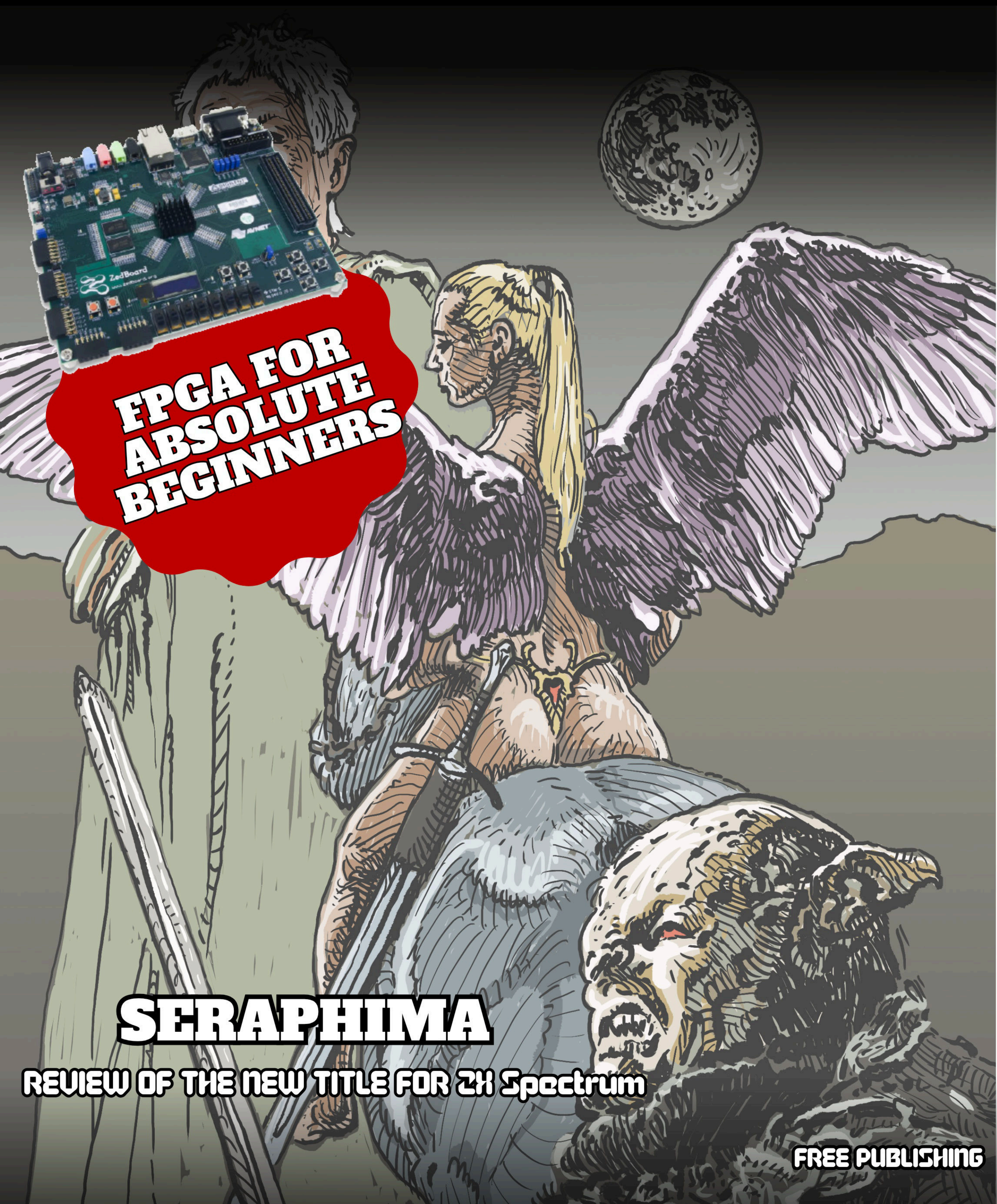
PROGRAMMING - HISTORY - INTERVIEWS - REVIEWS - INSIGHTS

Future days are back

RETROMAGAZINE WORLD

ISSUE 24 - SUMMER 2025 - WWW.RETROMAGAZINE.NET



FPGA FOR ABSOLUTE BEGINNERS

SERAPHIMA

REVIEW OF THE NEW TITLE FOR ZX Spectrum

FREE PUBLISHING

Numbers

The numbers of the retro-computing and retrogaming movement have now reached truly remarkable levels. The number of enthusiasts has been growing steadily for several years now, and with the ever-increasing audience there are many initiatives (commercial and otherwise) that are born and developed to intercept the demands and tastes of the public. There are also many editorial enterprises aimed at reviving the splendour of the magazines of the 80s and 90s, then the only real source of information and updates for all those approaching the world of home computers or gaming consoles.

In 2025, there is no shortage of tools to share information effectively and quickly, and everything would suggest that social media, blogs, forums and other forms of digital communication on the Net are the natural meeting point between those who seek the latest news on the retro-hardware and software world and those who systematically research, collect, process and publish the most interesting news and historical details on the many 8- and 16-bit platforms of the past.

The editorial initiatives that anyone approaching the world of retrocomputing (and I include those interested only in retrogaming) finds on the Net today are several and multiformed: from the blog of the individual enthusiast to the monothematic forum, from the generic Facebook group to dedicated Instagram or X profiles, from crowded Telegram channels to specialised websites, from YouTube accounts to interactive channels on Discord or Twitch. The supply is really rich, perhaps exceeding the actual demand.

Then there are the enterprises that wink nostalgically at the historical magazines of the past and somehow try to pay homage to it. RetroMagazine World is honoured to be part of this category and follows the path traced by a long tradition of spontaneous, unofficial or semi-official publications inspired by the amateur underground e-zines of the 1990s and the legendary Italian and European magazines that came out in the early 1980s to anticipate and then drive the success and spread of home and personal computers.

In the last four years, we have also witnessed the arrival of a number of printed publications (single-platform or 'generalist' publications) that are distributed by subscription or through newsstands, closely following the dynamics of the computer magazines of the past, with all that this entails in terms of costs, which are usually considerable when it comes to the printing and distribution of physical media. These editions are sometimes the result of a real passion for this world of ours, sometimes they are mere speculation aimed at riding the wave of renewed public interest in machines, consoles and video games of the past.

Distinguishing between various publishing initiatives is easy. It is equally easy to decide to support the most deserving ones, based on content, quality and visible commitment. We at RMW believe in an 'open' formula. Open to all the contributions that can come from the many true and sincere experts hiding here and there in the folds of social media or forums dedicated to the various computer and video game platforms. Open to collaboration with other groups of enthusiasts and with the editorial offices of other publications. The history of 50 issues in Italian and 24 in English testifies to active cooperation with any kind of proposal or suggestion coming from outside. We know for certain that this approach of ours works and that it manages to gather an ever-growing number of supporters and followers around RMW. The numbers say so.

David La Monaca

Table of Contents

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

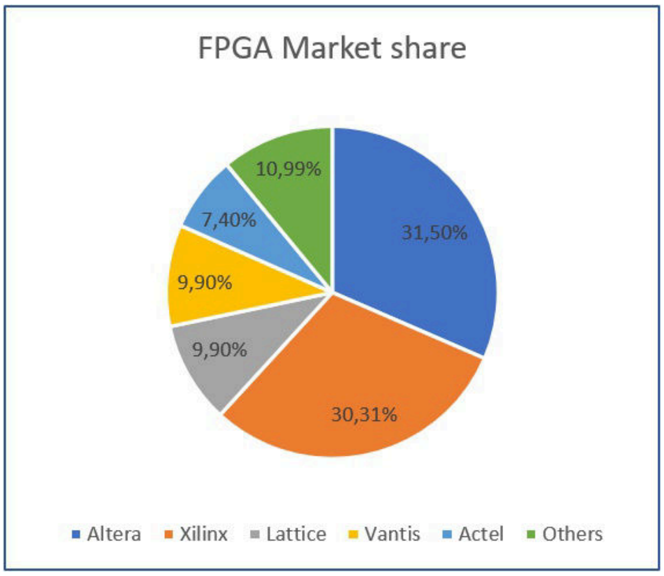
- Dave Nardella
- David La Monaca
- Carlo N. Del Mar Pirazzini
- Daniele Brahimi
- Francesco Fiorentini
- Francesco Donatini
- Roberto Del Mar Pirazzini
- Giampaolo Moraschi
- Giuseppe Rinella
- Fredrik Ramsberg
- Ingrid Poggiali
- Eugenio Rapella
- Takahiro Yoshioka
- Marta Rossmann
- Gianluca Girelli
- Marco Pistorio
- Cover Image: **Giuseppe Mangini**
- Cover Layout: **Carlo N. Del Mar Pirazzini**
- Closing Image: **Fabio de Renzis**

FPGA for absolute beginners: starting from scratch - part two

by Dave Nardella

Brands and families

The FPGA market sees two major players, Xilinx, acquired by AMD, and Altera, acquired by Intel, who equally divide about 60% of the total turnover. The remaining 40% includes Lattice, Vantis, Microchip, Achronix, Gowin, etc.



If you decide to devote yourself to FPGAs, the first non-trivial applications will force you to choose your 'religion', which will almost certainly be Xilinx or Altera.

Both have a very rich catalogue, from entry-level FPGAs up to military-grade chips.

Each proposes four or five families characterised by a 'trend' of use, within these families there are various FPGA models that differ in their equipment: number of CLBs, amount of RAM, number of DSP blocks, etc.

Their development tools are complete and constantly updated, and both provide a free lite version for programming small to medium-sized FPGAs. This is not a big limitation, consider for example that with Intel's Quartus II Lite you can run the Cyclone V family, which I assure you is very powerful.

Again, you will find avid supporters of one or the other brand. Personally, I do not see any great disparity in the catalogue or prices that would make one or the other prefer, tout-court.

The old Intel/AMD war is replayed, but in this case they are fighting on equal terms.

Recommended approach

FPGAs sound interesting, what do I need to get started? If I then find that they do not meet my interests, will I have wasted a lot of money?

The use of FPGAs is aimed at the professional market, generally development systems tend to cost more than what a hobbyist/maker is used to spending, but fortunately this is not always the case, there are entry-level systems on the market to suit all budgets, you just have to know how to choose.

But even before getting a development system, it is important to understand the right approach to these systems. When we buy an Arduino, we plug in our I2C display, download a couple of libraries and after a few minutes and a few lines of code we see our 'Hello World'.

With an FPGA it's different, we don't have I2C, SPI or video interfaces, we have to create them ourselves by instantiating IPs; at the beginning of the journey, the most exciting result will be to make an LED flash.

I say this, not to discourage you, but to point out that it is possible to achieve everything, but you have to proceed in small steps.

My tips for getting off on the right foot are:

1. Buy a very inexpensive FPGA board.
2. Avoid demos and start with a good Verilog/VHDL book, simple but complete.

Remember that everything you will achieve using a book, which I assure you is no small thing, you will be able to port to FPGAs of any brand.

Once you have acquired the basic knowledge, you will also be able to evaluate the next step, which is which board you need the most; at the beginning, it is easy to buy options that you don't need right away or won't need at all.

The problem with official demos is that they are always dated; once they are loaded into new versions of the development systems, a lot of warnings are issued, some of them invalidating, and it is difficult to handle them without the necessary knowledge.

Finally, if you unfortunately find that FPGAs just don't interest you, you will have invested less than the cost of a dinner.

Before we start with the buying advice, let's quickly look at the differences between the various board families.

Basically, the FPGA systems you find on the market fall into three categories, the division is not clear-cut, it can sometimes be difficult to categorise some boards; these are:

1. Development systems
2. Educational systems
3. Breakout Boards

The former are used to test the functionality of a family, contain dynamic RAM, integrated peripherals such as Ethernet, analogue/digital circuitry, and a large number of I/O connectors. These systems are dedicated to professional/evolved use, and start from a few hundred euros up to as much as 70,000 euros. The concept of use is as follows: I have to develop a board and have selected the most suitable family. I use a development system that contains the most powerful chip of that family to develop my programme. When everything works, I see how many resources my programme occupies; then for my production I select the most suitable model in the family: capable, but not too much.

Educational systems have on board buttons, LEDs, switches, 7-segment displays, buzzers and whatever else is needed to test the first programmes with everything already connected. Here the costs are lower, but remain in the 100-300€ range depending on the FPGA model they mount.

Finally, there are the Breakout Boards, which are low-cost educational systems that have no or very few display buttons and LEDs; they basically consist of an FPGA and little more, to which the pins are connected directly to the interface connectors.

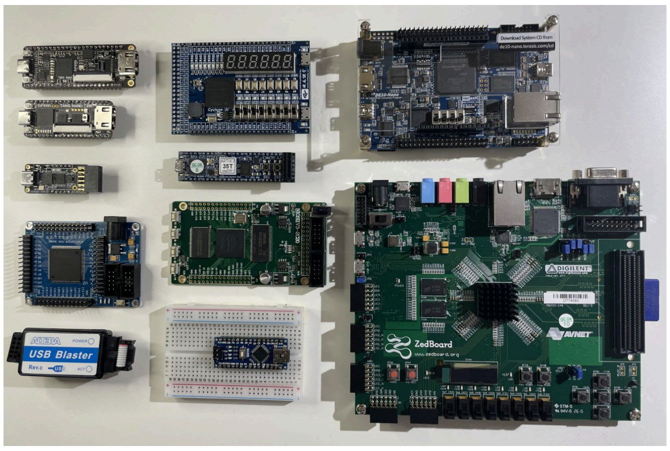
The advantage is that they can generally be plugged into a BreadBoard. The principle of use is similar to that of the Arduino Nano, ESP32 modules, etc. With a few euros and a little manual dexterity, we can connect buttons and LEDs at will. For entry-level educational use, these boards are perfect, remember that the FPGA already contains everything we need.

Purchase advice

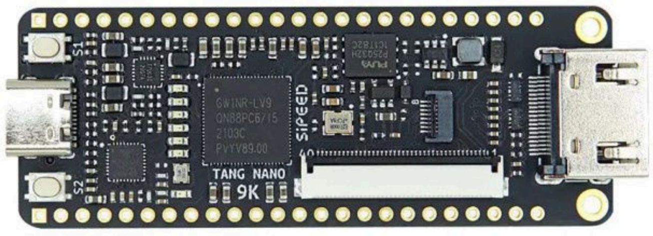
I will briefly list the models that I have at home, in order of price starting from the lowest (which can obviously vary depending on the supplier), and a few other boards that I do not own but know quite well, also pointing out in which learning step it is best to use them. I will only dwell more on the second model, which is the one I recommend to get started.

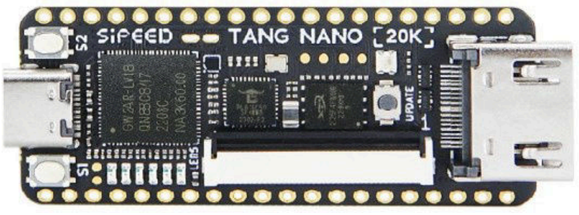
In the picture I have also included an Arduino Nano on a mini-Breadboard to give you an idea of the size.

Sipeed Tang Nano 9k and Tang Nano 20k



These are two boards produced by Sipeed, the latter is the evolution of the former, the difference is only a few euros but it offers much more.





This is the model I recommend for a number of reasons:

1. It mounts a rather interesting chip, the Gowin GW2AR-18 (20,737 CLB, RAM, DSP, PLL and Flash) which, in addition to the first experiments, allows you to host a RISC-V softcore and run a micro-distribution of Linux. There are a few retrogame projects realised for this small board.

