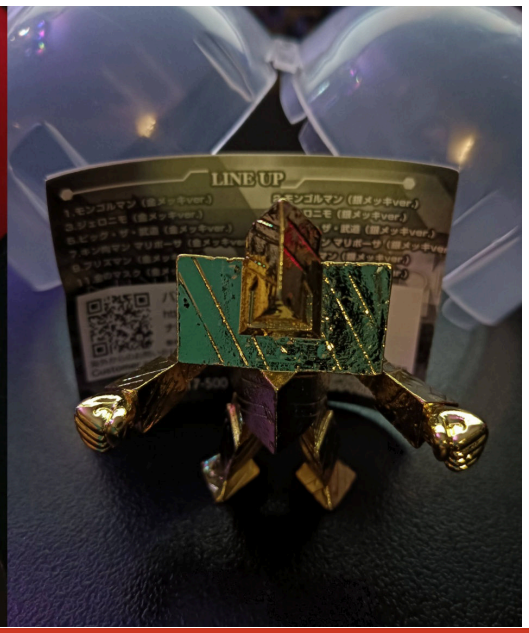
Is it possible that there has never been a solution to this dramatic, global, and ubiquitous problem of children, innocent souls oppressed by weekly family obligations?
Is it possible that in the minds of our most holy and patient parents, the desire for a solution to this global scourge has never been expressed? Why? Why is there no solution to all this? The crying and tantrums of children dragged by force to shopping malls... to the barber... to the dentist...
Are we really sure that there is no solution to this global problem anywhere in the world?
Dear readers, I have just returned from Japan and, as you can imagine, I am about to talk about the solution to this dramatic global scourge: saving children around the world from the hateful obligations of family shopping.

A tutorial to save parents' eardrums from the screams of their beloved children! As well as a suggestion for many shopkeepers!
Of course, I have exaggerated the concept, but if we think about it, temperature is the total amount of heat multiplied by the volume of a mass. So, by analogy, if we multiply lots of little cries (little heat) by a huge number of sad children (mass), we will surely reach a noteworthy global dissatisfaction thermometer: a thermometer that measures high fever!

Let's dig into our memories. When we were children, our parents would occasionally give us a small, inexpensive toy to take us to mass or the dentist or the shops... without too much fuss. The luckiest ones could stop by the newsstand to buy a beloved Commodore 64 cassette. We Italians especially loved these cassettes. Mysterious cartridges with a description of the games worthy of "glorious intentions" often reincarnated in a few sparse pixels!
If we dig even deeper into our memories, we can probably remember that in many cities and towns in the suburbs, outside and inside newsstands, as well as in other more or less unlikely locations, there were machines that dispensed chewing gum and plastic balls containing some mysterious object! They are still there today. Often ignored and yellowed in the corners of newsagents and some tobacconists!
The phenomenon of these balls is not very popular in Italy, while in Japan it is a real fashion, which has come back into vogue in the last ten years, after a brief hiatus around 2000.

In Japan, these capsule vending machines, containing any object our imagination can 'believe it can imagine', are located in every strategic tourist spot, shopping centers, next to barbers, children's clothing stores, along streets with long rows of machines, in airports, outside stores selling special collectibles, etc.
These machines, which are almost never powered by electricity, are based solely on the mechanical





principle of receiving 100-yen coins and then dispensing a ball by manually turning a round crank. The name Gachapon comes from the sound this crank makes when we turn it with our hand after inserting the correct number of coins. A plastic sound similar to "gachagachagacha".
After turning the crank, the reservoir of balls is unlocked and a ball falls out of the machine into the appropriate container with the classic "pon" thud. This is the origin of the term Gachapon. The Japanese often use this onomatopoeic logic to enrich many nouns and verbs. For example, "dokidoki" demonstrates the emotional tension of a heartbeat by repeating the sound of the beat itself.
The main registered brands of these Gachapon are Takara Tomy and, above all, Bandai. Both brands are so famous that any introduction is superfluous. What's more, Takara Tomy recently sent spherical mini rovers (SORAQ) to the moon, which successfully carried out various missions.
It's a pity that the main lander landed at an angle. In any case, the Japanese learn quickly from their mistakes, and the next mission will surely be more successful.

Let's return to our Japanese shopping malls with areas dedicated to vast corridors crammed with these machines. We can imagine these streets crowded with noisy children! No, nothing could be further from the truth! For every quiet child who carefully chooses where to spend their coins, there are at least two mothers chatting as they choose their machine, perhaps commenting on some funny anecdote related to the item that will come out of the desired Gachapon. Obviously, we are talking about a desired and not a predictable ball, in fact, in each machine there is a very specific series of variations of the object that is distributed. A simple example: being a musician, I was initially captivated by a Gachapon that distributed miniature musical instruments. The set of these instruments consisted of a piano, trumpet, clarinet, saxophone, etc.
There is even a series of Kawai pianos to complete the entire series of piano instruments alone. Imagine how many models there could be for each musical instrument inside that machine! This piano is truly faithful to the real thing, even the top cover can be opened and there is a lovely thin rod to hold it open. Amazing.

Other miniatures were dedicated to the various mascots of their various television channels, certain food and drink items, famous people, rare animals (Balaeniceps rex!) and pets, hardware and kitchen utensils, items related to the Tokyo subway and other subways and companies, with the various stops written on mini plastic plates, or miniatures of classic objects found in everyday life, etc... there is no limit to the imagination of such miniature collections! Often even extreme fantasies and "adult fantasies", I will talk about them in future articles! One series that we "boomers" just can't resist when we see Gachapon in Japan is Kinnikuman and various wrestlers, because in the past, plastic "m.u.s.c.l.e." figures had arrived in the West. In Italy, for example, we experienced the saga of the famous "Exogini," a line launched in 1987 by GIG. The original idea for these figurines actually originated in Japan in 1979 with soft plastics, similar to erasers, of the hero Kinnikuman.

These erasers were immediately considered collectibles in Japan. In Italy, the Exogini arrived with a hastily conceived and poorly defined story: alien toys who came to Earth as wrestlers, end of story! They were made of hard plastic and developed in four series, in various colors and transparencies.
The forty characters that made up the first series were called Exogini in Italy, Muscleman in France, and Musculman in Spain. In Italy, they were not available in vending machines but were sold in various formats only in physical stores, such as newsstands. They were distributed individually in bags at a price of 500 Italian lire. Later, they were also sold in jars of ten pieces (4500 lire) and pyramids of twenty or forty pieces (8000 and 12000 lire respectively). They are divided not only by character but also by color: the classic pink, green, orange, transparent, and opaque flesh colors, plus two special transparent colors containing glitter. Later, given their enormous success, they were reprinted in new colors: metallic blue, metallic gray, white, black, purple, and other colors. A resounding success! Today, they are collected and even reproduced in new variations by many people in different European countries. Four long-running series of Exogini! Nothing compared to the longevity of these figurines in Japan. They are still going strong today, in numerous formats, colors, and finishes.
And not just rubber. Since the Kinnikuman vs. Exogini debate is truly titanic, I have attached a photo of the beautiful miniature piano and a beautiful metal piece from "Bandai." This way, I can pique your curiosity about these Gachapon and refer you to the next article, which will be titled "Gachagacha... pon!".

See you next time on this journey into the past, which is now back in the present, with excellent prospects for the future. Basically, a reflection of their philosophy.
See you soon!

NEW GAME

PANG+

Year: 2024
Editore.: Syx, Red Angel
Genre: Action Arcade
Platform: Amstrad GX4000
Website: https://www.mojontwins.com/cepeceros/descargas/pang-plus-correccion-de-errores-y-mejoras-graficas-por-syx-y-redangel/

Back in 1989, Mitchell Corp. released a simple but addictive arcade game. The classic coin-eater attracted huge numbers of players.

This title was Pang, later converted to a multitude of systems by Ocean.

We saw it on the C64, ZX Spectrum, Amstrad CPC, Amiga, MS-DOS, and Atari ST.

The game was quite simple in its mechanics: the protagonist (or protagonists if playing in pairs) had to destroy huge bouncing balls that infested the levels.

Once hit, the spherical objects broke into smaller pieces until they disappeared from the screen. Once all the balloons were eliminated, the player moved on to the next level.

During the game, it was possible to collect power-ups or encounter obstacles.

Among the converted versions, one of the best was the one released for the ill-fated Amstrad console, the GX4000.

It was a very polished version, almost identical to its arcade counterpart.

In 2024, developers Syx and Red Angel gave us this beautiful Pang +. Optimized graphics, better use of colors, and faster action.



All this is complemented by the beautiful gameplay of the original title and a good soundtrack.
A game to replay and appreciate.

This "Plus" version can undoubtedly





be compared to the 16-bit versions and, in terms of optimization, surpass them.

by **Carlo Nithaiah Del Mar Pirazzini**

Hurry up and download it.





OUR FINAL SCORE

» Gameplay 90%

Pang is a classic of its genre, and this version is practically perfect.

» Longevity 90%

The dynamics of arcade games apply. Pang is a game that needs to be perfected game after game.

NEW GAME

SHADOW GANGS

Year: 2022
Editor/Developer: Wave Game Studios, JKM Corp
Genre: Beat em up
Platform: Sega DREAMCAST

Shadow Gangs, developed by JKM Corp based on the Isle of Wight and published by the increasingly prolific Wave Game Studios, has been eagerly awaited by fans.

The Dreamcast port, backed by a hefty £25,000 funding, hit the virtual shelves in March 2022.

The title is a love letter to Shinobi. It is a side-scrolling beat 'em up with platform elements reminiscent of Sega's fabulous title.

The aim of the game is to guide the ninja warrior DAN and maintain order and peace in the world with slaps and shuriken.

There are 10 levels to tackle, which are quite varied and full of enemies to fight, hostages to free, and items to recover.

Our hero's arsenal is enriched with powerful ninja attacks (reminiscent of the special attacks in Streets of Rage) and power-ups that improve the strength of our throwing weapon.

The title runs perfectly on the Dreamcast at a smooth 60 frames per second. The animations and scrolling are smooth, and the sprites and game levels have a nice graphic design.