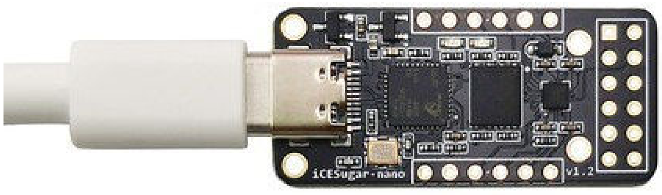
2. It has interesting hardware: many I/O pins, a dedicated connector for LCD screens, an HDMI connector, the socket for a micro-SD, a USB interface used for both programming and RS232 communication, a speaker connector, six LEDs and two user buttons.

3. The Gowin development system is very light and simple to use, and does not contain many advanced options, which could be misleading at first. This board can be programmed with the lite version.

4. It is possible to run, directly or with minor modifications, all the examples present for the Tang family, of which there are many.
You can buy it from the official Sipeed shop on Aliexpress, for around 34€ (Tang Nano costs around 19€) but it is available in many other online stores.
Here you will find the official wiki (of the entire Tang series) and links to tutorials and experiments.

ICESugar-nano

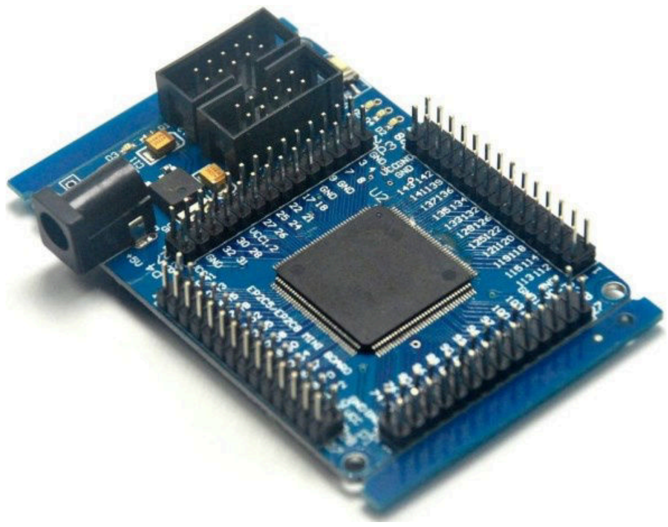
It is one of the smaller boards, mounts a Lattice iCE40LP1K-CM36 and is manufactured by MuseLab (activate browser translator). It costs very little, about 22€.



The particularity of this board is that it is completely open-source, and so is the development system, IceStorm, which allows the entire iCE40 family to be programmed. It is not very powerful, but it can host micro-softcores.
It can also be programmed with the iCEcube2 Lite version of Lattice.
You can find the wiki here: https://github.com/wuxx/icesugar-nano

Cyclone II EP2C5 mini Dev Board

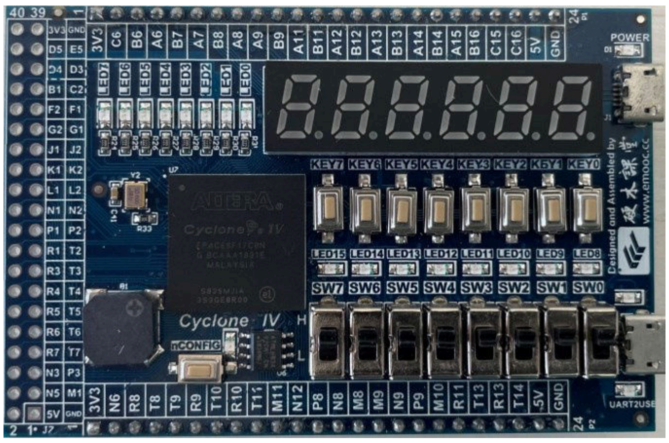
The birth of this board is shrouded in mystery. It has no official manufacturer, but can be found for a few euros in almost every online store.



It mounts a fairly old Cyclone II EP2C5T144C8 chip from Altera, and to program it you need to download an old Quartus II version, no later than V13 (it can still be found on the Intel site).
The advantage is that it is an extremely popular board and there is endless documentation on the net. It is not possible to programme it directly, you need to use a USB-Blaster adapter, which you can usually find bundled with the board or separately for very little money.
This is a good starting point:
https://land-boards.com/blwiki/index.php?title=Cyclone_II_EP2C5_Mini_Dev_Board.
You can also find many examples on Github:
https://github.com/search?q=EP2C5&type=repositories

Mini FPGA Cyclone IV Board

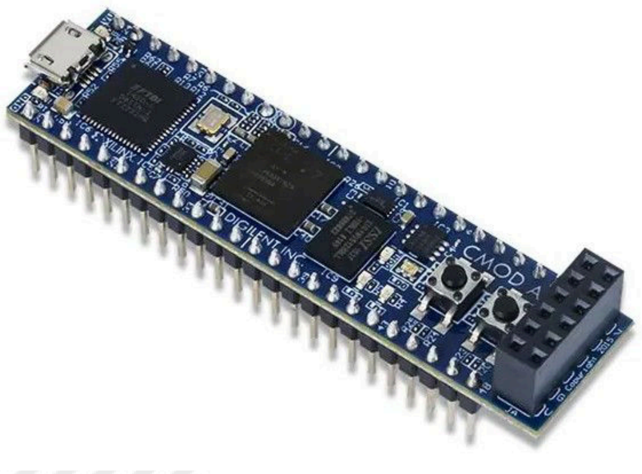
This board is very interesting, it costs very little (about 37€) and mounts an Altera Cyclone IV EP4CE6, it has eight buttons, eight LEDs, eight switches 6 seven-segment displays, a buzzer and two micro-USBs (it does not need the external programmer). It can be programmed with the latest version of Quartus II Lite.



I recommend it as a preferred purchase only if you have a bit of an adventurous spirit.
The purpose of this board is essentially didactic, on the manufacturer's wiki there are plenty of examples of well-calibrated increasing difficulty.
The problem is that the site is only in Chinese, but I assure you that the browser translator does a great job. The only problem is the writing in the pictures, but there are not many.
Unfortunately, if you want the examples, which are freely downloadable, to be used directly in the board, you have to interface with baidu, which is not accessible (as a registration) from all countries due to a trivial html page bug problem.
Bear in mind, however, that when working with FPGAs, the only absolutely indispensable document is the circuit diagram to obtain the numbering of the pins connected to buttons and displays, in which case it can be requested

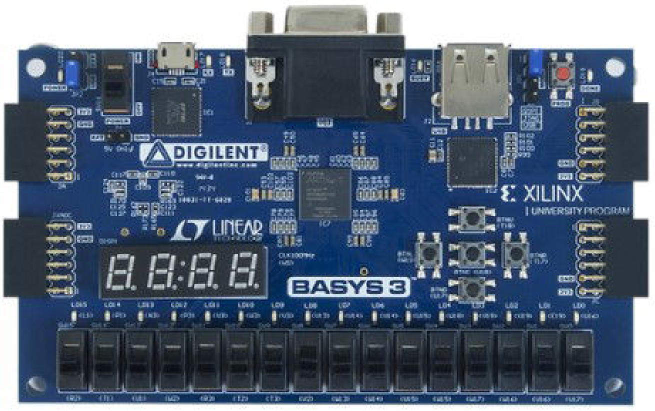
from the supplier and is sent by email immediately, the rest is not essential.
I found it a lot of fun, it is compact and works very well.
You can find the experiments here:
https://www.yuque.com/yingmuketang/01/ri3fkv
This is the manufacturer:
http://www.emooc.cc/
You can find the board on Aliexpress.
I would only recommend the boards from this point onwards as a purchase at a later stage of preparation because the cost increases; they are not exaggerated figures, but my aim is to get you to spend only the amount needed to get started, learn the basics and a little bit more. However, the final choice is yours, they are certainly very good products.

Digilent Cmod A7-35T
This is a breakout board but it mounts a Xilinx Artix-7 XC7A35T, a fairly high-performance mid-range board that has 33,280 CLB, 1,800Kb of RAM blocks, 90 DSPs and 5 Transceivers at 6.6 Gb/s. This board has 512MB of SRAM and allows MicroBlaze with performances of 303 DMIPs or other softcores to be hosted without problems. The interesting advantage, given its very small size, is that it is possible to use this board directly in one's own final realisations by mounting it on a host board that could contain the auxiliary interface circuitry.



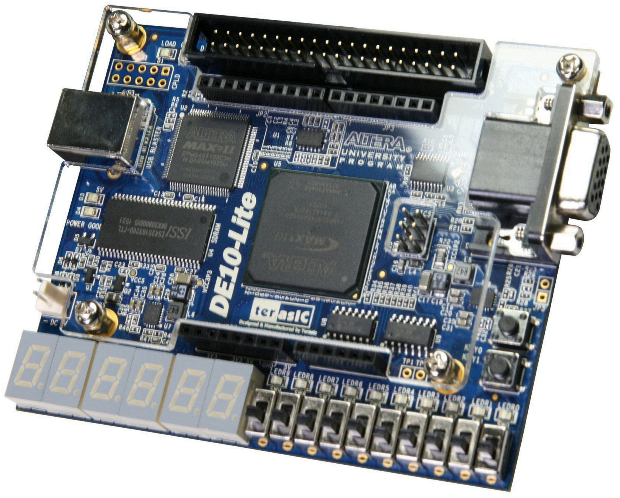
The Artix 7 family is more energy-intensive than its cousin Spartan 7; therefore, I sincerely expected it to heat up more, in reality the temperature, with 70% of the committed capacity (beyond that it is not advisable to push), remains at acceptable levels. For critical applications, however, I recommend using a mini-dispenser.
Its cost is around 111€.

Digilent Basys 3
It has the same chip as the Cmod A7 35-T but it is an educational-oriented board, in fact it has 16 switches, 5 buttons, 4 7-segment displays and a 12-bit VGA output as you can see from the picture.



It is a historical and very popular product, there are entire sites dedicated to it and many examples. You can find it for about 190€.
This is its official page: https://digilent.com/reference/programmable-logic/basys-3/start

Terasic DE-10 Lite
Here we are in the Intel-Altera world. This board mounts a MAX 10 10M50DAF484C7G containing 50,000 CLBs, 1,638 Mbit RAM, 144 18-bit DSPs and 4 PLLs.

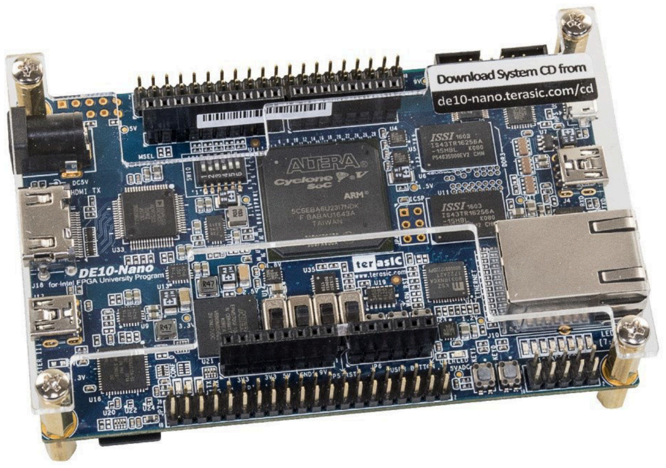


The FPGA is mid-range, practically a stripped-down version of the Cyclone family, but the board has a wealth of switches, displays and interfaces at an honest price of around €162.
It has Arduino-compatible expansion connectors, it's not the only board to have them, but beware, all FPGAs work at 3.3V max, so shields designed for 5V can damage the board. In that case, go for shields compatible with Arduino 2 or Arduino GIGA R1 WIFI, or alternatively, you can use all the interfaces designed for the ESP32 series or for

ARMs (STM32, Renesas, etc.).
It supports the softcore Altera Nios II, two instances for sure. You can find it here: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=1021

Terasic DE-10 Nano

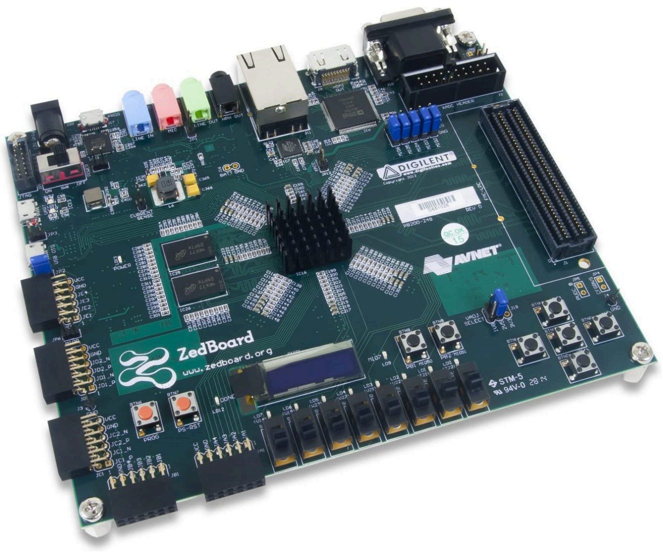
This is a beautiful board equipped with a Cyclone V SE 5CSEBA6U23I7, it is an SoC containing an ARM Cortex-A9 Dual core at 800MHz as HPS.
At this link: https://www.terasic.com.tw/cgi-bin/page/archive.pl?
Language=English&CategoryNo=167&No=1046&PartNo=1#contents
you can delve deeper into the specifications and the range of interfaces, which are really rich. The only note, as on all boards containing SoCs, the peripherals are split between HPS and FPGA, e.g. the Ethernet interface is connected to HPS and invisible from FPGA.



It only has four switches, so it cannot be considered an educational board, on the other hand there are many examples around.
Beware though, it is not a very simple board, the fact that it has an FPGA SoC complicates the first approach quite a bit; you need to study the manual very well. Using it without the HPS (the ARM cores) is possible but does not make much sense unless you at least plan to do so in the future. Programming HPS itself is not complicated, but connecting FPGAs with HPS can cause some headaches if you don't proceed in an orderly manner.
It is a very popular board, also thanks to the MisTer retrogaming project, for which, however, some additional boards are required.
The considerations for the Arduino connectors are the same as for the DE-10 Lite.
The only flaw is the plexiglass panel, which is unnecessary but above all harmful because it restricts air circulation. The FPGA also gets very hot 'at idle'. In my opinion, a low-profile elliptical copper heatsink should be installed on the chip, and in this case, the plexiglass panel, suitably drilled, can serve as a support for a small fan. You can find the DE10-Nano for around 260€.

Digilent ZedBoard

This is also a historical board, the benchmark for learning to work with the Xilinx ZynQ family. It mounts an XC7Z020 SoC and has an impressive array of peripherals.
It is advertised as a low-cost development board even though it costs around €680, but considering the FPGA installed and the equipment, it is still an appropriate price for a semi-professional tool.
You can find the official page here: https://digilent.com/shop/zedboard-zynq-7000-arm-fpga-soc-development-board/.



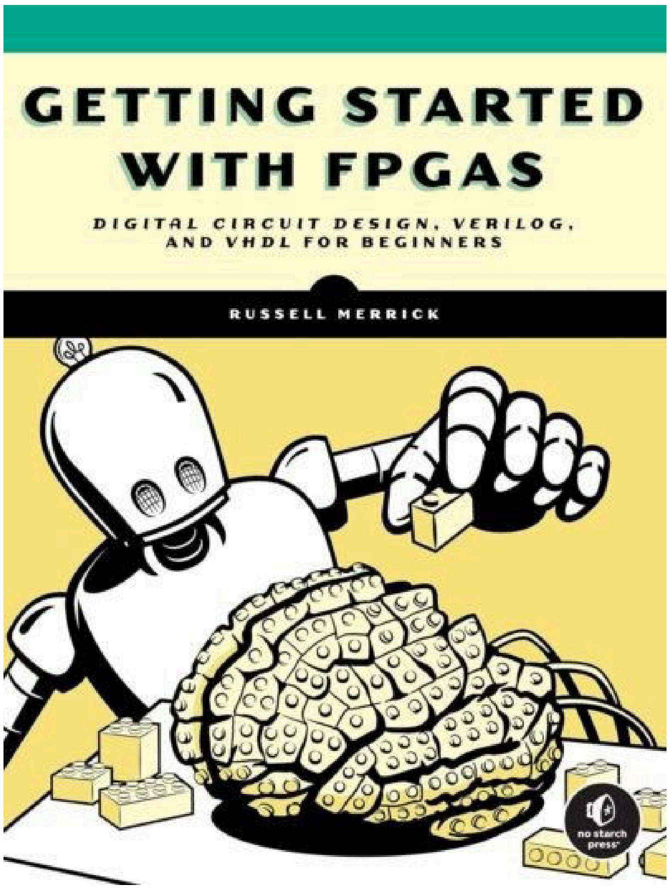
I don't recommend it as a first purchase, it is quite complicated to use if you are not experienced with the Xilinx world and the use of advanced IP.
The HPS is an ARM Dual-core A9 at 667Mhz, it makes no sense to use this board for small experiments, by default it is installed Petalinux, a very modular distro derived from Yocto, unfortunately it is not very easy to use; to modify/install it you need to use the Xilinx SDK and the BSP (Board Support Packages).

The first approach with this board was a bit tricky, fortunately there is a lot of material on the net.
If you happen to find it second-hand, it is important that you transfer the Xilinx licence attached to the board in order to be able to use the development system with this processor family, with the lite version you cannot do this.

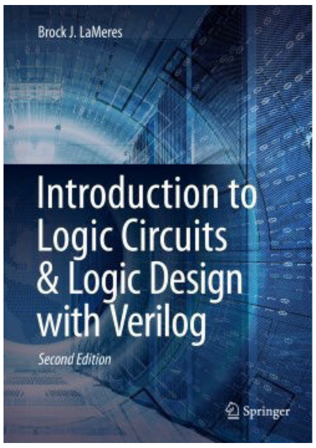
Recommended resources

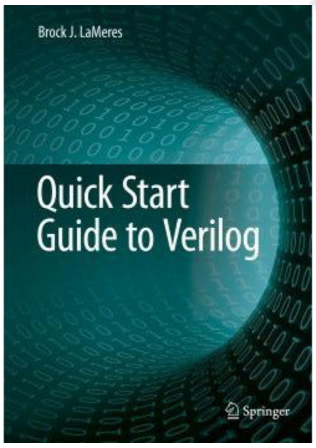
Books

I always prefer 'agnostic' books, i.e. not too tied to a brand, the specifics of its development tools and its special IPs. I recommend a few texts that I have read, some completely, others focusing only on certain topics.
The simplest and most down-to-earth book I have found is without a doubt:
Getting Started with FPGAs by Russell Merrick.


GETTING STARTED WITH FPGAS
DIGITAL CIRCUIT DESIGN, VERILOG, AND VHDL FOR BEGINNERS
RUSSELL MERRICK
no starch press

The experiments are conducted on a small, low-cost board containing a Lattice FPGA, but are extremely portable anywhere, covering both Verilog and VHDL.
It reads really well and the more complex concepts are always explored in depth with examples. If you have a limited budget (unfortunately books cost money) and can only buy one, this is the one I recommend.
You can find all the references on the NAND LAND website, which I invite you to visit because it contains many tutorials and useful information.
If you also want to learn more about digital electronics and Boolean logic, you can buy:
Introduction to Logic Circuits & Logic Design with Verilog by Brock J. LaMeres.
It is very comprehensive and easy to follow, and this author writes very well.
By the same author you can buy
Quick Start Guide to Verilog, which is more focused on





Verilog.
Finally, if you really want to learn about digital electronics, microprocessors and their implementation in VHDL, I recommend two books that you can find in both Italian and English.
Introduction to Digital Systems Design
Introduction to Microprocessor -based Systems Design
By Guido Donzellini and others.
(See figure on page 9).

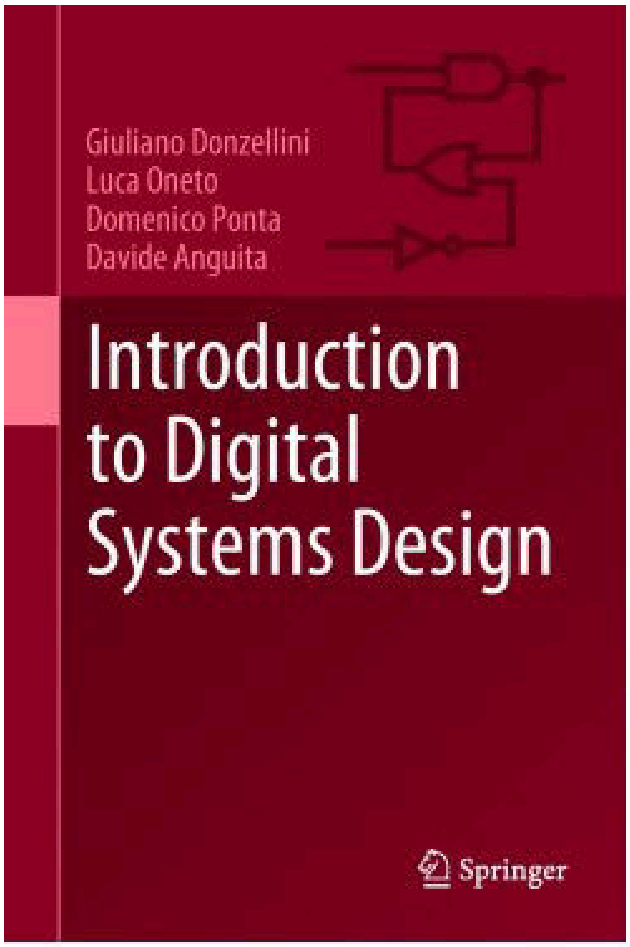
The experiments are conducted with deeds, a freeware editor/simulator written by the author himself, which is very simple to use but complete and allows the circuits created to be exported to VHDL.
This is the site where you can find links to the software, examples and books in two languages:
https://www.digitalelectronicsdeeds.com/index.html

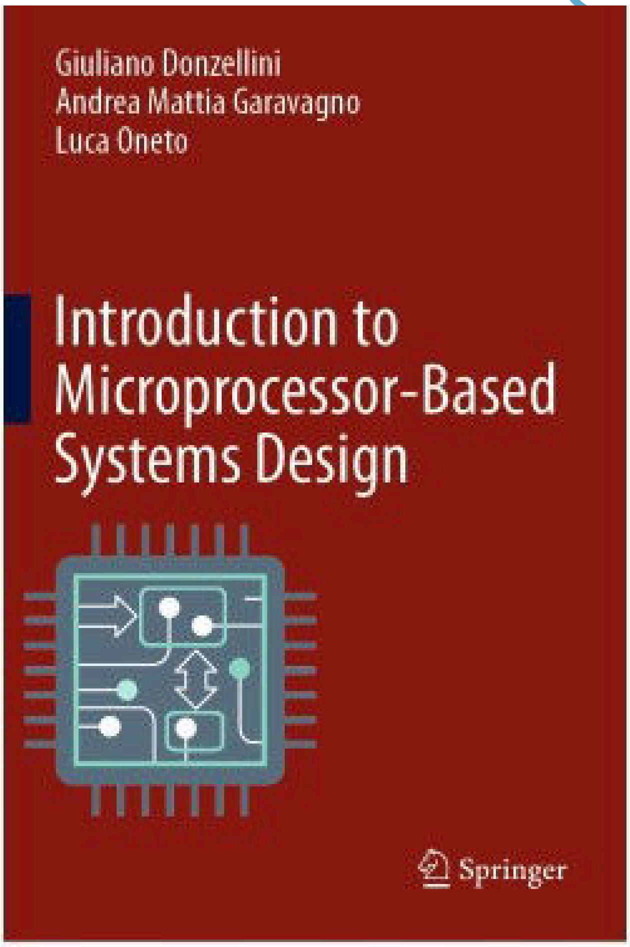
I should warn you, however, that these are university texts; I found them very complete and not very complicated, but they do require a certain amount of commitment. The possibility of visually simulating the circuits studied is certainly a considerable plus.

Tutorials

If you prefer to start 'soft' with a few tutorials, before buying one or more books, there are an infinite number of resources on the net; probably too many, and this, given that they are free, could give you indigestion.
I recommend just two links, start with these and follow them completely, eventually you will be able to select the resources best suited to your needs.
FPGA Tutorial: https://fpgatutorial.com/
NAND LAND: https://nandland.com/

Finally, on the OpenCores site you can find a plethora of example IPs and softcores: https://opencores.org/





Giuliano Donzellini · Luca Oneto · Domenico Ponta · Davide Anguita
Introduzione al Progetto di Sistemi Digitali
Seconda Edizione
Springer

Giuliano Donzellini · Andrea Mattia Garavagno · Luca Oneto
Introduzione al Progetto di Sistemi a Microprocessore
Springer

Conclusions

We have come to the end of this talk, and I hope I have stimulated your interest in these systems, which in my opinion are quite long-lived, in the sense that you will hardly get tired of them because no matter how much you dig, there is always another level of complexity to tackle.

Fortran on the C-64 - integer or float

by Francesco Fiorentini

In the last issue we saw how to use the Abacus Fortran-64 to write, compile, link and run a Fortran language program on our beloved Commodore 64. Since appetite comes with eating, I wanted to continue my experiments with this interesting language, especially to see if the C64 compiler could be used for professional or even semi-professional use.
Since Fortran code is compiled, I immediately thought that it was much more powerful than the "mistreated" Basic V2... But from what we will see, there will be no shortage of surprises.

What sample program to use?
My experience with Fortran is very little, so I had to find a way to write a simple program that could be easily replicated in Basic and Fortran, but at the same time put the calculation capabilities of the two languages to the test. My idea: why not writing a program that would add up all the numbers from 1 to 10000? Simple, effective and, I thought, error-free.

The version in Basic V2 is quite trivial, I only quote it for the sake of the record:

```
10 a=0: f=10000
20 for i = 1 to f
30 a=a+i
40 next i
50 print "the sum of first "; f ; " numbers: ", a
```

The Fortran version is slightly more complicated, but not too much. There are just a few tricks to be taken as we must be careful to use float and integer numbers in the right place:

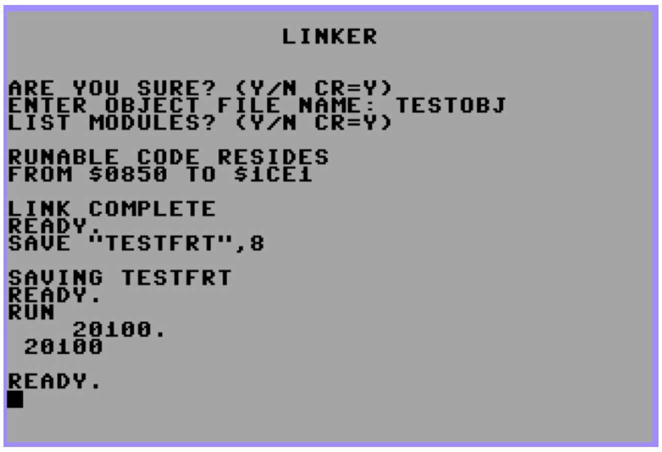
```
10 : program test
15 :* this program sum
16 :* the first 10000 numbers
20 : real f,a
25 : f=0.0
30 : open 3,3
40 : do 10 i = 1, 10000
45 : a=float(i)
50 : f=f+a
60 : 10 continue
65 : write (3,100) f
70 : close 3
75 : stop
80 : 100 format(f10.0)
90 : end
```

Note: To edit, compile and link the code above, please refer to the article in the previous issue.

At this point all I had to do was run the two programs and... I couldn't believe my eyes! The Basic code returned the result when Fortran still seemed to be stuck. I thought to myself: I must have made some mistake in the code, the conversion is probably uncorrect. I couldn't possibly think that Fortran was so slow. You can surely imagine my astonishment when, after a long time of waiting, the C64 on which Fortran was running returned its result: 49969900. Boy was it slow!
But, wait a minute. The result returned by Basic was 50005000, while Fortran's was 49969900. Who was telling the truth?
Since the sum of the integers from 1 to I is I*(I+1)/2, surely the correct value was that of Basic.
But why had Fortran returned such an incorrect value?
From a quick consultation with Eugenio Rapella from the editorial staff, we assumed that both the slowness and the calculation error could be caused by the conversion from INTEGER to FLOAT.
So how do we find the error (at this point, the speed factor was secondary)?

Adding up integers and floats
I did a test summing floats and integers up to 200 but, as we can see from the result (Fig. 1), there are no discrepancies:

```
10 : program test
15 :* this program sum
```



```
                 LINKER

ARE YOU SURE? (Y/N CR=Y)
ENTER OBJECT FILE NAME: TESTOBJ
LIST MODULES? (Y/N CR=Y)

RUNABLE CODE RESIDES
FROM $0850 TO $1CE1

LINK COMPLETE
READY.
SAVE "TESTFRT",8

SAVING TESTFRT
READY.
RUN
    20100.
 20100

READY.
█
```

Fig. 1 - Sum using only integers

```
16 :* the first 200 numbers
20 : real sf,a
22 : integer si
25 : sf=0.0
26 : si=0
30 : open 3,3
40 : do 10 i = 1, 200
42 : si=si+i
45 : a=float(i)
50 : sf=sf+a
60 : 10 continue
65 : write (3,100) sf
66 : write (3,200) yes
70 : close 3
75 : stop
80 : 100 format(f10.0)
85 : 200 format(i6)
90 : end
```

How then to find out where the error spawns? The consultation with Eugenio was not limited to suppositions, he also suggested a practical way to find the answer to this question: since the sum of the integers from 1 to I is I*(I+1)/2, you could insert a 'check' as the programme runs it to see when and why 'discrepancies' occur.

```
10 :program test
15 :* this program sum
16 :* the first 10000 numbers
17 :* with error control
20 : real f,a,c
25 : f=0.0
30 : open 3,3
40 : do 10 i = 1, 10000
45 : a=float(i)
50 : f=f+a
51 : c=a*(a+1.0)/2.0
52 : if (f .ne. c) then
53 : write (3,*) 'error:', f, c
55 : write (3,100) a
59 : endif
60 : 10 continue
65 : write (3,100) f
70 : close 3
75 : stop
80 : 100 format(f10.0)
90 : end
```

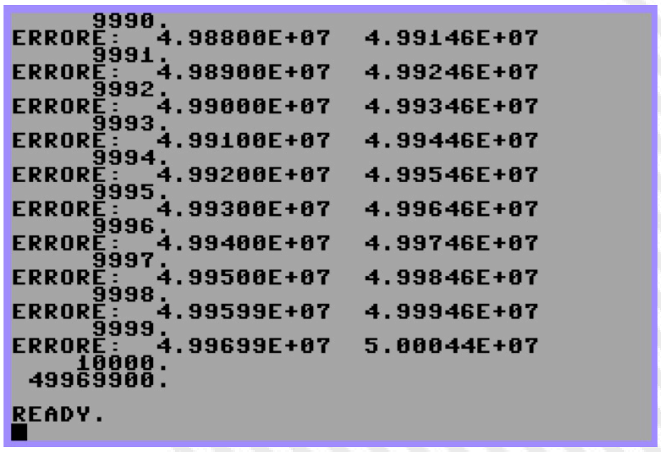
In this program, we are going to test the sum of the floating-point numbers with the calculation a*(a+1.0)/2.0 always obtained with floating-point numbers. As you can see (Fig. 2), the difference is immediate and the number returned from the sum is exactly the one shown at the end of the cycle... So our assumption regarding the conversion from integer to float was correct.