Shadow Gangs is also well-finished in terms of its audio features. The music is well composed and fits the game situations perfectly, and the sound effects are well done.

This is a direct conversion from the PC version, and I must admit that it carries the same problems, namely extreme difficulty.

It's a difficult game to play, with a difficulty curve that isn't always balanced and veers decidedly upwards. You need to master the controls perfectly, hone your reflexes, and, above all, memorize the flow of enemies level by level.



Those of us whose skills have been weakened by the more forgiving games released in recent years should steer clear of this title.





OUR FINAL SCORE

» Gameplay 75%
Precise controls but monstrous difficulty index right from the first level. Pays homage to Shinobi to perfection.

» Longevity 70%
Suitable for fans of the genre.

All they needed to do was include an 'easy' mode that would allow players to master the game and perhaps make it more accessible to those who don't want to spend an afternoon cursing.
As it stands, the 'Rising Ninja' mode somewhat satisfies my request, but only because it comes with 99 CONTINUES (as opposed to 15 in the classic mode).
However, despite the decidedly insane difficulty, Shadow Gangs is a title worth trying.
The physical version of the game can be purchased for £29.99 and the digital version for £19.99.
A title suitable for true hardcore gamers.

by **Giampaolo Moraschi**







NEW GAME

GREAT GIANA SISTERS

Year: 2024
Editor/Developer: TCFS, Unreal, Csabo
Genre: Platform
Platform: Plus 4
Website: https://plus4world.powweb.com/software/Giana_Sisters

Everyone knows Great Giana Sisters, especially Amiga and Commodore 64 owners.

Whether it's because of its similarity to Super Mario and an alleged lawsuit with Nintendo, Chris Huelsback's wonderful soundtrack, good gameplay, or beautiful level design... in short, it's impossible not to know this title, and even more difficult to have never played it.

A classic 2D side-scrolling platformer starring two sisters grappling with crystals to collect, monsters to blast, and magical items that transformed them from "chubby little girls" into angry punks.

A whopping 33 levels of frenetic gameplay and lots to explore, see, jump, and smash. An instant hit that has generated a large fanbase over the years.

It was originally released on the 64 in 1987 and the following year on Amiga, CPC, and Atari ST. Programmed by Armin Gessert, with graphics by Manfred Trenz and a soundtrack by the aforementioned Sid guru Chris Huelsbeck under the Time Warp label.

In 2014, after sequels, clones, re-releases, and remakes, here we are once again talking about the original title, and we're doing so on the Plus 4, the least understood home computer from Commodore itself, which is enjoying a second youth thanks to the Hungarian developer scene that has churned out new titles and unthinkable conversions of stunning quality over the last two years.

Giana is on Plus 4 and it's... beautiful!



A perfect conversion. All the levels, all the monsters, and all the madness of the C64 title are there.

The graphics move very well and are not affected by excessive slowdowns or choppy scrolling. It appears more colorful and pleasing thanks to the Plus 4's palette.

The audio reproduces well what is heard on SID but with the limitations of the TED sound channels.







OUR FINAL SCORE

» Gameplay 90%

The colorful world of Giana Sisters is here, and it really moves with style even on Plus 4.

» Longevity 95%

33 levels of gameplay and great gameplay.





A perfect job by the team led by independent developer TCFS and his team consisting of Unreal on graphics and Csabo on music.
Get it and play it because it's worth it even in its new/old guise on Plus/4.
Sensational!

by **Carlo Nithaiah Del Mar Pirazzini**





KING OF DEMONS

Year: 1995/2024
Editor/Developer:
Genre: Action/Platform
Platform: Super Nintendo
Website: http://retro-bit.com/majyuo/

1995 was a hotbed of great titles for the Super Famicom. Games of all genres and even several products with horror settings.
Majyuo (original title) was one of the titles belonging to this wave of splatter titles. A title that has never been localized into English, at least until today.
On the occasion of its release thanks to Retro-Bit, let's talk about one of the most adult action games for the Super Nintendo.
Just think, for years Nintendo's 16-bit console was considered a machine with titles for families or almost. Sunny and optimistic titles, great RPGs and action games. Yet in its vast game library, it is possible to find some horror gems intended for a mature audience.
This King of Demons is one of them. KSS has given us a raw action game with a strong storyline.
Abel is the protagonist of the adventure. A young hero betrayed by his best friend, who sold his soul to the demons. The scoundrel sacrificed our hero's poor wife, daughter, and even his little cat in order to awaken Lucifer, the king of demons, from his eternal sleep.
Abel decides to take revenge but fails. The game begins with Abel's death at the hands of the demonic Bayer, but the spirit of his murdered wife brings him back to life, and this is where the adventure really begins.
Hell is a lively place, represented by seven levels that are as varied as they are detailed.



The backgrounds, for example, are simply breathtaking.
The hero moves satisfactorily along the platforms with his double jump and sliding abilities (the latter being essential).
However, our hero's human form alone is not enough against the demons; to the rescue, we can use the ability to mutate after each end-of-level boss battle. Once mutated, Abel becomes







a strange construct reminiscent of Devilman.

This is an interesting addition to the game, allowing you to tackle some levels that would be impossible in human form.

It is worth noting that this is a very dark game. There is no joy or excitement, there is a lot of gore, and the plot is very melancholic.

As already mentioned, the technical aspect is definitely of a high standard, but the game suffers from a short lifespan. The seven circles of hell are few, and in about half an hour of practice, you can beat the game.

Special mention goes to the soundtrack, which contains some memorable tracks by Hiroshi Lizuka and Tomohiro Endo. They make hell an even more disturbing place.

Over the years, the title has become a cult classic, and it is finally possible to play it in physical format after enjoying the 2005 translation in an interesting patch.

However, it is a title that deserves to be rediscovered and perhaps added to your game library in physical format.

by **Roberto Del Mar Pirazzini**



OUR FINAL SCORE

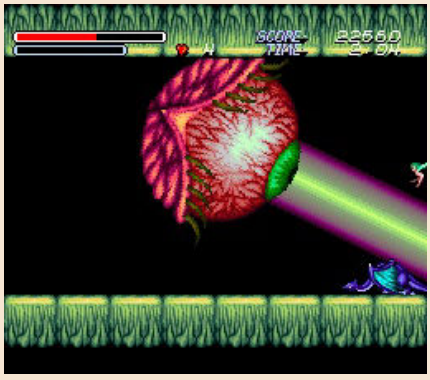
» Gameplay 90%
Excellent control system and nice level design. The game is very easy to explore.

» Longevity 75%
There are too few levels. I wanted to play more and more...









GO FOR IT! GOEMON
THE GRAND CYCLE OF OEDO

Year: 2001
Editor/Developer: Konami / Accadiez
Genre: Platform/Adventure
Platform: Playstation 1
Patch uk: https://www.romhacking.net/translations/7256/

So Japanese and so impossible not to love! This phrase perfectly sums up what I think of the Ganbare Goemon series.

A series that began on MSX and NES and was later brought to prominence on Super Famicom with four excellent chapters.

Today, it's time to talk about this chapter for the first PlayStation, which recently received a juicy patch to be played in English.

The game is about the terrible villain Ecorori who kidnaps the beautiful Omitsu. It is then up to the brave Goemon and his group of misfit friends (the obese ninja with bisexual tendencies Ebisumaro, the punk robot Sasuke, and the beautiful Yae) to save her.

The game is heavily inspired by the second chapter seen on Super Nintendo, which we reviewed in issue 18 of RMW Italian. We will therefore have some levels of pure horizontal scrolling platforming, accompanied by giant bosses in first person, followed by exploratory levels in cities and villages.

There are appearances by the crazy characters from throughout the saga and some faces seen in the Nintendo 64 chapter, all seasoned with a good dose of humor that is omnipresent in the series.

Compared to the Nintendo 16-bit chapters, this game uses pre-rendered graphics that did the job at the time. Nowadays, it looks rather strange and, like many other 3D productions of the time, it hasn't aged very well.



However, the title is colorful and chock-full of objects, monsters, and platforms everywhere.

The soundtrack from the first chapter of the title and Monoke Suguroku (the





title for Nintendo 64) is particularly enjoyable.

If I had to find a definition for this PS1 version, I would say that it is a "best of" compilation of the best elements of the saga, seasoned with the proverbial gameplay and beautiful level design in pure Konami style.

The translation by Accadiez makes the game more enjoyable and can be applied to the title (via a ROM patch, provided you have an original) using the classic xdelda, a lightweight application that can be downloaded for free from the internet.

The Japanese language barrier was never an obstacle in the original version, but reading the text in English makes the plot flow better and facilitates laughter.

A title to rediscover for your PS1 and to try in this version in the language of Albion.

by **Roberto Del Mar Pirazzini**



OUR FINAL SCORE

» Gameplay 90%

Beautiful level design and a great job in terms of gameplay in pure Capcom style, which, thanks to the patch, we can enjoy even more.

» Longevity 80%

It's one of the titles that can be completed most quickly, but it's a great game to play.









NEW GAME

ALICE SISTERS

Year: 2024
Editor/Developer: PPixel Heart/Orion Soft
Genre: Platform/Puzzle
Platform: Atari Falcon
Website: https://orionsoft.itch.io/alice-sisters

In December 2022, our dear Ingrid had already reviewed Alice Sisters for the Sega Megadrive, giving it 90% and 95% for playability and longevity.