Conclusions

What can I say, I never expected these difficulties with the Commodore 64's Fortran compiler. For the moment, given the chronic lack of time I will stop here, but in the future I want to find a way to perform this calculation with Fortran properly.

I already have some ideas...

How about you? Do you have any suggestions?



```
      9990.
ERRORE:   4.98800E+07   4.99146E+07
      9991.
ERRORE:   4.98900E+07   4.99246E+07
      9992.
ERRORE:   4.99000E+07   4.99346E+07
      9993.
ERRORE:   4.99100E+07   4.99446E+07
      9994.
ERRORE:   4.99200E+07   4.99546E+07
      9995.
ERRORE:   4.99300E+07   4.99646E+07
      9996.
ERRORE:   4.99400E+07   4.99746E+07
      9997.
ERRORE:   4.99500E+07   4.99846E+07
      9998.
ERRORE:   4.99599E+07   4.99946E+07
      9999.
ERRORE:   4.99699E+07   5.00044E+07
     10000.
 49969900.

READY.
█
```

Fig. 2 - Output from the control program

Power C - the best C compiler for C64/128?

by David La Monaca

Personal notes

I have to be honest: in the heyday of the popularity of the Commodore 64, the second half of the 80s, I had never heard of Power C, nor (mea culpa) of other compilers for the C language for what was my first computer. A combination of reasons lay at the root of my lack of knowledge: I wrongly considered the C64 too underpowered for a C language compiler that I associated instead, again wrongly, with more advanced machines like the Amiga or PC 386. Those years were for me the last of high school and the first of university and all I knew from direct experience was BASIC and assembly programming for the C64. When I switched to Amiga, C seemed very complex for me to program to create something useful under the Workbench, so my first real experience of coding in C was on a 386 PC with Borland's infamous Turbo C. In the first two years of Electrical Engineering, the only courses that involved the use of a programming language were Fortran (Numerical Analysis) and Pascal (Electronic Computers). Not even a shadow of C. In order to practise in Pascal, I had obtained the excellent Oxford Pascal for C64 and I sometimes used C1 Pascal on the Sinclair QL of a good friend of mine who was in the Maths faculty. When I began to explore C in earnest, the DOS PC with VGA card and Turbo C 1.0, with its convenient integrated environment, greatly simplified the process of writing, editing, compiling and linking the small programmes I was trying my hand at.

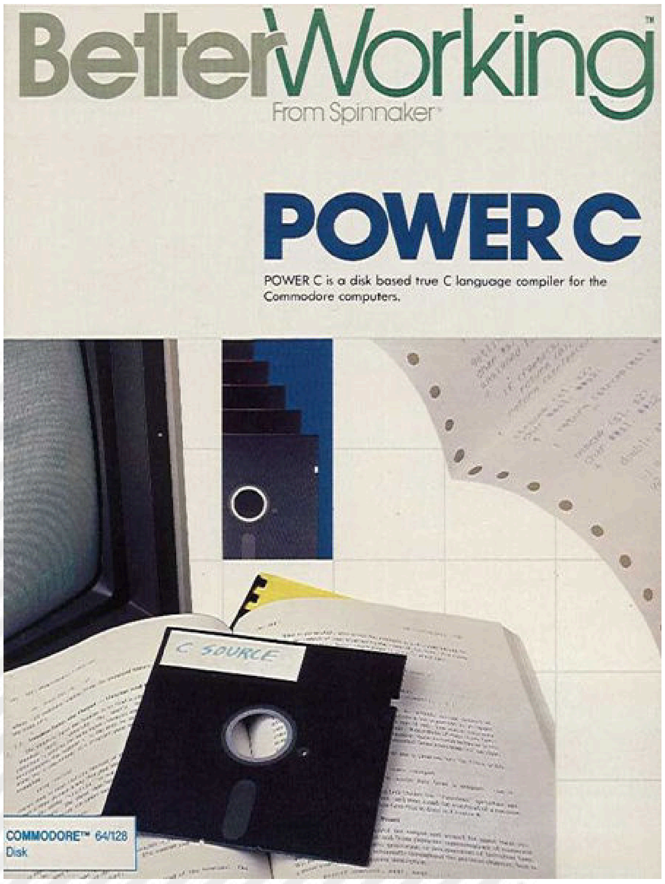
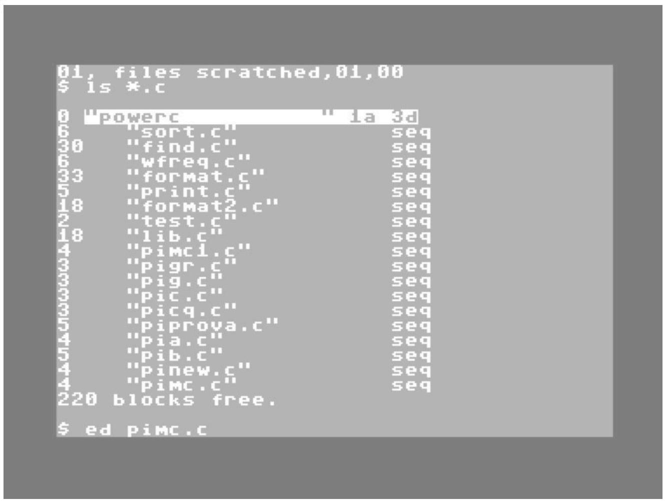

BetterWorking™
From Spinnaker
POWER C
POWER C is a disk based true C language compiler for the Commodore computers.
C SOURCE
COMMODORE™ 64/128
Disk

Fig. 1 - Original cover of the Power C package

Today, after discovering Power C for Commodore 64, actually published back in 1986 by Spinnaker Software Corporation, I have to admit that if I had known it at the time, I would hardly have given it up in favour of Turbo C or at least I would have done so with greater effort and sense of nostalgia. Yes, because Power C, in my opinion, truly represents a small shining star in the galaxy of existing Commodore 64 software. A formidable example of a complete programming tool with a thousand possibilities. A basic Commodore 64/128 equipped with a drive and the basic Power C package diskette are all you need to start programming in C and, I would add, get off to a great start! The development system proposed by Power C is simple, well-designed and optimally made available to the user. It is a DOS-based system, i.e. all the components of the development chain are disk-resident programmes that from time to time are called upon to act on our source code, object code and function libraries necessary for our project, in order to obtain a finished programme that can also run independently of the Power C shell. And it is precisely the shell that is the innovative and central element of this compiler, which provides a prompt very similar to that which can be found on a Unix system.

The Software Package

The manual that accompanies Spinnaker's software shows in plain sight its main feature, namely that it is a complete C language compiler for C64/128 entirely based on disk.



```
01, files scratched,01,00
$ ls *.c

0 "powerc             " 1a 3d
6     "sort.c"              seq
30    "find.c"              seq
6     "wfreq.c"             seq
33    "format.c"            seq
5     "print.c"             seq
18    "format2.c"           seq
2     "test.c"              seq
18    "lib.c"               seq
4     "pimc1.c"             seq
3     "pigr.c"              seq
3     "pig.c"               seq
3     "pic.c"               seq
3     "picq.c"              seq
5     "piprova.c"           seq
4     "pia.c"               seq
5     "pib.c"               seq
4     "pinew.c"             seq
4     "pimc.c"              seq
220 blocks free.

$ ed pimc.c
```

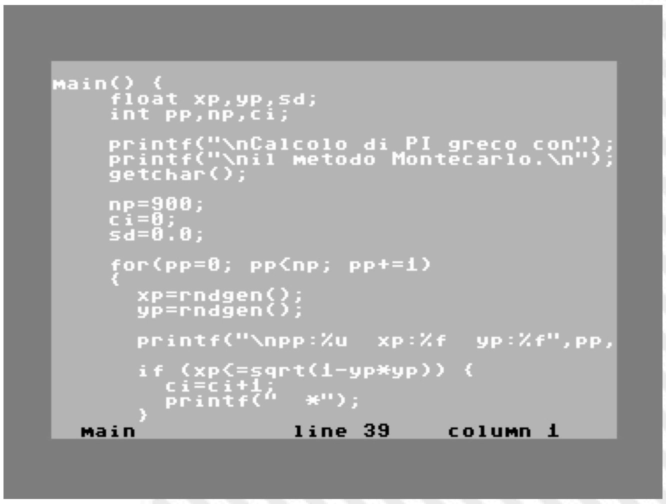
Fig. 2 - Screenshot of the C Shell

In addition to the Commodore 64, the Commodore 128 is also supported, whose larger memory and hardware features it exploits. On the most popular Commodore 8-bit platforms, Power C promises to generate programmes ten times faster than BASIC, producing compact and reusable native machine language object code. The components of the compiler (shell, full-screen editor, command-line compiler, flexible linker, complete library of 95 functions for a wide variety of application fields, additional general-purpose C libraries for sorting, searching and formatting text) combine perfectly to offer the programmer an efficient and streamlined environment as well as a profitable development experience.

The standard C, that is the Kernighan and Ritchie's ANSI C, is fully supported, and for those who wish to exploit the processor, graphics and audio potential of Commodore computers, the support libraries, even those provided by enthusiastic users over time, offer a whole series of specific functions to easily interact with the C64's custom chips. The basic package, as already mentioned, consists of a single 5.25" double-sided diskette. The first side called System Disk includes the C Shell, Editor, Compiler, Translator, Linker, the standard StdIO and Math libraries and 5 examples of C source code listings. The other side of the floppy disk is the Library Disk and contains the Stdlib.l and Syslib.l function libraries, which are useful when compiling and linking projects. The manual also includes a handy tutorial to speed up learning to use the various tools offered by the compiler.

Taking action (nowadays)

If we were still in the 80's, to proceed with using the compiler we would have made a backup copy of the original floppy disk and started with the classic LOAD "*",8,1 command followed by the RETURN key to start the Power C shell and begin our first project. In case we had a 1571 drive, which was rather rare for C64 owners, less so for C128 users (and of course C128D users who had it built in), the floppy would be read from both sides, while for all other drives (1541, 1541-II, 1570, etc.) the compilation phase would involve a few disk changes (or, better yet, side changes) to complete the procedure and get the program executable in Shell or loadable from BASIC. Since it was recommended to use a blank disk formatted as a working disk in which to store the C file sources, object codes and executable files, the number of disk changes would have soared. Of course, the lucky owners of a 1581 drive would have had an easier time: they would have only had to pour the contents of the two sides of the supplied floppy disk onto a single 800 KB 3.5" floppy disk. And they would still have had plenty of space for many projects. This is exactly what we did in order to test Power C both on a PC emulator and on real hardware, thanks to the large amount of modern peripherals available on the C64 retrocomputing market. In particular, we used VICE 3.7 on a Windows 10 PC, configuring it with a single 1581 drive and an Action Replay cartridge to speed up loading from disk. The virtual 800 KB floppy disk was created with DirMaster (a very useful tool for managing D64, D71, D81 image floppy disks, etc.). On the D81 floppy disk we then transferred the contents of the two original



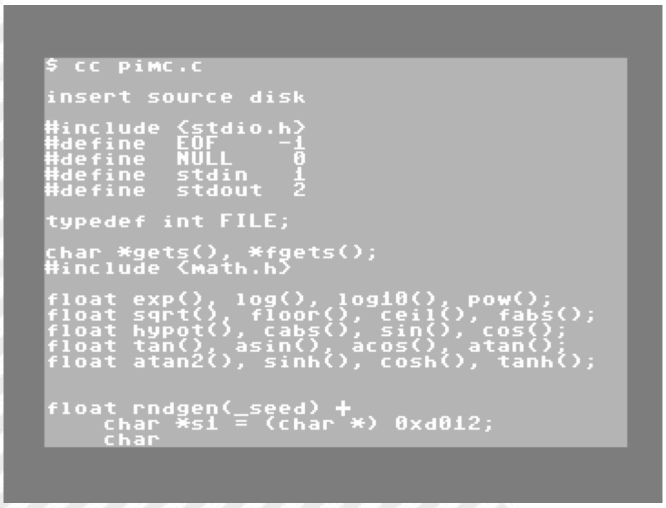
```
main() {
    float xp,yp,sd;
    int pp,np,ci;

    printf("\nCalcolo di PI greco con");
    printf("\nil metodo Montecarlo.\n");
    getchar();

    np=900;
    ci=0;
    sd=0.0;

    for(pp=0; pp<np; pp+=1)
    {
      xp=rndgen();
      yp=rndgen();

      printf("\npp:%u  xp:%f  yp:%f",pp,

      if (xp<=sqrt(1-yp*yp)) {
        ci=ci+1;
        printf("  *");
      }
  main                line 39    column 1
```

Fig. 3 - Editor at work

D64 from Power C. For testing on real hardware the same .D81 file was copied to a USB stick, which was then connected to an Ultimate II+ connected to an original Commodore 64 PAL. The Ultimate II+ interface was also configured with a version of the Action Replay cartridge (CRT file) to provide a definite boost to the transfer speed on the C64's serial port.

The usual example program? No, not this time!
We could have used the usual 'hello_world.c' or the included 'test.c' file to test the development chain of a C program using Power C, but frankly, I hope you agree with me, we can't take it any more! So we have chosen a short program for calculating Pi according to the Monte Carlo method, which, if you are familiar with RMW's programming pages, will not sound at all new to you. In the past, it has been used as an example of an algorithm and programming in BASIC, Forth and other languages of various 8-bit platforms. Below we show the operations performed on the emulator or the real machine to illustrate the use of the compiler. There are no major differences between the two contexts, but of course seeing the modest C64 grind out data and spit out the perfectly functioning executable program in a relatively short time is a considerable satisfaction for those who, like myself, took their first steps in the field on this very computer.

We then start by loading the Shell, the real nerve centre of the software package, and to do this we type the classic command



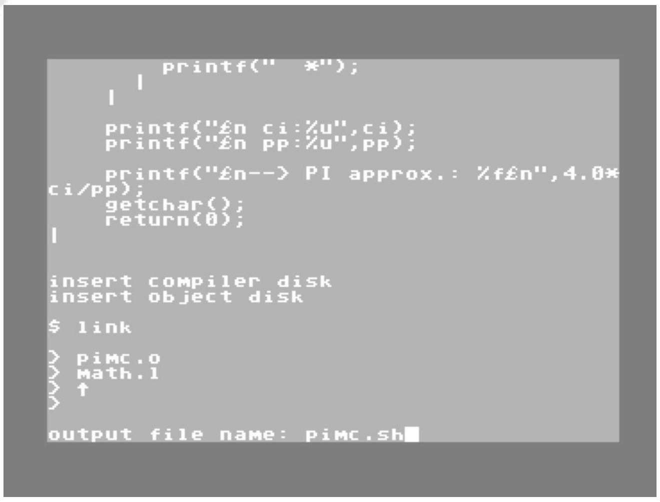
```
$ cc pimc.c

insert source disk

#include <stdio.h>
#define  EOF     -1
#define  NULL     0
#define  stdin    1
#define  stdout   2

typedef int FILE;

char *gets(), *fgets();
#include <math.h>

float exp(), log(), log10(), pow();
float sqrt(), floor(), ceil(), fabs();
float hypot(), cabs(), sin(), cos();
float tan(), asin(), acos(), atan();
float atan2(), sinh(), cosh(), tanh();


float rndgen(_seed) +
    char *s1 = (char *) 0xd012;
    char
```

Fig. 4 - Compilation in progress

LOAD "*",8 <RETURN>.

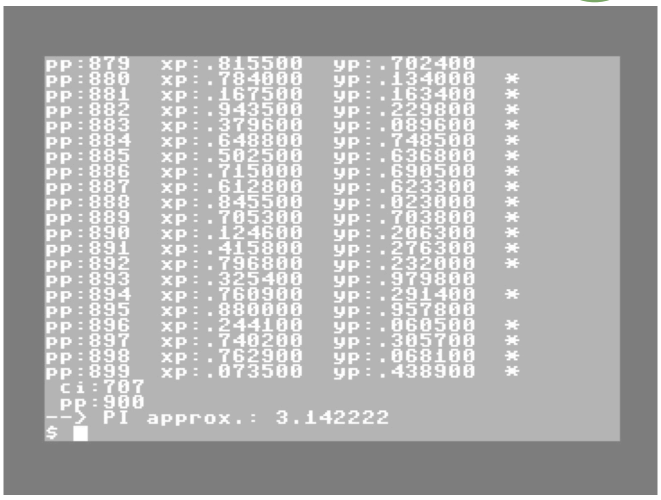
At the end of the short loading process, after the RUN, we will have the $ symbol of the Shell, ready to receive commands to manage the file system and launch Power C programmes. Among the various commands available, we highlight the most important ones, referring to the official documentation for the complete list:
$ l (or ls) [pattern] ; lists the files on the current disk, e.g. ls *.c
$ rm filename ; delete the specified file
$ mv file1 file2 ; move or rename file1 to file2 on the working disk
$ pr [filename] ; displays the contents of the file on the screen
$ pr >> [filename] ; prints the contents of a file to device 4 (printer)
$ disk [command string] ; sends a command to the drive: e.g. disk n0:[headername],[id#] formats the working disk
$ ed [filename] ; run the full screen editor of the given file
$ ced [filename] ; as above but the editor performs syntax checking
$ cc [-p] [filename.c] ; compiles the source code contained in the filename.c file. The -p option tells the compiler that there are two drives in use
$ link [-s [address]] ; launches the linker to finalise the production of an executable. The -s option specifies that the resulting program will be executable without the use of Shell, thus directly from the BASIC of the C64. If specified, [address] indicates the starting memory location of the executable code, default = $0801 beginning of BASIC

Our first step is therefore to open the editor to create our program in C for calculating pi using the Monte Carlo method. In Box 1, you can see the appropriately commented source code. We then type into the Shell:
$ ed pimc.c
; or $ ced pimc.c
The editor will be launched where you can write line after line of the entire programme. The editor is very flexible and offers various commands for editing, searching and formatting code. By pressing the Run/Stop key on the keyboard, the editor will enter command mode in order



```
         printf("  *");
       |
     |

     printf("£n ci:%u",ci);
     printf("£n pp:%u",pp);

     printf("£n--> PI approx.: %f£n",4.0*
ci/pp);
     getchar();
     return(0);
|


insert compiler disk
insert object disk

$ link

> pimc.o
> math.l
> ↑
>

output file name: pimc.sh
```

Fig. 5 - Linking process



```
pp:879   xp:.815500   yp:.702400
pp:880   xp:.784000   yp:.134000   *
pp:881   xp:.167500   yp:.163400   *
pp:882   xp:.943500   yp:.229800   *
pp:883   xp:.379600   yp:.089600   *
pp:884   xp:.648800   yp:.748500   *
pp:885   xp:.502500   yp:.636800   *
pp:886   xp:.715000   yp:.690500   *
pp:887   xp:.612800   yp:.623300   *
pp:888   xp:.845500   yp:.023000   *
pp:889   xp:.705300   yp:.703800   *
pp:890   xp:.124600   yp:.206300   *
pp:891   xp:.415800   yp:.276300   *
pp:892   xp:.796800   yp:.232000   *
pp:893   xp:.325400   yp:.979800
pp:894   xp:.760900   yp:.291400   *
pp:895   xp:.880000   yp:.957800
pp:896   xp:.244100   yp:.060500   *
pp:897   xp:.740200   yp:.305700   *
pp:898   xp:.762900   yp:.068100   *
pp:899   xp:.073500   yp:.438900   *
 ci:707
 pp:900
--> PI approx.: 3.142222
$
```

Fig. 6 - PIMC program running

to issue various management commands, including those to read (GET filename), save (PUT filename), print (PRINT filename), search the disk directory (DIR), search text (/searchstring<F3>), etc. Within the editor some special symbols that are widely used in C can be obtained on the C64 keyboard using the key combination. For example curly brackets {} are obtained with <SHIFT>+ and <SHIFT>-, back-slash '\' with the '£' key, underscore '_' with the Commodore+@ combination, pipe '|' with Commodore+*. The BYE or QUIT commands terminate the editing program and return to the Shell.

Having typed and saved the file with the name pimc.c, we are now ready to compile it with the command

$ cc pimc.c

The compilation process will begin, during which any language syntax errors and other anomalies in the code will be reported. At the end, if all has gone well, we will obtain the object code in the form of a file named with the same name as the source file and with the extension ".o".

All that remains now is to link the object file with any non-standard libraries used in the source code. The linking phase is perhaps the least straightforward procedure of those provided in the Power C development toolchain, but nothing impractical. We then launch the linker programme by specifying the -s parameter if we want our programme to be able to run outside of the Power C Shell, otherwise the linker will automatically produce a file with the extension ".sh" that can only be run in the Shell.

$ link [-s]

The programme will produce a prompt allowing us to specify in order: the object code, the non-standard libraries to be included, and finally the character/command to start the actual process. For our example, since we have made use of mathematical functions in the programme, in addition to the object code pimc.o we will also include the math.l library.

>pimc.o

>math.l

> 🡅

The 🡅 key/character ("up arrow", not the "cursor up" key) is used to start the linking process at the end of which we will obtain the executable programme. In our case we will get the file pimc.sh (executable via shell) or pimc.prg (loadable and executable from the BASIC environment of the C64). The name of the executable file can be given at this stage when the linker requests it. In the event of an error, it will be necessary to provide the linker with the necessary non-standard function libraries. Of course, in the test we conducted, the emulated 1581 drive allows us to avoid all the diskette changes we would have to face in the case of a system with two 1541 drives or a single drive.

The Power C package provides many other utilities to simplify and optimise all stages of C programme development. It also offers additional libraries provided by third parties to effectively handle the graphics, sound and I/O capabilities (user port and joystick ports) peculiar to the C64 platform.

Conclusions

In the past, several articles have appeared on RMW concerning interpreters and compilers of various more or less popular and widespread languages for 8-bit platforms. For example, the Abacus Super C compiler was extensively reviewed by Francesco Fiorentini, and I must say that nowadays one is always amazed at the level that our beloved little retro computers can still reach after 30-40 years when they run the right development package. Abacus Super C had proved to be an easy-to-use and very versatile and efficient compiler (ref. RMW #32-IT and #10-EN). I too had tried it on an emulator and on a real machine, and I can honestly say that I was impressed by its capability and flexibility. Equally sincerely I can say, without fear of being contradicted, that Power C was and is one of the fastest and most powerful compilers for Commodore's 8-bit computers. Learning to use it as an expert and in a productive way is not an immediate process, but the learning curve, although seemingly longer than that of other compilers, holds many rewards along the way. The Shell's operating tool is truly surprising: with due proportions, it is like having a Bash for Linux, dynamic and full of useful commands and functions for managing the filesystem and the development environment. Really unexpected for a system with only 64KB of RAM and a 1MHz CPU. The compiler and linker are fast and useful when checking C code, the full-screen editor in its small size does not make one regret the IDEs of much more powerful platforms than the C64, and the whole software production chain (edit-compile-link) moves smoothly and without any particular hiccups. After some training with the commands available in the various tools of the package, efficiency, productivity and, why not, fun are guaranteed.

References

C Programming Language Book - https://en.wikipedia.org/wiki/The_C_Programming_Language
DirMaster - https://style64.org/dirmaster
Power C - https://www.lyonlabs.org/commodore/onrequest/powerc/index.html
Power C Wiki - https://www.c64-wiki.com/wiki/Power_C
RMW past issues – https://www.retromagazine.net

```
#include <stdio.h>
#include <math.h>

float rndgen(_seed) {
   // This function's code (c) Marco Spedaletti
    char *s1 = (char *) 0xd012;  //pointer to raster register
    char *s2 = (char *) 0xa2;    //pointer to location 161-162
    char *s3 = (char *) 0xa1;    //jiffy clock updated by KERNAL IRQ every 1/60 second
   int seed = 0;

   if ( _seed ) {
       seed = ((*s1<<8)^(*s2<<8)).((*s3)^(*s2)); // computes seed
   } else {
      seed = ((*s3<<8)^(*s1<<8)).((*s2)^(*s1));
   }

   //generates random number
   srandom(seed);
   unsigned int result = (unsigned int)random();
   return( (result%10000)/10000.0 );
}

main() {
   float xp,yp;
   int pp,np,ci;

   printf("\nCalculus of PI with");
   printf("\nMontecarlo's method.\n");
   getchar();

   np=300;    // no. of iterations
   ci=0;      // counter

   for(pp=0; pp<np; pp+=1) {
     xp=rndgen();
     yp=rndgen();

     printf("\npp:%u  xp:%f  yp:%f",pp,xp,yp);

     if (xp<=sqrt(1-yp*yp)) {
       ci=ci+1;
       printf("  *");  //prints * if point is in
     }
   }

   printf("\n ci:%u",ci);
   printf("\n pp:%u",pp);

   printf("\n--> PI approx.: %f\n",4.0*ci/pp);
   getchar();
   return(0);
}
```

Listing of the pimc.c program for calculating Pi using the Monte Carlo method

Libdragon Development: the kit for creating games on Nintendo64 - part 2

by Takahiro Yoshioka – translation: Carlo Nithaiah Del Mar Pirazzini

Libdragon Console

The Libdragon console is used to display plain text in a simple and secure way on the N64 display. It is mainly used for debugging, but can also be used as an engine for a simple text-based game.

Starting the Console

Starting the console is quite simple, all you need is this following line of code. It will start the console and allow you to start printing text on it.

```
console_init();
```

The console has 28 lines with 64 characters each and is filled 8 pixels vertically and 64 pixels horizontally. You can refer to these values with these macros:

```
#define CONSOLE_WIDTH 64
#define CONSOLE_HEIGHT 28
#define HORIZONTAL_PADDING 64
#define VERTICAL_PADDING 8
```

Optional: set the mode

Two modes are available for printing to the console:

RENDER_MANUAL- Keeps printed instructions in a buffer and waits for them to be called manually by console_render().

RENDER_AUTOMATIC- Renders each console print command as it is printed.

These options are set like this (choose only one line):

```
console_set_render_mode(RENDER_MANUAL);
console_set_render_mode(RENDER_AUTOMATIC);
```

Printing text

Once the console has been activated, it is time to start printing text on it. This can be achieved by using functions that print to stdout, mainly the following:

printf()- Prints a formatted string

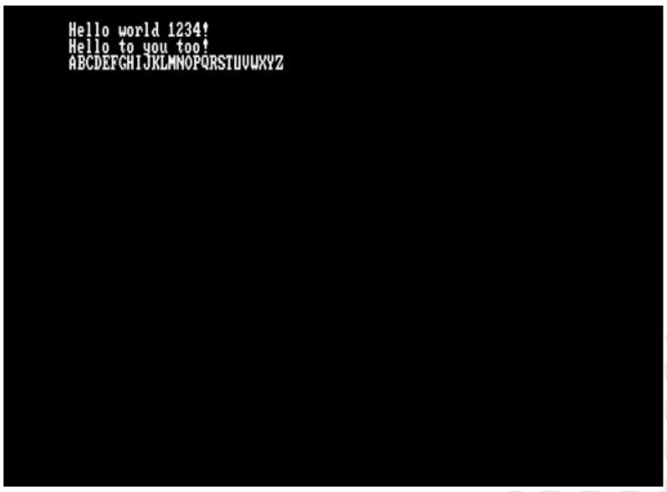
puts()- Prints an unformatted string with an end-of-line character (\n) at the end

putchar()- Prints a single character

These functions are present in the standard <stdio.h> library, so they should be quite familiar.

```
printf("Hello world %i!\n", 1234);
puts("Hello to you too!");
for (char i=0; i<26; i++) {
    putchar(65+i);
}
putchar('\n');
```

This snippet will return something like this:



```
Hello world 1234!
Hello to you too!
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

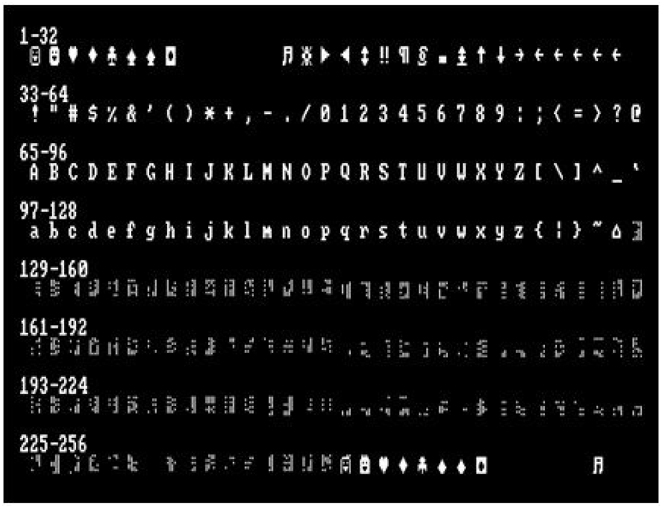
Clipboard

It is important to add a carriage return at the end otherwise the previous text will not be printed. It is possible to print multiple print statements on a line without a carriage return (like putchar()above) as long as it has a \following \note at the end.

Carriage returns (\r) do not work on the Libdragon console, nor is there any way to edit text that has already been printed without deleting and retyping it.

ASCII Table

Here is the ASCII table of printable characters. The main ASCII set (32-127) is more or less what you would expect, but some non-printable characters (1-31) emit some symbols. The extended ASCII characters (128-255) are mostly meaningless or slightly damaged versions of other characters, so they should be avoided unless you are trying to create some sort of ASCII art.



```
1-32

33-64
 ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @
65-96
 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _ `
97-128
 a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
129-160

161-192

193-224

225-256
```

Clearing the screen

If the console prints reach the bottom of the screen, the console will automatically scroll down. Once you reach the bottom of the screen there are only two ways to change what is above it: erase and overwrite.

Overwriting means that you simply keep printing the text to move the console back down until you get it where you want it. It can be a bit tedious to keep track of all the lines, so it is not advisable unless you want to keep some of the previous printouts.

Deletion involves cleaning the console buffer and placing the cursor in the top left of the array as if nothing had been written. This can be done using the following function. It is particularly useful when you wish to write output over several lines on the console, for instance by emulating a GUI in text-only mode.