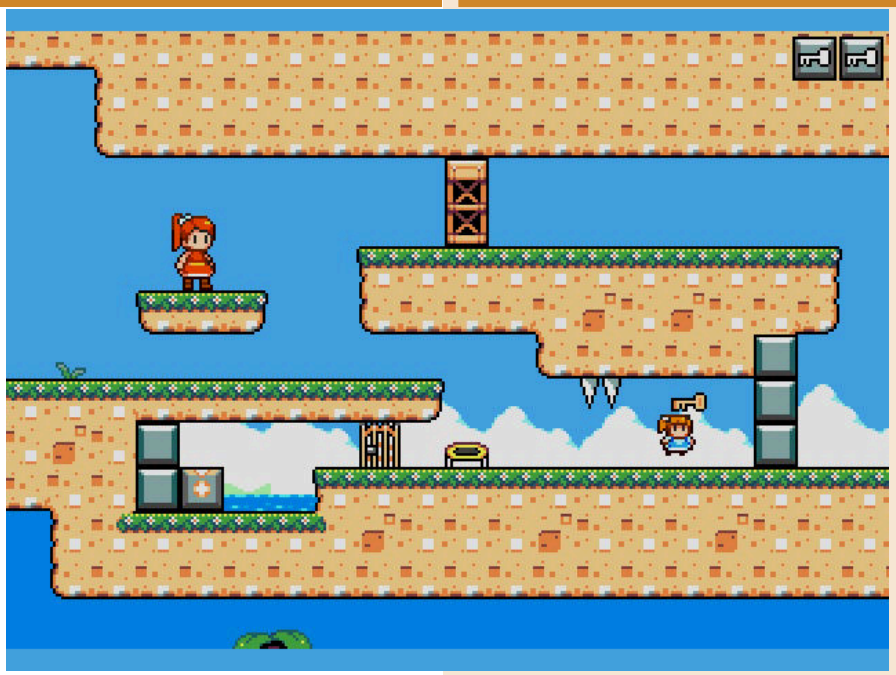
She was enthusiastic about the title because of its gameplay and the many levels it offered.

Two years later, the version for Atari Falcon, an unfortunate personal computer developed by Atari Corp in 1992, arrived. It was a little-known platform with unique features, killed by its exaggerated cost and a strange 32-bit interface with a 16-bit bus that ensured compatibility with ST software but somewhat clipped its wings for the "future."

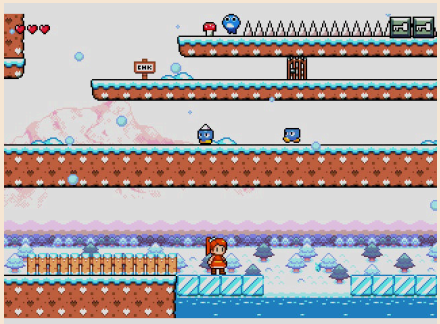
But enough chatter, let's get to the game.

An evil ogre has captured the mother of two cute, chubby little girls. Our task is to help them through 28 levels spread across 4 game worlds to rescue 'parent 2' and bring her home.

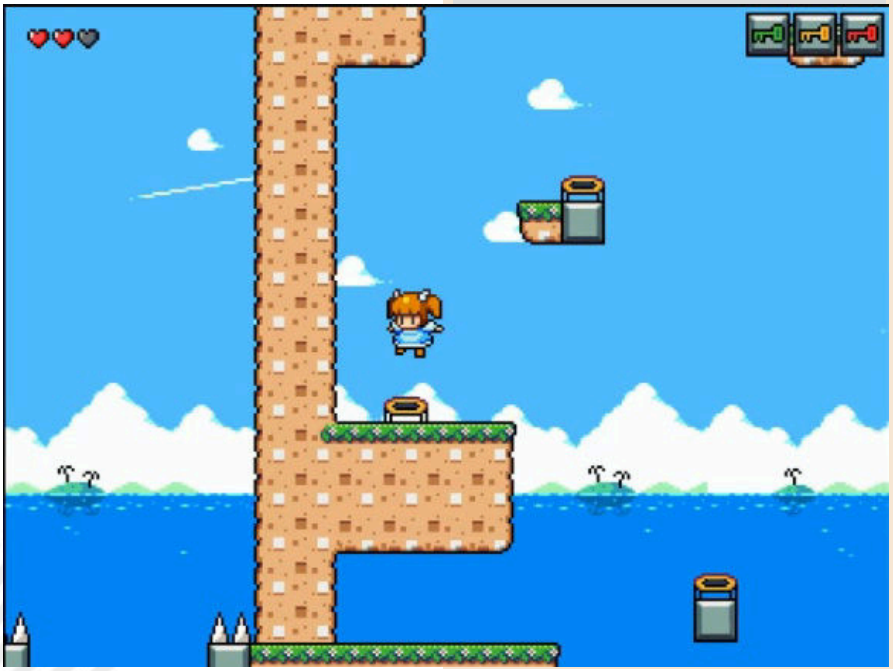
We control both girls, who 'switch' by pressing the joystick down. The smaller, chubbier one can change size using magic mushrooms (don't get any ideas) and jump deeper, while the older one can defend herself by throwing a red ball at enemies.
The switch sets in motion combinations that are essential for completing the

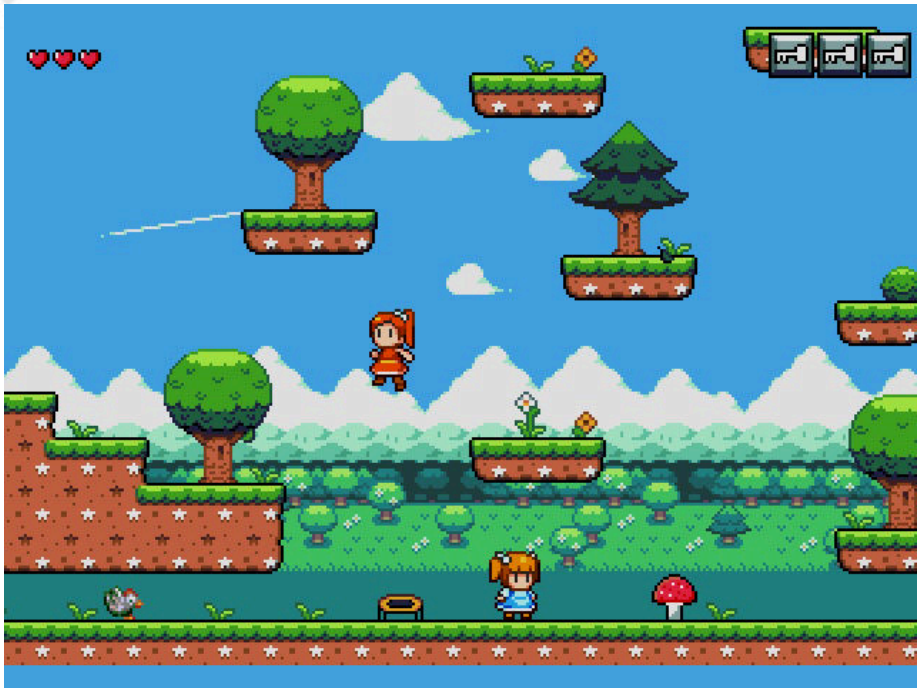


level and moving on to the next one.



As with the MD version, the FALCON version is also beautiful to look at and listen to. The graphics are pleasant and very colorful. The characters are well characterized and move well.





OUR FINAL SCORE

» Gameplay 85%

The use of the joystick is initially disconcerting (you jump by pressing up and not, as on the Megadrive, with the second button), but the game is well developed and fun.

» Longevity 95%

Lots of levels, 4 game modes, and several hours of gameplay.

It's a fun title that keeps you glued to the screen thanks to its well-balanced difficulty and well-designed levels.

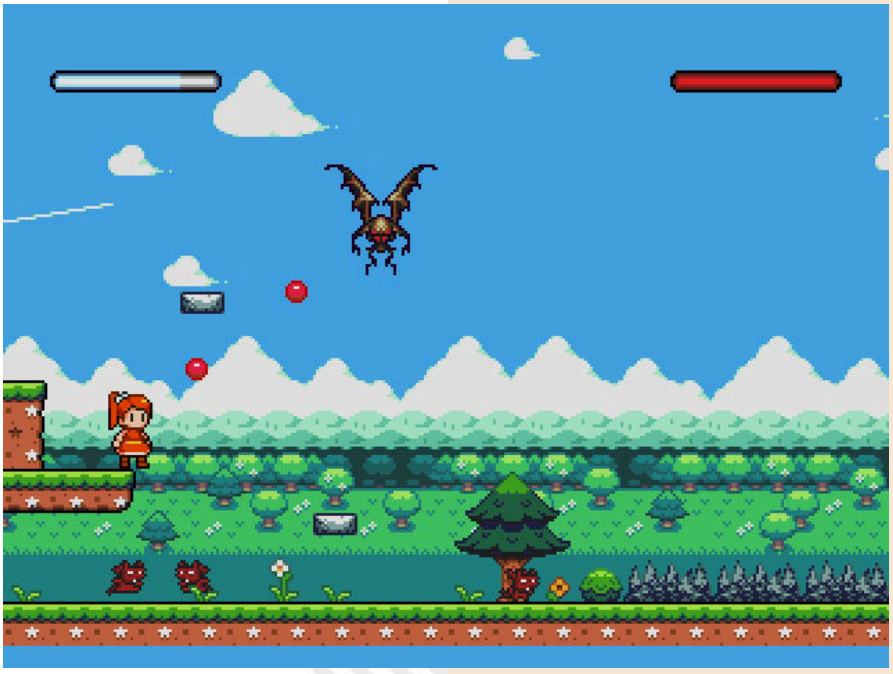
This FALCON version is paid and costs $12. The file you purchase also includes the PC version (identical).

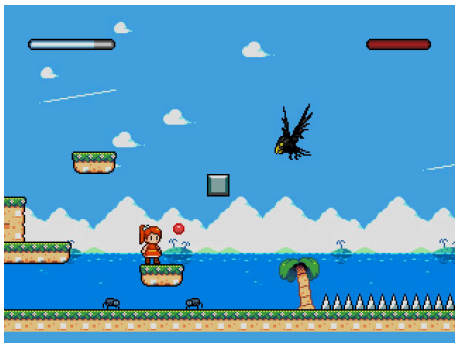
The game requires 8MB of RAM and installation on a hard disk. It is also compatible with the Hatari emulator.

All that remains is for me to confirm the validity of the game and recommend that you purchase it and support OrionSoft.

by **Giampaolo Moraschi**







NEW GAME

TETRIS

Year: 2024
Editor/Developer: Eric Carr
Genre: Puzzle
Platform: Atari 8bit (800 48k, XE/XL)
Website: https://bunsen.itch.io/tetris-atari-8-

The annual Basic 10Liner competition celebrates its 13th edition in 2024. This competition for developers focuses on short lines of Basic code and is open to all 8-bit computers.