```
console_clear();
```

Customising the console

You can use the standard white text on a black background, but you can also customise the colour of the console as you see fit. To do this you need to use the graphics_set_colour()function, which accepts two unsigned 32-bit ints as input; the first for the text colour and the second for the background. Here is the prototype of the function:

```
void graphics_set_color (uint32_t forecolor, uint32_t backcolor);
```

Then you can set the colour using an RGBA hexadecimal number or by using the graphics_make_colour() function in this way (both methods do the same thing):

```
// Prototype for syntax purposes only, not needed in actual code
uint32_t graphics_make_color (int r, int g, int b, int a);

graphics_set_color(
    graphics_make_color(255,0,0,255),
    graphics_make_color(0,255,0,255)
    );
graphics_set_color(
    0xF801F801,
    0x07C107C1
    );
```

Please note that the hexadecimal number is 32-bit but the value inside it consists of two 16-bit repeated colours. In addition, the colour is set for the entire console at once, it is not possible to specifically encode certain characters or lines at a time.

Cleaning things up

Once you are finished with the console, you can call the following function to close the console and free up the memory it was using:

```
console_close();
```

Keep in mind that the text will remain on the screen unless you use console_clear(); before closing the console.

Debug

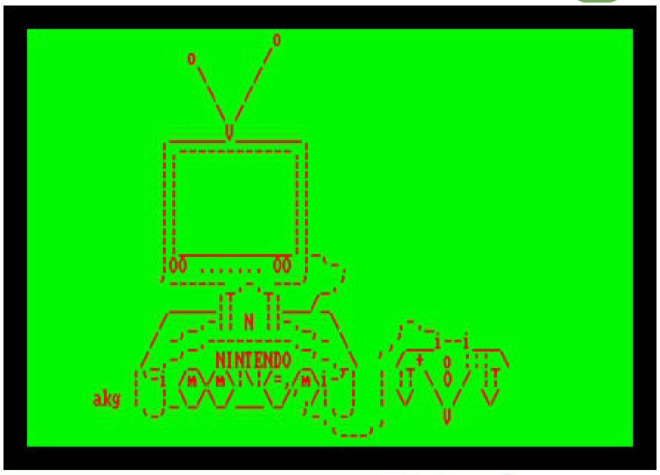
If you use a flash cart with USB debugging, you can use the following to enable debugging using this method:

```
// Enable it
console_set_debug(true);
// Disable it
console_set_debug(false);
```

References

You can follow the pages for more documentation:

- https://libdragon.dev/ref/group__console.html
- https://github.com/DragonMinded/libdragon/blob/trunk/src/console.c
- https://github.com/DragonMinded/libdragon/blob/trunk/include/console.h



Source of the Ascii art: ascii.co.uk

```c
#include <stdio.h>
#include <libdragon.h>

int main(void)
{
    console_init();
    graphics_set_color(
        0xF801F801,
        0x07C107C1
    );
    printf("                          o\n");
    printf("                 o       /\n");
    printf("                  \\     /\n");
    printf("                   \\   /\n");
    printf("                    \\ /\n");
    printf("               ______V________\n");
    printf("              |.------------.|\n");
    printf("              ||            ||\n");
    printf("              ||            ||\n");
    printf("              ||            ||\n");
    printf("              ||            ||\n");
    printf("              ||____________||_\n");
    printf("              |OO ....... OO | `-.\n");
    printf("              '------_.-._---' _.'\n");
    printf("               _____||   ||___/_\n");
    printf("              /  _.-|| N ||-._  \\      .-._\n");
    printf("             / -'_.---------._'- \\     ,'___i--i___\n");
    printf("            /_.-'_  NINTENDO _'-._\\  ' /_+  o :::_\\\n");
    printf("           |`-i /m\\/m\\|\\|/=,/m\\i-'|  | || \\ O / ||\n");
    printf("       akg |  |_\\_/\\_/___\\_/'./|  |  | \\/  \\ /  \\/\n")
    printf("            `-'               '-.`-'  ,      V\n");
    printf("                                `---' \n");

    while(1) {
    }
}
```

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

by Francesco Donatini

Created and currently maintained by Stephane Dallongeville, the Sega Genesis Development Kit (SGDK) is a mature open source framework (under MIT licence) available for Windows, Linux and MacOS X, which allows to develop games for the Sega Megadrive (some titles: The Cursed Knight, Metal Dragon, Tanzer, Demons of Asteborg, or the better known Xeno Crisis) using the C language.

Coming from programming experiences with high-level languages such as Java and C#, and being for a long time looking for a way to obtain decent performances on 8-16 bit machines without passing through assembly, the SGDK has literally constituted my entry point for the development of games on retro machines, so in this and the following articles I would like to share with the readers of our beloved magazine my experiences with this tool, trusting that they could be of interest for other retro coding neophytes like me.

LET'S PREPARE THE SETUP

To start using the SGDK, one can either take advantage of the Docker image or, once we have ensured that we have installed the Java JRE runtime (from version 8 upwards), for those like me who use Windows, we can
- download the latest version of the framework from Github (currently 1.90 from July 2023);
- create a folder on the filesystem to host our project with the following subfolders

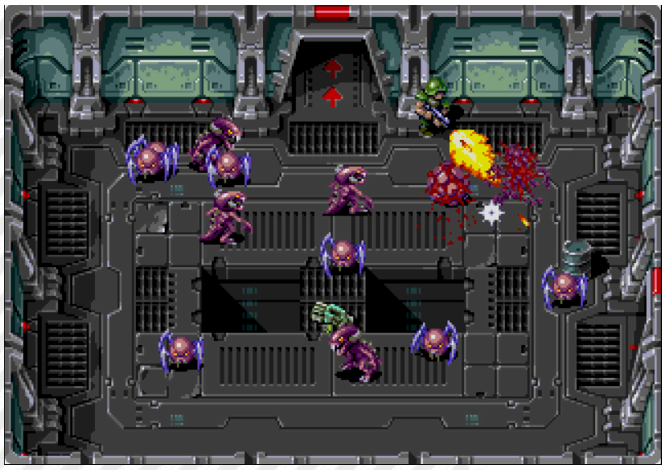
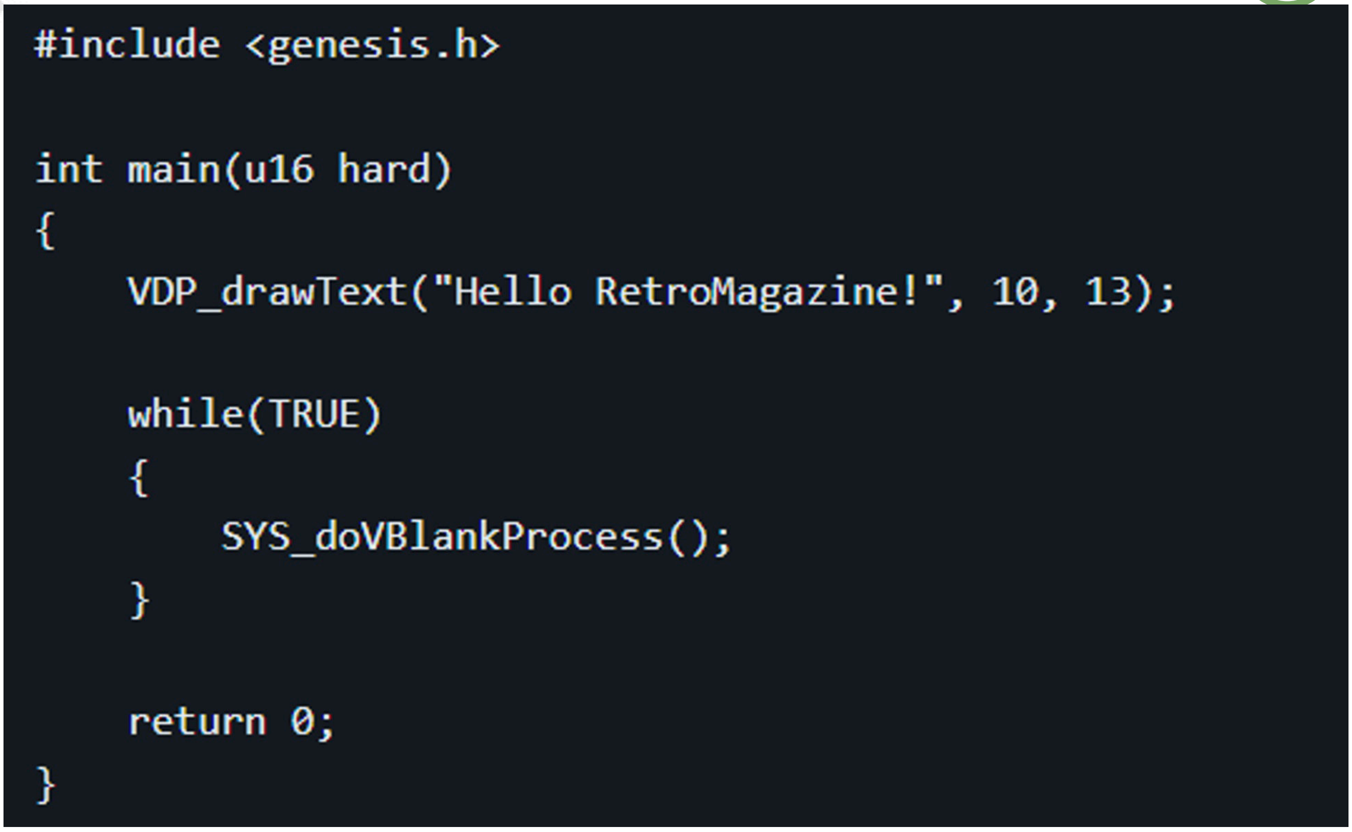


Fig. 1 - Xeno Crisis (2019 - Bitmap Bureau) convertito anche per Switch, PS4, PC, Dreamcast e Neo Geo

Fig. 2 - My Megadrive (with Everdrive) in action at the 4GAME Festival (Cerea - September 16-17, 2023)

- src → where we will put the C source code
- inc → where we put the C include files
- res → where we put the game resources (graphics, audio, etc.)
- out → the compilation result will be automatically generated in this folder
- write the source code;
- compile using the GCC 6.3 binaries included in the downloaded SGDK archive (make.exe in the bin folder) by executing the following command in the root folder of our project: {FOLDER SGDK}\bin\make -f {FOLDER SGDK}\makefile.gen

In case of a successful compilation we will obtain in the "out" folder a "rom.bin" file that if fed to an emulator (or to a real megadrive via a flash cart like the Everdrive) will allow our creation to be executed and played!
Personally, I use Visual Studio Code with zerasul's Genesis Code and Microsoft's C/C++ for Visual Studio Code extensions as my development IDE, and I have prepared a simple BAT file whose execution, associated with an IDE shortcut, starts the compilation of the project and the emulator with the newly compiled game loaded.



```
#include <genesis.h>

int main(u16 hard)
{
    VDP_drawText("Hello RetroMagazine!", 10, 13);

    while(TRUE)
    {
        SYS_doVBlankProcess();
    }

    return 0;
}
```

Fig. 3 - Our Hello World code

WE OBTAIN OUR FIRST ROM

To obtain our first Rom with the classic Hello World, it will be sufficient to create a file with the extension .c containing the code in Fig. 3.

Even those completely unfamiliar with C will not struggle to find an easy-to-read structure consisting of:

- the inclusion of the framework via the genesis.h header;
- the starting point of the programme, which in C is given by the contents of the main function;
- the VDP_drawText command which is responsible for writing the text at a given position on the screen;
- the loop identified by the while loop which is executed ad infinitum;
- the execution within the game loop of an unspecified 'process' contextual to the vertical blank

LET'S BETTER UNDERSTAND WHAT WE HAVE WRITTEN

As is always the case when we get our hands on our beloved retro consoles/computers, in order to know what we are doing and what is possible, we need to know the characteristics of the hardware we are programming on, and the SGDK, although it 'hides' some of what goes on behind the scenes via high-level functions, is no exception.

When the framework library is included, it takes care of a number of preliminary actions before the main function is even executed:

- the 64KB that make up the megadrive's RAM are emptied;
- the VDP (Video Display Processor), which is the custom video chip used for graphics, is prepared with a default configuration;
- the 64KB that make up the VRAM (video RAM), which is RAM dedicated to graphics, are emptied;
- four default palettes are loaded: grey, red, green, blue. The megadrive can display on screen up to four palettes of sixteen colours each from a total of 512 possible;
- a default font is loaded (with which we can display our Hello World);
- input management is initialised;
- the sound system is reset (managed by two chips: Yamaha YM2612 and Texas Instruments SN76489)

When the main function is called, the parameter defined by the unsigned 16-bit hard integer indicates:

- when it is zero, that the megadrive reset button has been pressed;
- when worth one, that the console was switched on using the power button

The text position parameters that we have entered with the VDP_drawText command (10, 13) are not expressed in pixels but in tiles, which are squares of 8x8 pixels and represent the minimal graphic unit that the VDP can represent, so the text will be placed at the coordinates in

0	0	1	1	1	0	0	0
0	1	1	0	1	1	0	0
2	2	0	0	0	2	2	0
2	2	0	0	0	2	2	0
1	1	1	1	1	1	1	0
1	1	0	0	0	1	1	0
1	1	0	0	0	1	1	0
0	0	0	0	0	0	0	0

Fig. 4 - Bit representation of a tile, which occupies 8 bytes in memory, where each byte represents a row and the value of the individual bit represents the color

pixels (80, 10). Behind the scenes, the VDP_drawText method takes care of transferring the graphical data inherent to the text we wish to display (see note in fig. 3) into the VRAM of the VDP (which can accommodate up to 2048 tiles) using a plane (by default plane A) and a transfer method (by default via CPU). The planes, transfer methods and other features of the VDP will be covered in the next article on sprites and the graphics compartment.

When SYS_doVBlankProcess is called inside the game loop, the SGDK waits for the vertical blank (i.e. the time it takes for the raster to return to the first line of the screen once it has finished projecting the last one) in order to perform a number of tasks during that time, (such as transferring data into the VRAM of the VDP) including some that must necessarily be performed frame by frame such as polling the input etc.

Returning to our code in which we are displaying text on the screen, one might expect that by eliminating the SYS_doVBlankProcess command we would no longer see the text on the screen since we would not be performing the activities related to the vertical blank including the transfer of data into the VDP, but in reality the transfer method through the CPU used by default by the VDP_drawText command performs an immediate transfer of data to the VDP and therefore the text will still be visible; on the contrary, if we had specified a transfer method that implements the transfer to the vertical blank, the text would not be visible except by executing the SYS_doVBlankProcess command (you can try this by replacing VDP_drawText with this command: VDP_drawTextEx(BG_A, "Hello RetroMagazine", TILE_ATTR(PAL0, FALSE, FALSE), 10, 10, DMA_QUEUE);

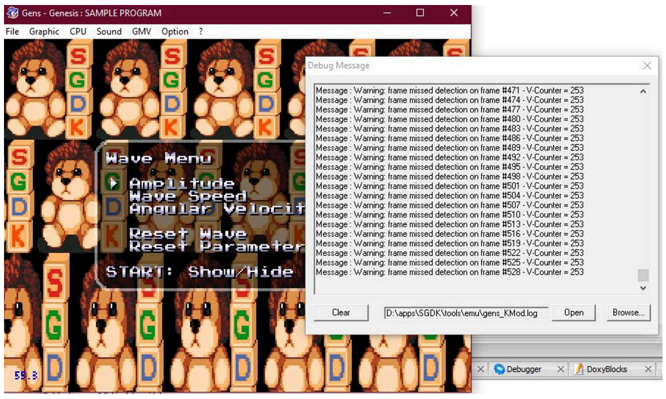
A LOOK AT EMULATORS IN DEVELOPMENT

The emulator that I consider to be the most accurate in my experience, in which I have found an almost perfect correspondence with the real machine (the game crashed both on the emulator and on the real machine at the same point!) is definitely the BlastEm, which also offers the possibility of customising the settings and, for example, starting the emulator in full screen and making it practically 'invisible' so that executables can be generated that start the game directly, without having to use the emulator interface to load the rom.

The other emulator that I use and that I find very nice for the graphic effects is the Kega Fusion, while for debugging purposes (an issue that we will go into in more detail) the Gens_KMod, which used to be included among the framework files themselves, and the Regen Debug Version (v0.972) can also be used.

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



```
Message : Warning: frame missed detection on frame #471 - V-Counter = 253
Message : Warning: frame missed detection on frame #474 - V-Counter = 253
Message : Warning: frame missed detection on frame #477 - V-Counter = 253
Message : Warning: frame missed detection on frame #480 - V-Counter = 253
Message : Warning: frame missed detection on frame #483 - V-Counter = 253
Message : Warning: frame missed detection on frame #486 - V-Counter = 253
Message : Warning: frame missed detection on frame #489 - V-Counter = 253
Message : Warning: frame missed detection on frame #492 - V-Counter = 253
Message : Warning: frame missed detection on frame #495 - V-Counter = 253
Message : Warning: frame missed detection on frame #498 - V-Counter = 253
Message : Warning: frame missed detection on frame #501 - V-Counter = 253
Message : Warning: frame missed detection on frame #504 - V-Counter = 253
Message : Warning: frame missed detection on frame #507 - V-Counter = 253
Message : Warning: frame missed detection on frame #510 - V-Counter = 253
Message : Warning: frame missed detection on frame #513 - V-Counter = 253
Message : Warning: frame missed detection on frame #516 - V-Counter = 253
Message : Warning: frame missed detection on frame #519 - V-Counter = 253
Message : Warning: frame missed detection on frame #522 - V-Counter = 253
Message : Warning: frame missed detection on frame #525 - V-Counter = 253
Message : Warning: frame missed detection on frame #528 - V-Counter = 253
```

D:\apps\SGDK\tools\emu\gens_KMod.log

Fig. 5 - The Gens_KMod emulator with an example loaded from the SGDK and the window with debug messages

Identity Cases... - part 1

by Marco Pistorio

Welcome back, fellow readers of RetroMagazine World!
As some of our most faithful readers may know, I have always been involved in computer science in my everyday life. I took my first steps along this tortuous path by studying programming with my trusty "breadbin", my Commodore 64, and still today, in the (little) free time I manage to carve out for myself from time to time, I try to make something for this machine that I still adore.

It has to be said that they are in good company. The admirers of these wonderful machines are many and still growing, unbelievable to say. And many of them are really good, really talented.
Perhaps it is because the information available today is numerous and varied and it is easy and immediate to access. Perhaps the nostalgia of those green years, when these machines were brand new and many of us were kids or little more, also plays its part. Probably there are other ingredients that make these old machines so attractive and that make their programming interesting and enjoyable even today, despite the fact that their hardware is certainly not recent and extremely limited, for example if one compares their technical characteristics with those of today's PCs.

I anticipate that this article is very technical and is aimed, in particular, at all those who have some development experience on Commodore 64 (or similar machines) and who use modern programming tools, such as IDEs, cross-assemblers, data compression tools, etc., for their work.

What exactly is it about? It is mainly an experience that I decided to share with you, dear readers of RetroMagazine World.
Probably some of the conclusions I came to will be useful (I hope) to you as well.

Where did all this come from that I want to tell you about? From a 'big' problem that I have, alas, set myself. I found a joking definition of computer scientist on the Net that I think really fits, in this particular case. :D
The definition goes something like this: 'A computer scientist is a person who constantly solves problems you didn't even know existed, in ways you don't understand'.
At this point, I present my dilemma.
A few years ago now, I developed some work, simple games, simple 'intros' in assembly for the C64. I then decided to publish most of these works on the CSDB website.
For those who are curious to see them, I post the link for you to click on immediately below:

https://csdb.dk/scener/?id=29165

For those of you who prefer YouTube videos, I post below the link to my personal channel, where you can find some of these works that I have published on CSDB:

https://www.youtube.com/@mpsoftita-retroprogramming8233/

Well, the problem I faced was: "how do I know that the executable I published matches EXACTLY what I get by recompiling the source code in my possession?"
Many of you will be thinking right now, "well, if you own the source code, where is the problem? No doubt it is straightforward to get the executable code from it..."

Here, as a die-hard Star Trek fan, I am reminded of a phrase uttered by Mr Spock in the film Star Trek-Route to the Unknown: "Logic is only the premise of wisdom, Valeris, not its epilogue."
I will now explain why this sentence comes to mind.
It is very true and immediate that by recompiling the source code, one eventually obtains the corresponding executable code.
Developers know this all too well.

However, obtaining an executable that is 'perfectly identical' to another, starting from the same source code, is not quite so simple.
The compiler that is used during the two different operations MUST be exactly THE SAME, and this must apply to any subsequent steps (e.g. the tool that is used to compress the data in order to obtain an executable code that is smaller in terms of memory occupation).
But sometimes this is not enough.

One usually has to work with the same operating system to achieve perfect binary compatibility between the two executables. Sometimes with the same system updates, which modify certain libraries and bring them to the same precise version, so that the executable code obtained in the second instance is perfectly analogous to that obtained previously.
Moreover, it should be pointed out that these latter elements (system updates and libraries) are important when generating executable code intended to run within the operating system of a PC.

So, to return to Mr. Spock's maxim, being able to recompile the original source code is only a prerequisite for being able to obtain an executable that is an exact duplicate of a previously generated executable.
Here is a 'real' programmer's problem, isn't it? I refer here to the joking definition of a programmer that I stated earlier. :D

I was presented with a particular case of one of my intros that I made in 2017 and tried to compress it (after recompiling it, of course) by employing different versions of the data compression tool I used at the time, which is called 'EXOMIZER', and I could not get the same, identical result as I got then.
After several unsuccessful attempts, I decided to give up.
It is clear that by launching the executable I get from the compilation, the intro behaves as expected.
However, the code I get, after compressing it, does not have the exact same 'signature' as the one I published in 2017.
There is little I can do. The signatures I get are always different.
I remembered that my production machine was then a Linux machine and therefore, most likely, the EXOMIZER works differently when running on a Windows platform than on a Linux platform.

Fortunately, in several other cases, examining other works of mine published at different times, the outcome was favourable :) (see fig. B).

So here is another concept I wanted to share with you. I was talking about 'signature' just now. What is it?
Let us first reflect on a fundamental, I would say basic aspect.
An executable programme, from the point of view of our PC, is always and in any case a file.
A file is a set of characters (i.e. bytes) stored in the PC's memory.
It is therefore possible to examine any executable programme that is in such a form, byte by byte, i.e. character by character, in order to calculate a value, a checksum, a 'signature' of such a file.
This 'signature' must be different if even a single byte examined in the set under analysis is different.
Various methods can be used to calculate these signatures. Personally, I prefer MD5, which can also be easily implemented in simple .Net applications (see Box 1 and Fig. A).
For those who wish to explore this topic further:

https://it.wikipedia.org/wiki/MD5

Now that we know that it is possible to generate 'MD5 checksums' (or 'MD5 Hash') from our executable programmes, let's talk about what is good to avoid if we are interested in analysing our work using these methods.
One of these things that should be avoided is the systematic inclusion of one or more randomly obtained characters in our programmes at each compilation.

Specifications of this kind in our assembly programme (the example is in Kick Assembler syntax):

```
................
................

random:
.fill 256,round(15*random())

................
```

would make the signatures of our exegibles different each time we generate them.
Why? Because we would introduce into them a sequence of random X-characters (256 in the example), which would be recalculated each time we recompile. Result? Always different 'signatures'.
A possible remedy? Generate a segment containing such bytes only once and then link it to our source programme, so that it always uses the same set of random bytes even if we have to recompile it several times.

Another practice that I strongly suggest you adopt is to ALWAYS write down, perhaps in special textual notes

within the source code of your programmes, the name and version of the compiler used, as well as the names of all the tools that may be used in cascade, together with their version numbers, and finally the generation date and the platform used for development.
This will make it easier for you to 'put the pieces back together' if necessary.

But, suppose we have two executables with two different signatures.
How do we determine whether they can be traced back to the same source code or not?
This is the topic I will try to explore in the next installment. For the moment, there is already enough meat on the fire! :D

A warm greeting to all of you, fellow readers. Until next time!

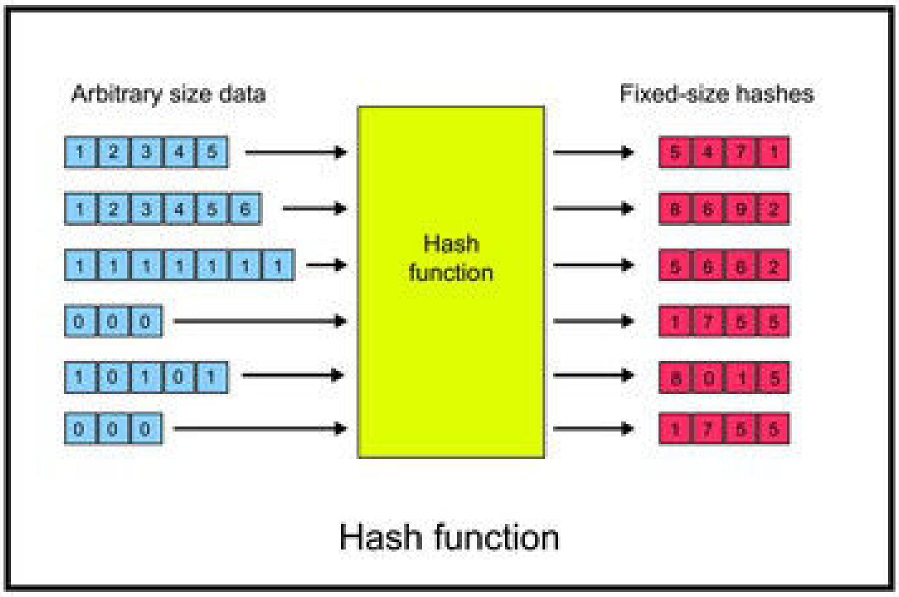


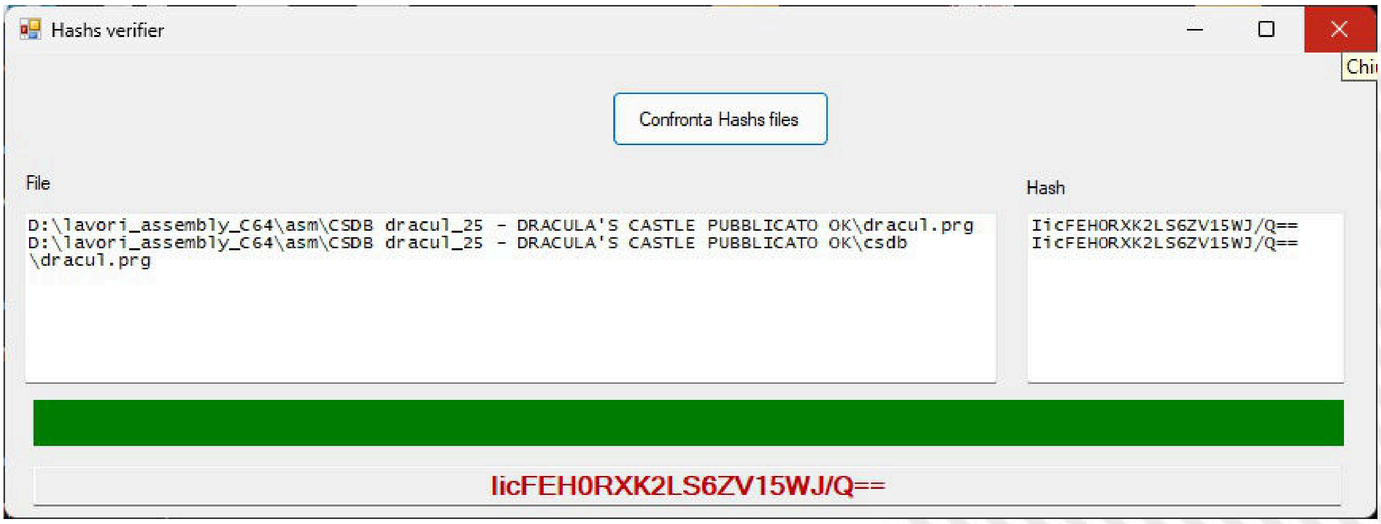
Fig. A - Conceptual diagram of the HASH function (source: https://stock.adobe.com/search?k=md5)

Elem contiene il nome del file, completo del suo path, da esaminare

Public Function calcola_hash(elem As String) As String

```
        Dim fileBytes() As Byte
        Dim Md5 As New MD5CryptoServiceProvider()
        Dim byteHash() As Byte
        fileBytes = File.ReadAllBytes(elem)
        byteHash = Md5.ComputeHash(fileBytes)
        Return Convert.ToBase64String(byteHash)

    End Function
```

Box 1 - Example of VB.net code to determine the MD5 HASH calculation of a specific file



```
D:\lavori_assembly_C64\asm\CSDB dracul_25 - DRACULA'S CASTLE PUBBLICATO OK\dracul.prg
D:\lavori_assembly_C64\asm\CSDB dracul_25 - DRACULA'S CASTLE PUBBLICATO OK\csdb
\dracul.prg
```

```
IicFEHORXK2LS6ZV15WJ/Q==
IicFEHORXK2LS6ZV15WJ/Q==
```

IicFEH0RXK2LS6ZV15WJ/Q==

Fig. B - Comparison of the MD5 HASH of the game I published on CSDB "Dracula's Castle" in 2017 with the one I recently recompiled. The comparison result is positive, so the two .prg files are absolutely identical, down to the last byte

PunyInform, a new library for writing text adventures for old computers

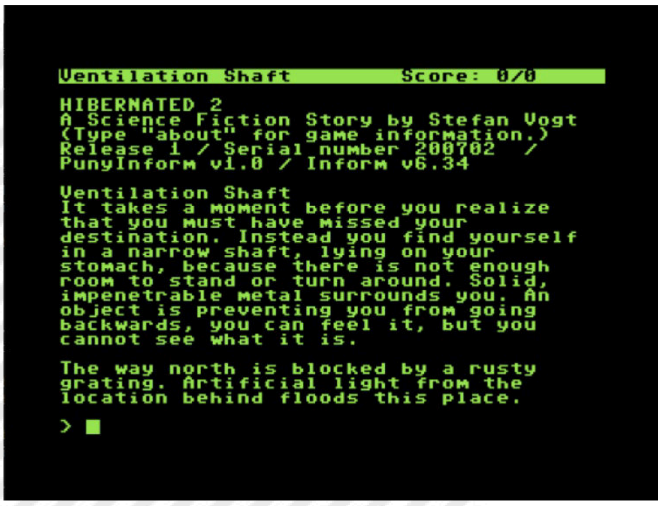
by Fredrik Ramsberg, translated and adapted by Gianluca Girelli

Initial note: this article is part of a series of five tutorials written by Fredrik Ramsberg, and reproduced here with the author's permission.
Fredrik, with his friend and colleague Johan Berntsson, developed both PunyInform (a compact library for developing text adventures on 8-bit machines) and Oozmo (a Z-code interpreter that can be used to make PunyInform games playable on many Commodore machines). Fredrik and Johan are also the organisers of PunyJam (now in its 4th edition), a friendly (but no less fierce) competition where text adventures are developed within a time limit. The subject of the adventure changes from jam to jam and is communicated just before it starts.

PunyInform is a library for the Inform 6 programming language, developed to facilitate the creation of simple or advanced text adventures that run well on 8-bit computers such as Commodore 64, Amstrad CPC, Apple II, Spectrum +3 etc. The first version of PunyInform was released on 5 July 2020.

The roots

The Inform programming language, compiler and standard library were developed by Graham Nelson in the 1990s to provide a way to write new text adventures and compile them into Z-code, the game format invented by the founders of the text adventure company Infocom in 1979. By the end of the 1990s, Inform had reached version 6 and was a rather mature, C-like language specialised for the task of writing text adventures. The standard library was also very good. At that time, few people were interested in using the slow computers of the 1980s, so the library was never developed or tested for those old computers. Two decades later, retro computing became a big thing and being able to use this excellent language and library to write 8-bit computer games was a goal to be achieved. Unfortunately, however, the language worked well but the library was too big and too slow to be suitable for 8-bit computers.