The 2024 edition closes at the end of March 2024.

You can participate in multiple categories:

PUR 80 – Game on 10 lines, max 80 characters per logical line.

PUR 120 – Game on 10 lines, max 120 characters per logical line.

EXTREM 256 – Game on 10 lines, max 256 characters per logical line.

SCHAU: Game on 10 lines, max 256 characters per logical line.

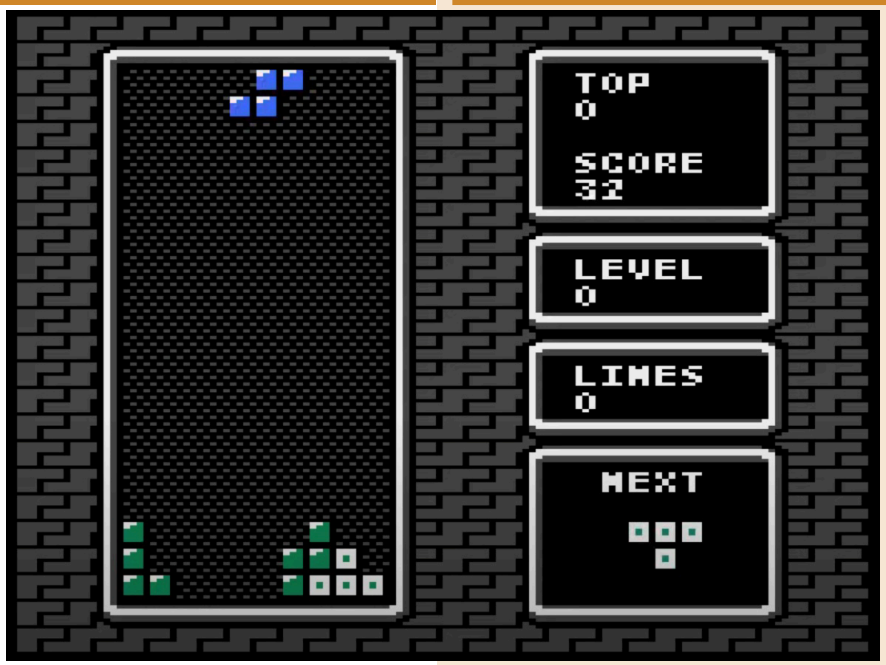
The first three categories are exclusively for Basic games, while SCHAU can be anything: a demo, a utility, or an application program.

Programs submitted that do not fall into any of the categories are grouped into the WILD category.

Retromagazine has also participated in the competition in the past with its editor and Sinclair enthusiast Alberto Apostolo (we miss you in the editorial office, Albe!!!).

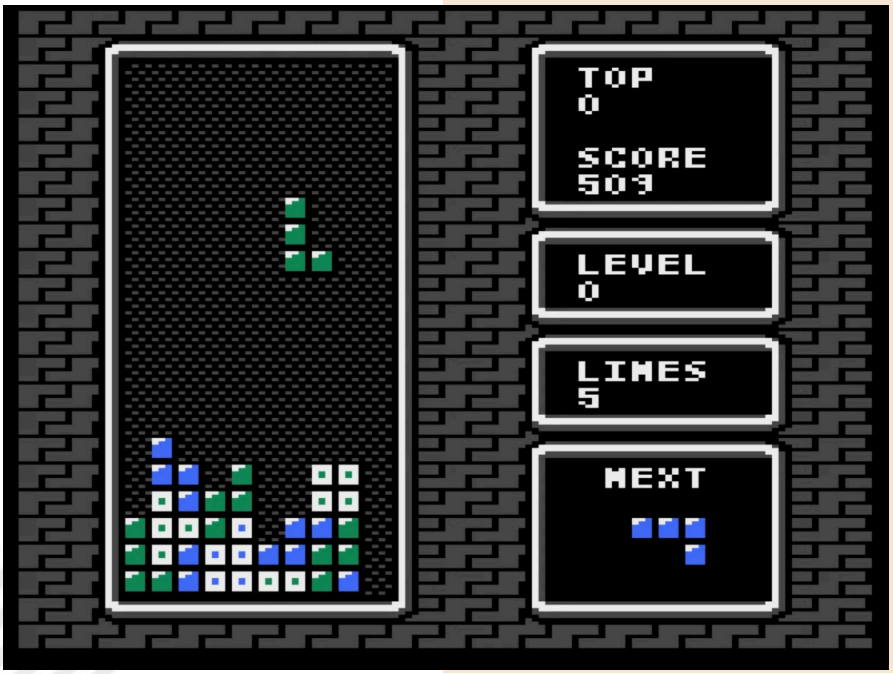
It is an event that I personally love and that always showcases the ingenuity of the many participants.

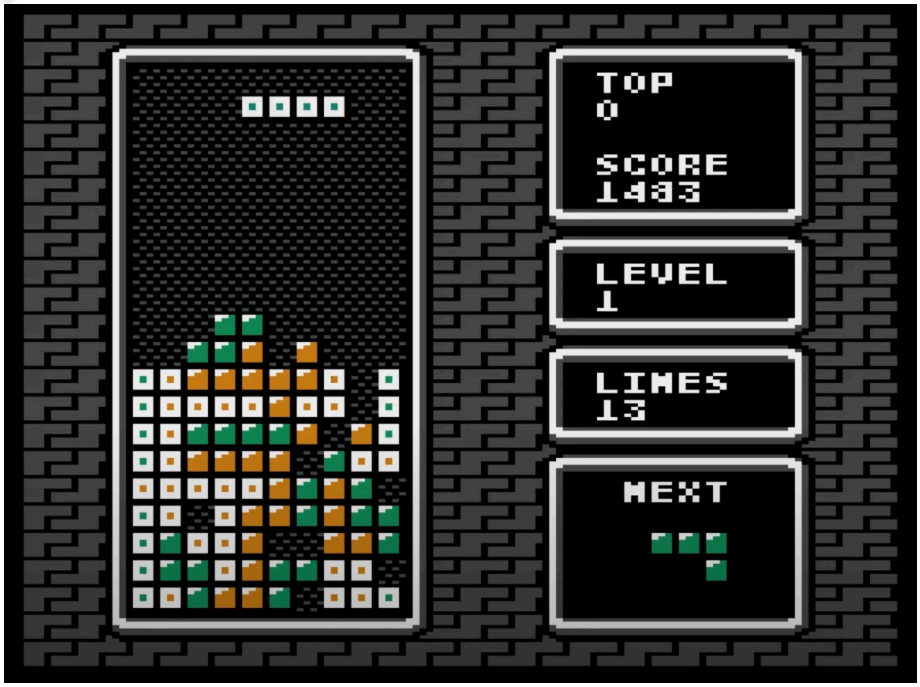
Eric Carr's Tetris is an example of how to do things right in just a few lines of code.





The game needs no explanation or introduction. It's Tetris, so unless you've been living in a volcano for the last 40 years, I won't go into detail. What I can say for sure is that it's a really well-made version. It was





OUR FINAL SCORE

» Gameplay 90%
Can 10 lines be perfect? Yes. Here you have all the magic of TETRIS and a great tribute to the NES version.

» Longevity 99%
Eternal.

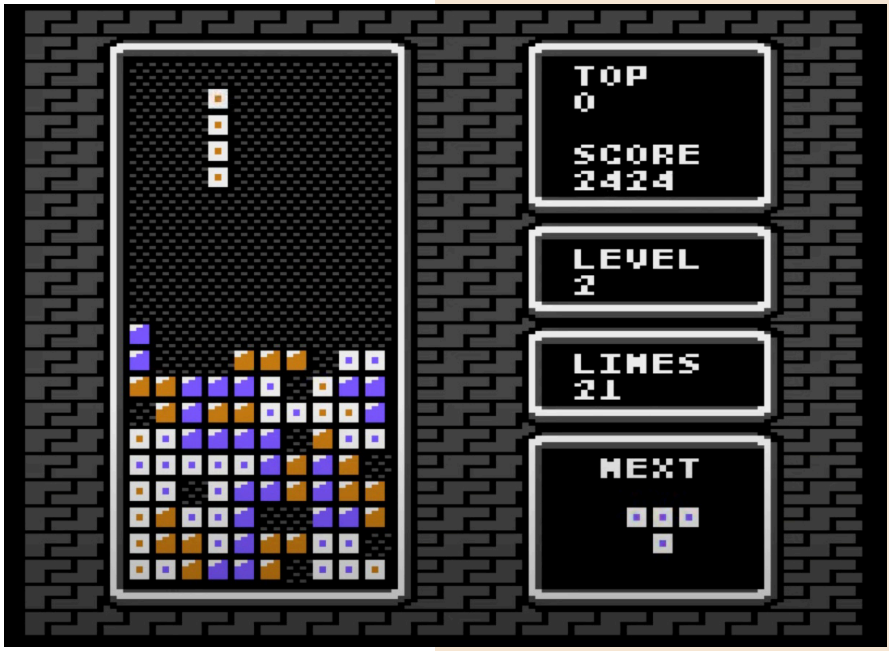
inspired by the Nintendo NES version. A custom character set, music code written from scratch, a FastBasic 4.6 compiler to compile to XEX, Visual Studio for the code, and a lot of passion were used.

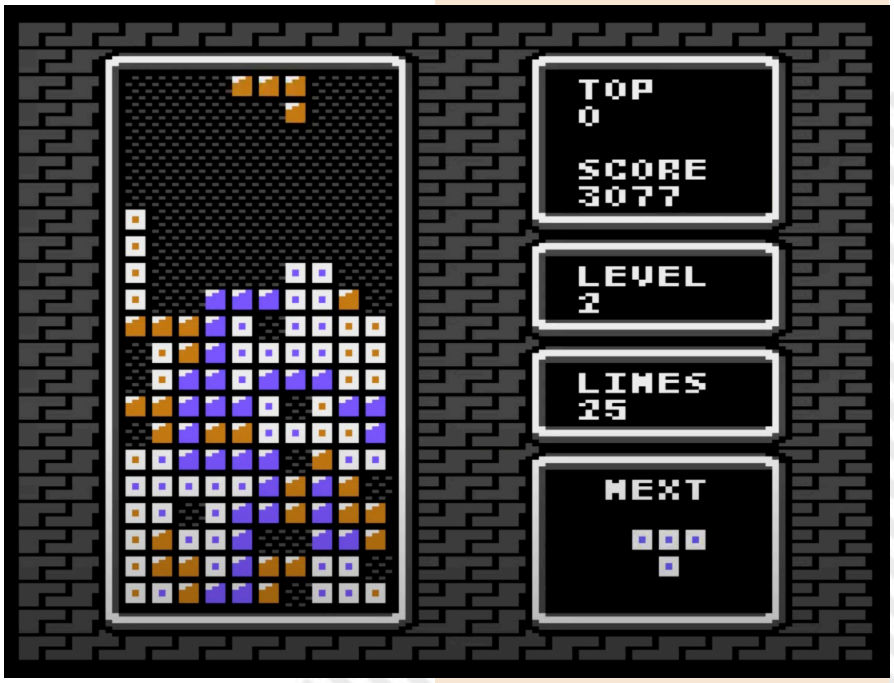
A great Tetris, perhaps one of the best I've played in years!

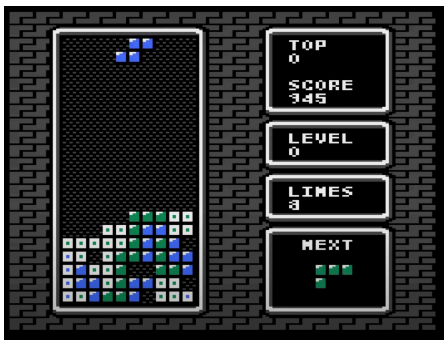
Fast, fun, and practically timeless.

Yet another demonstration of a community, that of retro developers, alive and capable of doing impossible things... in just 10 lines!

by **Carlo Nithaiah Del Mar Pirazzini**







NEW GAME

XIAOMEI AND THE FLAME DRAGON'S FIST

Year: 2023
Editor/Developer: PiXEL
Genre: Beat em up
Platform: Sega Megadrive
Website: https://leoful.com/xiaomei-and-the-flame-dragons-fist-en/

Developed by Japanese designers PiXEL, Xiaomei and the Flame Dragon's Fist is a tribute to old 8- and 16-bit side-scrolling fighting games.

There's a bit of Kung Fu Master in this title, but just a little... a little bit.

The action takes place on side-scrolling levels where hordes of ninjas and swordsmen continuously rush towards Xiaomei. Most require one or two hits to defeat, with the occasional mini-boss having more advanced attack techniques. Each level ends with a tricky boss that will require skill to figure out how to take down.

You can strike with kicks and punches, and you can attack by jumping up and combining with your preferred strike button.

As you attack the enemy, a special meter builds up, allowing you to launch a powerful fireball capable of clearing the screen of enemies.

Xiaomei is a difficult game by design, even in easy mode. It certainly pays homage to the glorious games of the past and kung fu movies, but it does so with embarrassing technical shortcomings.

Poor animations, terrible use of color, and very little variety in opponents and levels. There is little charm in the protagonist's character, and some choices regarding sprite collisions are, how shall we say... peculiar!

Shall we talk about the audio department? Here too, there is cause for tears. An uninspiring soundtrack and terrifying sound effects.

On top of all this, the gameplay is wooden. The incorrect use of the



Megadrive buttons (one of which is completely inoperative) is compounded by the banality of the game levels and the decidedly wrong difficulty level.

As a dear friend says... "It looks like Kung Fu Master, but better!" (followed by loud laughter, of course).







OUR FINAL SCORE

» Gameplay 30%
The audio/visuals are poor, but the real sore point is the terrible gameplay and absurd lack of ideas.

» Longevity 10%
I had fun when the screen was still off and the console... too.



The Megadrive has lots of side-scrolling fighting games, and almost all of them are of a high standard, or at least decent. Give this one a miss. What's more, the cartridge is very expensive.
Leave it there.
P.S. The title is also available for new platforms...

God forbid!

by **Carlo Nithaiah Del Mar Pirazzini**



NEW GAME

HARHARGON

Year: 2024
Editor/Developer: Icon 64 – Psyktronik
Genre: Action/Platform
Platform: Commodore 64
Website: https://www.psytronik.net/newsite/index.php/c64/147-harharagon

A spy-themed game that reminded us a lot of Impossible Mission in terms of structure and design, and which presented itself as one of the "budget games" of yesteryear.

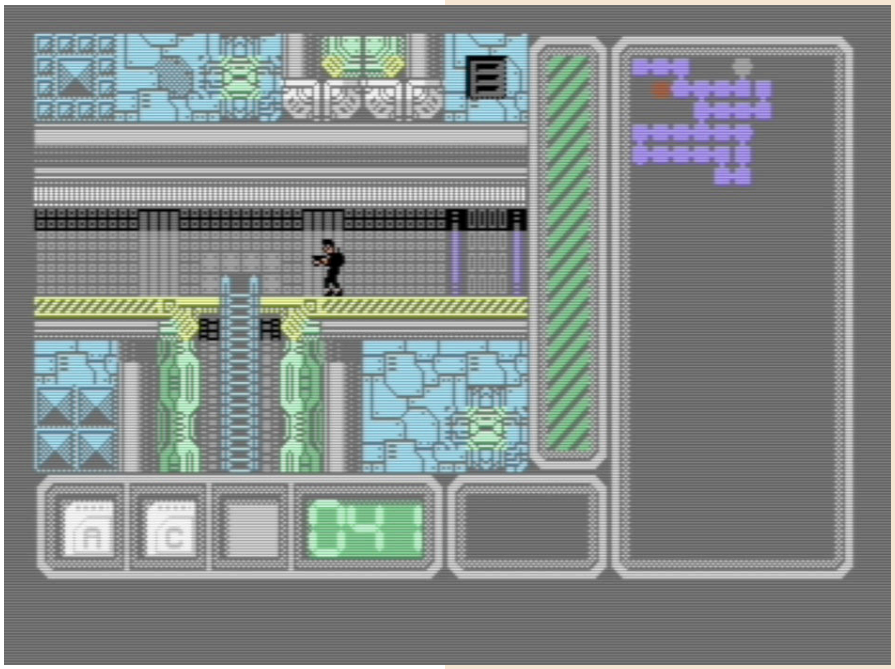
Harharagon puts us in the shoes of a prisoner in a maximum security prison. We have been convicted of a crime we never committed, simply because we had a relationship with the wrong person's daughter at the wrong time.

We find ourselves inside a prison/maze and our goal is to explore it, recover 18 pieces of the escape pod, and escape.

The prison has 230 rooms and it won't be an easy task to explore them all. Numerous cybernetic guards will deal with anyone who attempts to escape... by any means necessary!

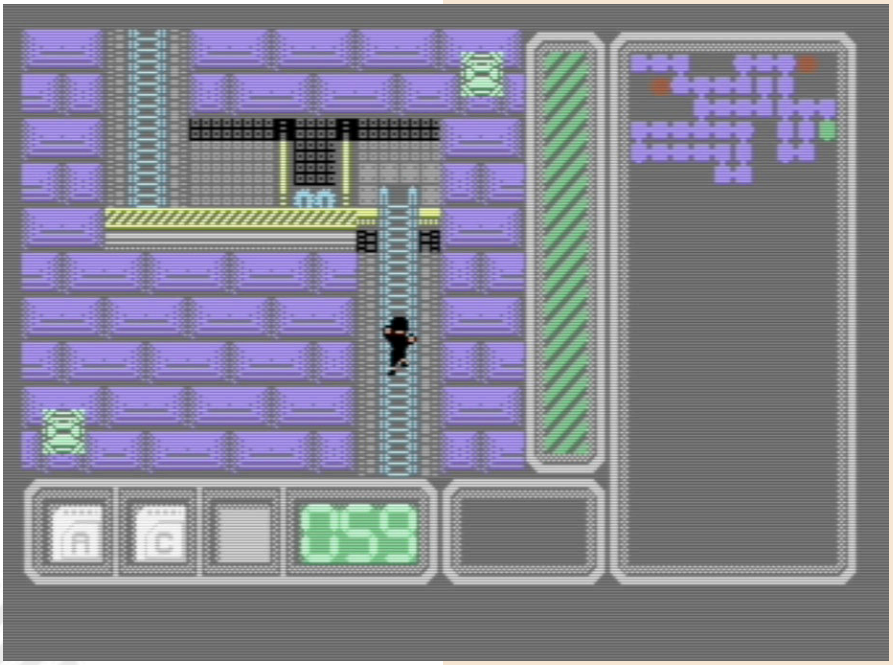
As if simple exploration, searching for missing pieces, and enemy robots weren't enough, our hero must also recover crystals that allow him to upgrade the firepower of his weapon. The crystals are dropped by destroyed enemies or recovered along the way. Icon64 has created a very solid run 'n' gun game that provides about 90 minutes of arcade-style gameplay.

The high-resolution graphics are functional and very attractive, and the character moves well and with detail in the animations.



The 8-direction firing mechanism is well developed, allowing you to effectively hit opponents who will mercilessly descend upon you.

Saul Cross's music provides a single





OUR FINAL SCORE

» Gameplay 80%
A well-designed title in terms of structure. Simple controls and a good firing system for the protagonist.

» Longevity 80%
Offers a good degree of challenge with its 230 rooms. Some are repeated, but we don't even notice.

melody with strong 80s influences. It is pounding and well developed, making full use of the SID.

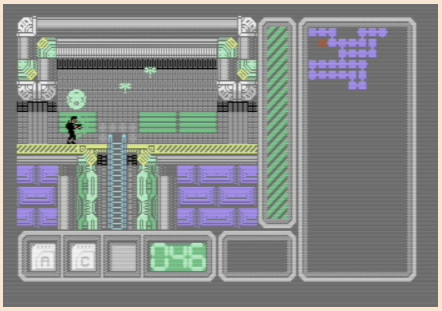
The game is easy to learn and provides an increasing degree of challenge. The in-game screen shows the action with the item menu at the bottom and the prison map on the side, which, once explored, will show us the path taken, the exits, and the areas containing items and so on.

If we had to find a classic fly in the ointment, we could say that some rooms are repetitive in terms of layout and enemy reproduction. Nothing major in terms of playability and fun.
There is a lot to appreciate in Harharagon.

It is well made overall, even if it offers nothing new in the C64 genre.

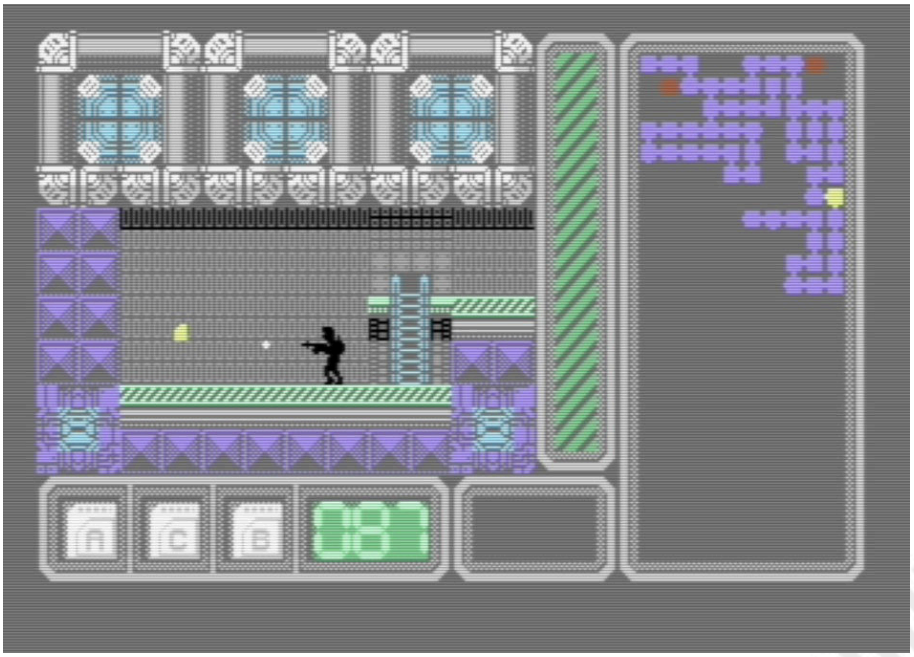
However, it is fun and you are sure to enjoy it.

by **Carlo Nithaiah Del Mar Pirazzini**









NEW GAME

TRUCHO

Year: 2024
Developer: TheNestruo
Genre: Action game
Platform: MSX
Website: https://thenestruo.itch.io/trucho - https://msxcartridgeshop.com/

Great! Great! Great! Trucho is a simple game, but well-designed in every respect and also fun.

Inspired by PC and console "fishing games," this title puts us in the shoes of a pixelated fisherman competing in a series of competitions.

Actually, there are three fishermen to choose from, each with their own casting skills and all well characterized.

Once we have selected our character, we are catapulted into the competition area and the action begins.

We select the type of bait based on our skills and cast our bait, paying attention to what is happening at the bottom of the game screen.

At this point, several fish (with different characteristics) will appear, and it will be up to us to get them to bite our hook. This is not as easy as bringing them back to the boat.

Once we have caught the required number of fish to complete the level, we move on to the next one, and the difficulty gradually increases.

It sounds simple on paper, but I assure you that you will enjoy it as much as any modern fishing game.

Fishing requires patience, and so does Trucho. Waiting for the right moment and then pulling your prey into the boat requires precision in managing the power bar with your joypad/joystick... and I forgot to mention that there is also a time limit for completing the level.





The work done by TheNestruo is well done and works great. I really liked some of the color choices and the ability to use certain techniques in the game, such as attracting the fish's attention to the bait.





A game developed with technical expertise and also featuring a nice soundtrack and good sound effects.
Of course, it's a game aimed at fishing enthusiasts or those looking for something without too many pretensions, but it's definitely worth trying.
The title is available for purchase in two versions: a physical version, accompanied by a nice manual, a really nice package, and a gadget (in addition to the cartridge); and a flash rom version for only $5.26 with the files needed to play.

by **Carlo Nithaiah Del Mar Pirazzini**

OUR FINAL SCORE

» Gameplay 90%
Simple but comprehensive game controls and truly engaging gameplay that keeps you hooked on the challenge.

» Longevity 80%
If you love this type of game, it's perfect. If you're looking for something more "dynamic," this may not be the best title for you. However, it remains a really well-made game with an excellent difficulty curve.





NEW GAME

GOOD KNIGHT

Year: 2024
Editor/Developer: Psytronik Software - Icon64
Genre: Platform
Platform: Commodore 64
Website: https://psytronik.itch.io/good-kniight

Here we go again... the usual fire-breathing dragon suffering from loneliness and delusions of grandeur has kidnapped the princess (on duty). The king is very worried and doesn't know who to turn to. On Commodore 64, all the heroes are busy on heroic missions to save worlds, princesses, and themselves, and at the moment, no one is available (Sir Arthur from Ghost 'n Goblins... let's not even talk about him!).

All that remains is to turn to Sir Darren of Durham, known as the Good Knight. He's definitely a good guy, but definitely not the most heroic.
He is a warrior more suited to battles at the table against first courses, second courses, and desserts than to fighting monsters and bizarre creatures.
But life is hard, and beggars can't be choosers, right?
In any case, Sir Darren accepts, and the adventure begins.
Good Knight is a well-made platform game reminiscent of Hunchback.

Sir Darren moves from left to right in single screens. Each level presents a different type of challenge: platform jumps, fireball throws, rotating saws, enemy knights, monsters, poisoned arrows, and so on and so forth.
To his advantage, he can jump to avoid all this, use platforms, ropes, and his general mobility.
All this until he reaches the evil dragon holding the princess hostage.
There are about 40 levels (with some bonus screens) to tackle for a challenge that, once learned carefully, will take about 30 minutes.
The graphics are simple but well-designed and carefully animated. The color scheme is also beautiful.



A medieval-style SID soundtrack accompanies us throughout our "battle," but there are no sound effects whatsoever.
Good Knight is a "bastard" game! It looks simple but is lethal. If you miss a jump... you die! If you misjudge the timing of your opponent's weapons... you die! If you rush into everything without thinking... you die!
It requires quick reflexes and memory







OUR FINAL SCORE

» Gameplay 75%
In its disarming simplicity, it is a well-structured title and decidedly lethal in its method of learning the levels.

» Longevity 75%
You die often, very often, but the desire to see all the levels is so strong that you can't easily put down the controller.

for each level. These two things are essential.
However, this lethality does not affect the fun; on the contrary, it increases the desire to continue and see what lies ahead.

The title is available on the developers' itch.io portal for $3.99. An excellent budget price.
It runs perfectly in real life, in emulation, and on the THEC64 family.

My advice? Get yourself a good gamepad... and a looooooot of patience!

by **Carlo Nithaiah Del Mar Pirazzini**









NEW GAME

SAVAGE PRINCESS 2

Year: 2024
Editor/Developer: Monster's Legs
Genre: Platform
Platform: ZX Spectrum
Website: https://monsters-legs.itch.io/savage-princess-2

The authors of this saga are not lacking in humor.

After the exertions of her first adventure, the wild princess goes to the pub.

Here she indulges her gargantuan appetite and thirst to the point of getting (well-deserved) drunk.

While she is sleeping soundly, she is abruptly awakened by Eldritch and the screams coming from outside the inn.

Cthulhu has awakened! And he is furious! Because they have awakened him from his eternal sleep and because he is a Great Old One who aims to conquer the world.

Ava, the wild princess, thus sets off on an adventure with the aim of driving the tentacled monster back into the depths of his city of R'lyeh... obviously after swearing like a dockworker...

I repeat, the authors are either jokers or madmen, but in any case, Savage Princess 2 is fun.

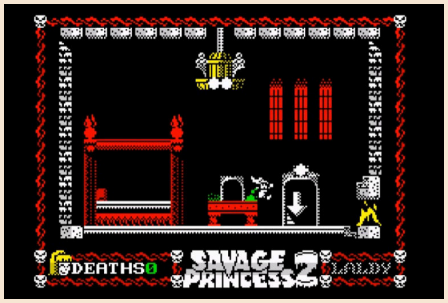
A simple, carefully animated single-screen platformer with cute graphics that is easy to master.

Easy to learn and with a gradual difficulty level.

To continue the adventure, you need to collect five tokens: the first three

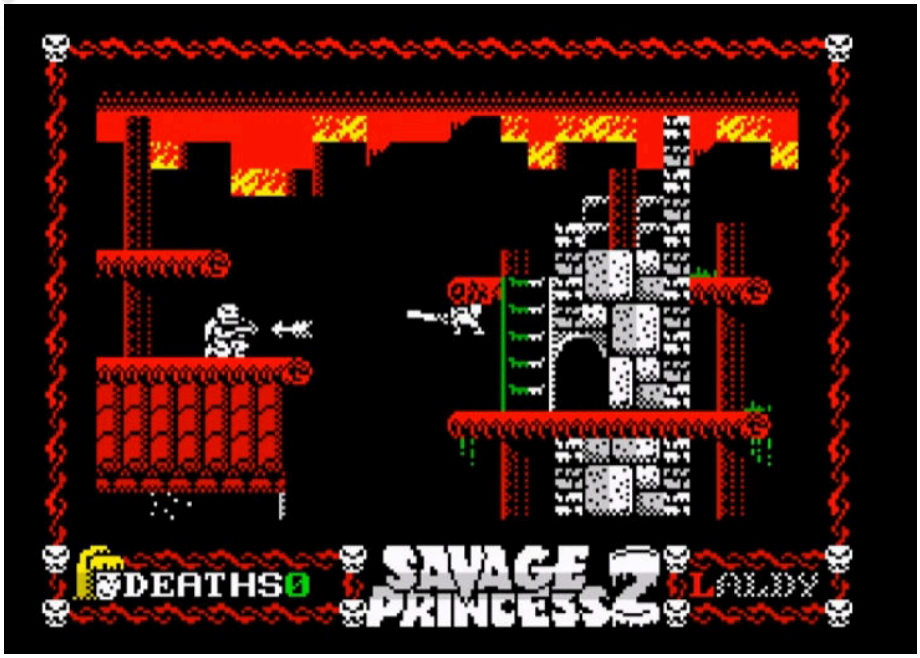


are in the first level and one token in each of the following levels. Only once these coins have been collected will Ava be able to reach the titanic Cthulhu.



Excellent use of MPAGD and a nice idea to be scolded (insulted) every time you lose a life.





OUR FINAL SCORE

» Gameplay 85%
The title is well developed in all respects and is enjoyable. The crazy humor is its strong point.

» Longevity 80%
It's not very long, but it's very commendable.

The file is .tap and can be loaded onto any ZX Spectrum emulator without any problems or played on real hardware.

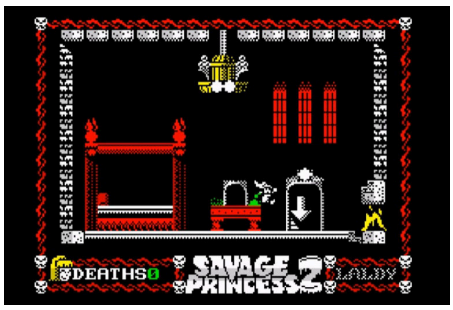
If you want to have a laugh and play a fun game, Savage Princess 2 is for you.

by **Giampaolo Moraschi**









NEW GAME

GODZILLA BALLS

Year: 2024
Editor/Developer: Paxanga Software
Genre: Puzzle/Action
Platform: MSX
Website: https://paxanga.itch.io/godzilla-balls

The monstrous Godzilla has just turned 70. Since his first appearance in the 1954 film, he has been crowned King of the Monsters, and no one has been able to dethrone him since.

He was also one of the first Kaiju to appear in Japanese science fiction cinema. The term refers to giant monsters born as a result of radiation from atomic tests and American bombing of Japan during World War II.

The concept behind this mythical creature is simple: humanity has unleashed a power it cannot control, and its hubris has led to its downfall. Its birth is spiritually linked to the regret of having created nuclear weapons, but over time it has evolved, changing several times over the decades.

Its success led to the birth of Monster Island and many rivals/friends over time, including Gamera. A mysterious monster born in 1965 as an antagonist; a giant toothed turtle capable of breathing fire and flying.

Over time, it has become a favorite of children, so much so that it has been featured in countless advertisements, manga, and even a real family of sea turtles.

Now our gargantuan monsters are facing a big problem. The evil Professor Shorizawa has reduced them to small, chubby creatures and put them in a huge labyrinthine factory.

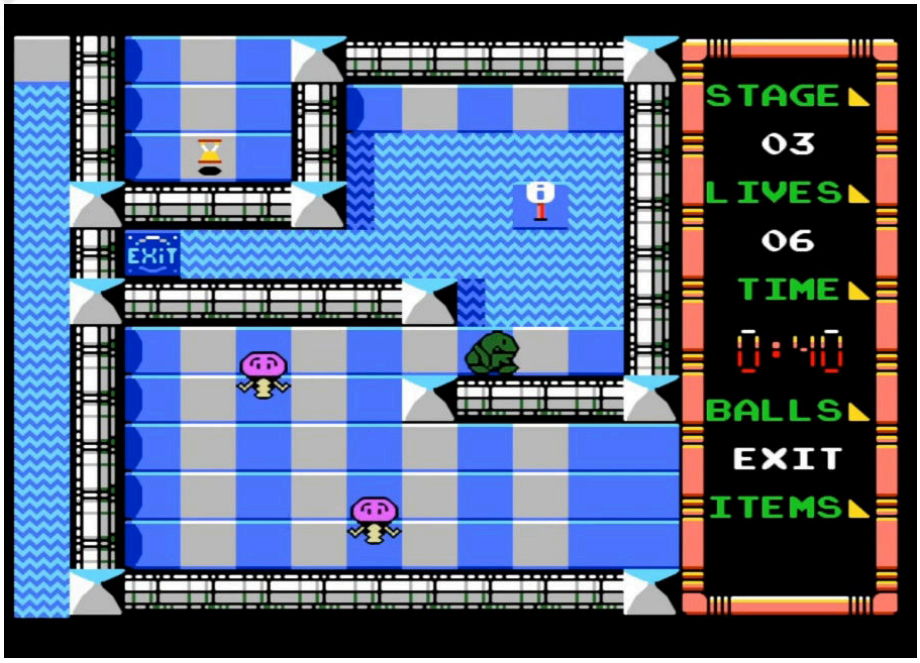
In order to return to their former selves, the two friends/enemies must work together and recover small spheres scattered throughout the game levels within a time limit. This is the only way to become giant again, free and happy to destroy everything in their path.





Moving around the game is simple. Just collect the balls, avoid the





OUR FINAL SCORE

» Gameplay 90%
Simple and stimulating in its learning curve and gameplay. We really like the in-game collaboration.

» Longevity 90%
A title that can be played without too much hassle and with the right degree of difficulty.

monsters (which can only be stunned), follow the advice, collect the passwords, and work together. Some levels can only be completed if Godzilla gets help from Gamera and vice versa.

A fun and stimulating idea.

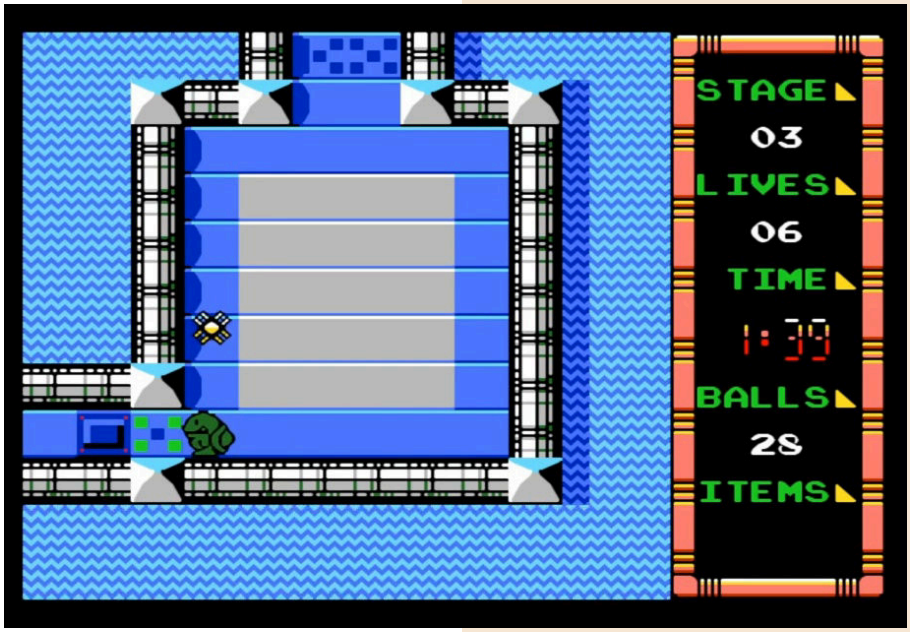
The title is fun and well developed. There is a wonderful use of colors and the sprites are well characterized and in pure "Super Deformed Kawai" style. The music is also well done and quite catchy.

The game mode is unique and there are no difficulty levels, so we strongly recommend reading the manual files contained in the game ROM.

Godzilla Balls runs on MSX1, MSX2, 2+, and TR, uses 16kb of RAM, and is a 64kb ROM. A physical edition is also planned for release by the end of the year.

A nice puzzle game with action elements that is worth trying and playing. Great work by Paxanga Software!

by **Giampaolo Moraschi**





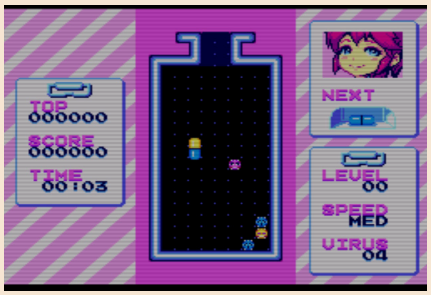
NEW GAME

MAI NURSE

Year: 2024
Editor/Developer: Lunoka
Genre: Puzzle game
Platform: Game Boy, Pc Engine Turbo Grafx 16, Sega Game Gear, Sega Master System
Sito: https://itch.io/queue/c/630067/game-boy-games-console-retro-homebrew-gbc?game_id=2603919

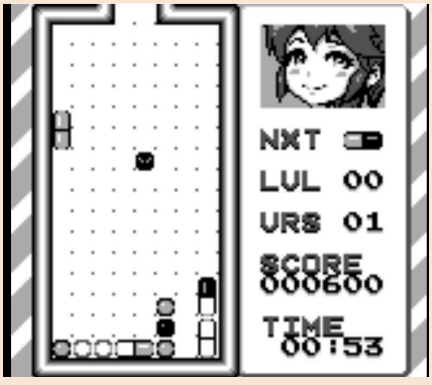












The instructions for this game say, "Help Mai get rid of viruses with pills, line up 4 slots of the same color horizontally or vertically to destroy them, and progress through the levels of this classic puzzle game!"
Does that remind you of anything? Of course!
It's Dr. Mario, an absolute icon of classic Game Boy games.
Mai Nurse is a perfect clone in terms of gameplay mechanics. Same grumpy viruses, same type of pills to line up. Only the music and the protagonist have changed.
Enough with that mustachioed plumber dressed up as a doctor! Here comes the busty and kawai Mai.
A fun title developed for multiple platforms and created with GBDK2020, a development platform for sm83, Z80, and 6502.
A shameless clone, but still fun and available in the future on Megadrive, Snes, Saturn, and GBA.
The developer is also considering releasing a physical version towards the end of 2024.
I tried both the Gb version (in real) and the PC Engine version (in emulation).
The game is simple and maintains the same level of interest as its illustrious 'father'.
I found the Gb version more enjoyable, ideal for carrying around between shifts at work or at university.
On the PC Engine, everything is beautifully colored and accompanied by a nice little tune.
If you liked Dr. Mario, you'll love this title.

by Marta Rossman

OUR FINAL SCORE

» Gameplay 90%
Retains all the simplicity and gameplay of the original title.

» Longevity 90%
Less incisive than Tetris or Puyo Puyo, but still a great title to play and replay.

NEW GAME

BOMB JACK
THE REMAKE

Year: 2024
Editor/Developer: MikGames
Genre: Platform
Platform: Sega Master System
Website: https://www.smspower.org/Homebrew/BombJackRemake-SMS









When I was young, I loved playing this game on the Mame at home.
I belong to a generation born after the arcade boom, but I've always been fascinated by it.

I'm one of those people who, ever since I was a kid, has rushed to the few arcades left at the seaside.
My passion for old games is thanks to my father and those who raised me on bread and "old school."

This 8-bit version of Bomb Jack for the Sega Master System is... a bomb! The game is as addictive as it was in the arcade and also has all the stages and features of the original, which in other cases has not been possible.

It's all there. The fast and furious game stages are perfect for those who love to brave danger while trying to defuse all the bombs on the platforms. All the foul monsters are there, as is the fluidity of the little red and blue superhero.

You can set the difficulty level and other options before embarking on the adventure.

I think it's one of the most entertaining versions of this title, perfect for loading onto your favorite portable device, even in emulation (I play it in emulation on my smartphone).

An unexpected gem created to participate in the contest organized by the SMS Power! page, a reference point for the lively community dedicated to the Master System.
Great!

by **Ingrid Poggiali**



OUR FINAL SCORE

» Gameplay 90%
A title with simple but deadly gameplay dynamics. You need to know how to move well between levels, but each completed level gives you a unique taste.

» Longevity 90%
A game to be approached with the spirit of the old titles of years gone by. One token (start in this case) leads to another.

NEW GAME

ARKON BLAZE

Year: 2024
Editor/Developer: VoxVideogame
Genre: Shoot em up
Platform: Commodore 64
Website: https://voxvideogame.itch.io/arkon-blaze-c64 - https://voxvideogame.itch.io/

Sensible Software and Commodore have been intertwined for years.

A fruitful partnership during the two golden ages of home computer video games.

In the 16-bit era, we had many beloved titles created by the British software house, two of which are engraved in the tablet of commandments of Amiga games: Cannon Fodder and Sensible Soccer.

But the love affair began long before, in the 8-bit era, when they produced a number of small masterpieces such as Wizball, Microprose Soccer, and International 3D Tennis.

Among their successes was also S.E.U.C.K., or Shoot'em up Construction Kit, a program that allowed the creation of video games on the C64, but also on the Amiga and Atari ST.

The C64 version was so popular that it created a real 'scene' and many young, budding developers. The glorious Italian Zzap! magazine of the time also devoted a lot of space to titles created with SEUCK.

It is now 2024 and the program is 37 years old (it was released in 1987), but there are still many developers who use it for their work.

Among them is Pinov and his Voxvideogame, which develops at a rapid pace for the C64.

Today, we take a look at the latest addition to the Voxvideogame family: Arkon Blaze.



The story begins in another dimension, in a world called Arkon. This planet was created by an evil being called "The Dark Lord," who discovered the essence and secret of Arkognosim, a source of energy found on the earth inhabited by the docile and peaceful "gnosim," a gnomic population.

These small creatures serve as a source of energy for the Dark Lord, who orders his demons to kidnap them.







A rebel opposes the evil will and throws himself headlong into the task of saving all the little inhabitants of the planet and defeating the evil despot.

The goal is therefore simple: fight the monsters, save the little "gnosim," collect bonus energy fish (extra points and lives), and reach the last level to destroy the Big Boss.

A quick task on paper, but obviously not easy. No game is easy, and of course, the enemies will try to kill us.

But what is Arkon Blaze like as a whole?

A budget product made with determination. It vaguely reminded me of Dragon Spirit in style and setting, but without all the power-ups.

Our hero moves well and is carefully animated.

The various monsters and "friendly" sprites on the screen are also carefully animated.

Perhaps I would have taken more risks with the enemy assault waves, but what's there works.

From a graphical point of view, I would have pushed a little harder on the choice of colors. Sometimes the objects in the background get mixed up and everything becomes confusing, but this doesn't make it any less playable.

The sound effects are decent.

The difficulty level is balanced, although, as I said, I would have taken a chance on adding a few more elements to the gameplay.

Power-ups or things like that. Just extra points and extra lives are a bit too little.

I would have added different weapons or a smart bomb, for example.

All in all, however, I would recommend the game for its technical effort and structure. Perhaps an 'Enhanced' version in the future will set the style and increase the final rating.

You can find it on the website I mentioned in the description comment for $3.49, which is nothing ;-).

by **Carlo Nithaiah Del Mar Pirazzini**

OUR FINAL SCORE

» Gameplay 70%
I would have dared more, but what's there is fine.

» Longevity 75%
Not many monsters on screen, but the difficulty is well structured.







BEACH HEAD

Year: 1983

Editore: Access Software

Genre: Shoot'em up

Platform: Commodore 64

With all due respect for the Palestinian situation and also for past wars and their victims, we present a game where war is fortunately only pretend (and I can tell you in confidence that it is the only fun war, because it is simulated).
What's more, the only victims could be the joystick, due to excessive use, or our social life, if we have one.

Beach Head by US company Acces Software is a game structured on five levels, which I like, i.e., different game modes. Each has a specific task that will affect the subsequent ones depending on our skill and military strategy.

Given the period in which it was released (from 1983 to 1985), it can be assumed that the setting was World War II, during the Allied landings, given the title.

Let's start with the first level, which is optional, meaning that it is not mandatory to complete it, although I highly recommend it in order to win more easily in the next one: here, our ship is involved in a plan where torpedoes and mines are visible, preventing us from reaching the opposite shore.

The second level is a fixed-screen shooter on the open sea in which we take command of an anti-aircraft turret located on a ship and have to shoot down a certain number of enemy aircraft coming towards us to prevent us from becoming a wreck.

The third level is also fixed screen, very similar to the previous one, but this time we have to shoot down enemy ships with great precision, adjusting the height of the cannon to hit the target before it hits us, otherwise we will be hit by one or more enemy torpedoes.

The fourth level is similar to the first, but this time we are driving a tank in a level that scrolls from right to left, facing various enemies and obstacles worthy of a war zone.

The fifth and final level is where we finally reach our hill. Well, try to enjoy it and find your own way to wave the white flag, otherwise everything here becomes a solution if not a spoiler. I can only tell you not to arrive with just one tank, otherwise even if you are a highly decorated retired general, you won't be able to complete it. In terms of playability, I didn't find Beach Head boring at all, even though the levels, although different from each other, have some similarities.

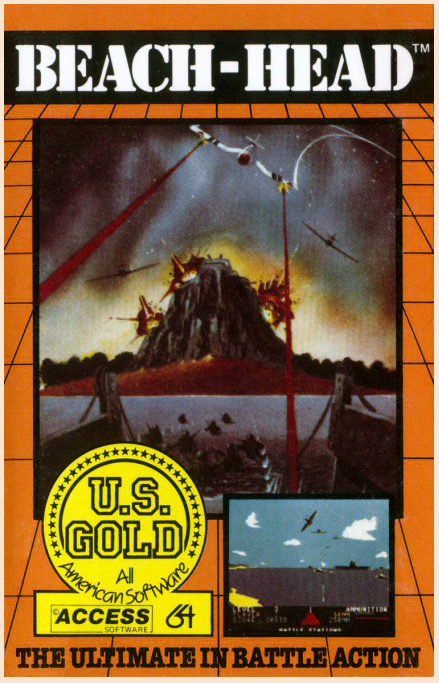
As for longevity, it took me longer than compulsory military service to reach the last level (even with ten lives), so there's plenty of fun to be had and company is guaranteed. Unfortunately, there is no sound, although this seems more than justifiable in a war game.
The game was successful enough to warrant a sequel, Beach Head 2, and as if that weren't enough, the series inspired a number of games released for PC in the 2000s.

I'll close by hoping that hostilities around the world will cease and more light-hearted, friendly, and/or video-game-like competitions will begin, like the games found in stores, on the internet, and... in this magazine!

by **Daniele Brahimi**







OUR FINAL SCORE

» Gameplay 70%
Dieci vite sembrano tante per superare cinque livelli, ma...

» Longevity 750
Abbastanza semplice, anche se richiede troppa precisione in alcuni punti.

Preserve... to experience the future

Video games are one of the most important cultural products of our time, and as gaming experiences become increasingly complex, interactive, and up-to-date year after year. So, preservation will be one of the most complex challenges of our future.

In a recent study, the Video Game History Foundation revealed a shocking truth for all of us enthusiasts: 87% of video games released before 2010 are "critically endangered". We are talking about something like 9 out of 10 games that could become inaccessible to future generations. But why might we lose this piece of life/history? The answer is more complicated than expected.

Video games, by their very nature, have a life cycle. They disappear from stores and shelves and become impossible to find. Only the most famous sagas are re-released, but thousands of titles disappear without leaving a physical trace.
The copyright law is also to blame, as it prevents libraries and cultural institutions from preserving, displaying, and sharing video games in the same way as films, books, or music.

All this, together with the inflated market driven by retro collectors and the physical disappearance of media, could result in an incalculable cultural loss. Yet the video game industry is currently worth around €180 billion, surpassing both the film and music industries! So why is this happening?

2024 could be the crucial year. In the US, a discussion on copyright laws is planned, although time is running out.
A strong action is needed, a public motion, an initiative, something that will change the current laws. Emulation and implementation need to be recognized as sources of "historical preservation". Associations need to be allowed to "dump" the titles of the glorious home computers of the past without repercussions.

This is not about piracy, it's about freedom and history.
Because of all the adventures we have had over the past 40-odd years, we risk no longer being able to press the 'fire' button and seeing the words 'GAME OVER' appear on the screen. Long live preservation!.

Carlo Nithaiah Del Mar Pirazzini

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 25 - 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