```
Ventilation Shaft              Score: 0/0

HIBERNATED 2
A Science Fiction Story by Stefan Vogt
(Type "about" for game information.)
Release 1 / Serial number 200702  /
PunyInform v1.0 / Inform v6.34

Ventilation Shaft
It takes a moment before you realize
that you must have missed your
destination. Instead you find yourself
in a narrow shaft, lying on your
stomach, because there is not enough
room to stand or turn around. Solid,
impenetrable metal surrounds you. An
object is preventing you from going
backwards, you can feel it, but you
cannot see what it is.

The way north is blocked by a rusty
grating. Artificial light from the
location behind floods this place.

> ■
```

Fig. 1: Hibernated 2, an upcoming commercial text adventure by Stefan Vogt, which runs on an (emulated) Commodore 64

A smaller and faster library

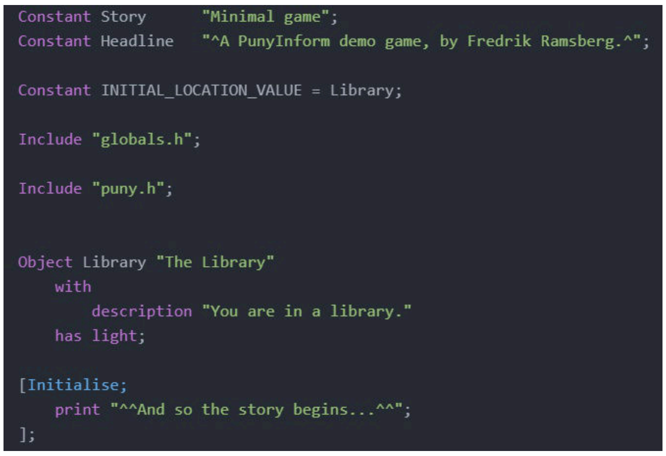
PunyInform is a library in which the main parts have been written from scratch, with an emphasis on small size and efficient code. From a game programmer's point of view, it is very similar to the standard library of Inform 6 and many routines were copied from said library, but each routine was carefully examined to see if it could be made faster and more compact. While the smallest possible game using the standard library is 59 KB, the smallest possible game using PunyInform is only 24 KB. It has most of the functionality of the standard library, including an advanced parser, a good set of standard verbs and a system of attributes and properties that decide the behaviour of each object in the game world. Some functionality has been made optional, to allow the game author to skip some less necessary parts to save space. Support is available for the addition of NPCs (people, animals, etc. with whom the player can interact), conversations, doors, darkness, timers/fuses (code that is executed after a certain number of turns), daemons (code that is executed every turn) and much more.

The principle behind the library is that everything an adventure author needs on a regular basis should be provided by the library from the outset, so the developer generally does not have to write ad-hoc code, but can also choose to modify all these features and add new ones.

Programming in PunyInform

Writing a minimal game involves the definition of two constants for the game name and title, a constant for the initial position of the player, two 'Include' statements to include library files, the definition of an object for the initial position, and an 'Initalise' routine to print an initial text file. That is all.

Suppose you want to create an empty kitchen and a bedroom with a wardrobe. The player should be able to open and close the wardrobe and even lock and unlock it if he has the right key, he can put things inside it, but it must be too heavy to move. Also, place the key inside the cupboard. This is child's play in PunyInform.

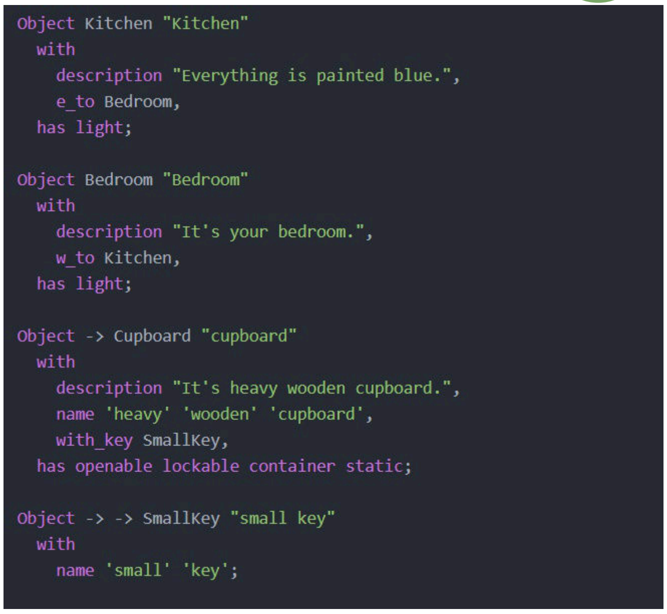


```
Constant Story      "Minimal game";
Constant Headline   "^A PunyInform demo game, by Fredrik Ramsberg.^";

Constant INITIAL_LOCATION_VALUE = Library;

Include "globals.h";

Include "puny.h";


Object Library "The Library"
    with
        description "You are in a library."
    has light;

[Initialise;
    print "^^And so the story begins...^^";
];
```

Fig. 2: A minimalist game

By using the standard properties (description, name, with_key etc.) and attributes (openable, lockable, light etc.) most things can be made to work without the need to write special code. However, when you have an object that cannot be defined in this way, you can add code to the object to make it react the way you want. This allows for incredible flexibility. Suppose you want the cupboard door to be locked at the start of the game, so opening it will fail on the first attempt but will be possible on the second.

"General" is an attribute that can be used for a wide variety of reasons. It is a customised variable, automatically defined locally for each object. In this case it means "Will the cabinet open easily?". Here is how the resulting game looks in Windows Frotz, one of the most popular Z-code



```
Object Kitchen "Kitchen"
  with
    description "Everything is painted blue.",
    e_to Bedroom,
  has light;

Object Bedroom "Bedroom"
  with
    description "It's your bedroom.",
    w_to Kitchen,
  has light;

Object -> Cupboard "cupboard"
  with
    description "It's heavy wooden cupboard.",
    name 'heavy' 'wooden' 'cupboard',
    with_key SmallKey,
  has openable lockable container static;

Object -> -> SmallKey "small key"
  with
    name 'small' 'key';
```

Fig. 3: Utilisation of properties and attributes

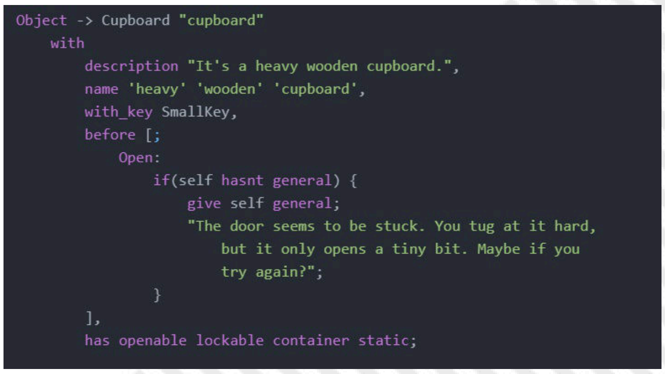
interpreters.

Documentation

PunyInform comes with full documentation. Since it is very similar to the standard Inform 6 library, you can start by reading the official Inform Designer Manual, 4th edition (DM4), and then take a look at the PunyInform manual to see what is different. Or, of course, you can always dive straight into programming and look for the things you need as you go along.

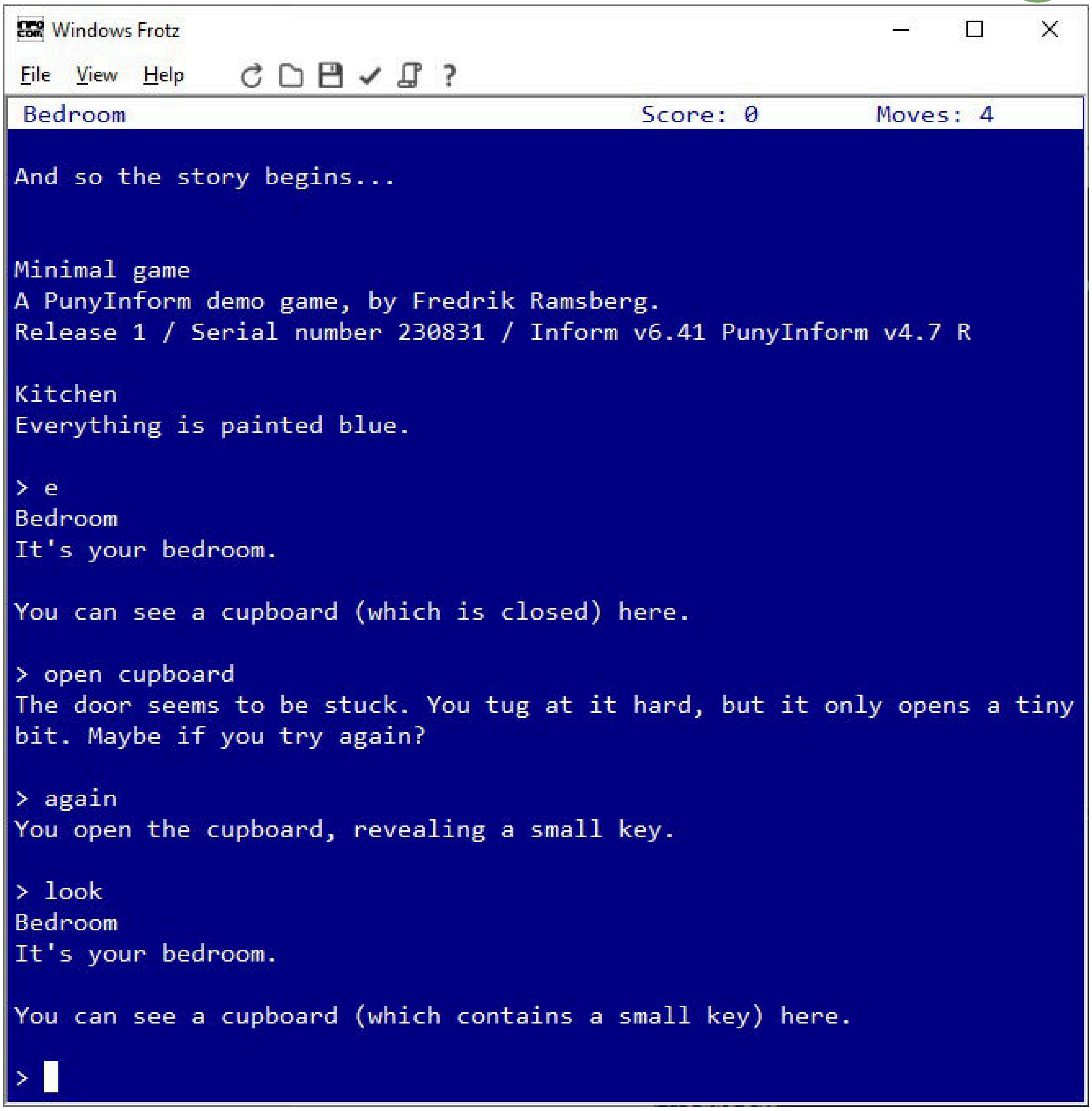
Conclusions

In this first article, we have analysed the basics of the Inform6 language and the PunyInform library, seeing how, with a lean structure and a few simple commands, it is possible to build a text adventure. In the next tutorial, we will focus on how to organise and initialise a development environment. Next, we will get into the thick of the action by developing simple (but highly reconfigurable) games. Have fun!



```
Object -> Cupboard "cupboard"
    with
        description "It's a heavy wooden cupboard.",
        name 'heavy' 'wooden' 'cupboard',
        with_key SmallKey,
        before [;
            Open:
                if(self hasnt general) {
                    give self general;
                    "The door seems to be stuck. You tug at it hard,
                        but it only opens a tiny bit. Maybe if you
                        try again?";
                }
        ],
        has openable lockable container static;
```

Fig. 4: Addition of a 'before' rule



```
Bedroom                                  Score: 0        Moves: 4

And so the story begins...


Minimal game
A PunyInform demo game, by Fredrik Ramsberg.
Release 1 / Serial number 230831 / Inform v6.41 PunyInform v4.7 R

Kitchen
Everything is painted blue.

> e
Bedroom
It's your bedroom.

You can see a cupboard (which is closed) here.

> open cupboard
The door seems to be stuck. You tug at it hard, but it only opens a tiny
bit. Maybe if you try again?

> again
You open the cupboard, revealing a small key.

> look
Bedroom
It's your bedroom.

You can see a cupboard (which contains a small key) here.

>
```

RESOURCES:

Main project page for PunyInform: https://github.com/johanberntsson/PunyInform

PunyInform releases: https://github.com/johanberntsson/PunyInform/releases

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

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

A friendly forum to ask for help: https://intfiction.org/

A Discord server for support and discussions about PunyInform: https://discord.gg/zyYvb8JjE6

Ozmoo, a Z-code interpreter that can be used to make PunyInform games usable on Commodore 64/128/Plu4 and MEGA65: http://microheaven.com/ozmooonline/
Official PunyJam#4 page: https://itch.io/jam/punyjam-4

Tribute to Doriath (Merlino)

by Eugenio Rapella

Starting in the early eighties, Italian newsstands began to fill up with magazines, connected to Vic20 - Spectrum - C64 etc., first with listings, then with cassettes and finally with floppy discs.

My first contacts with an electronic computer dated back to the early years of university (this was around ... 1973!); at that time, I really would not have imagined that, in the future, I would have a 'personal computer'; just 'personal'. All mine! To use in my own home!

Naturally, as soon as it became available, I got myself a Vic20 and then switched to a Commodore 128, which I almost always used in the C64 version.

While I was dabbling with my little programs (I especially liked to use the Commodore in 'stochastic simulations':

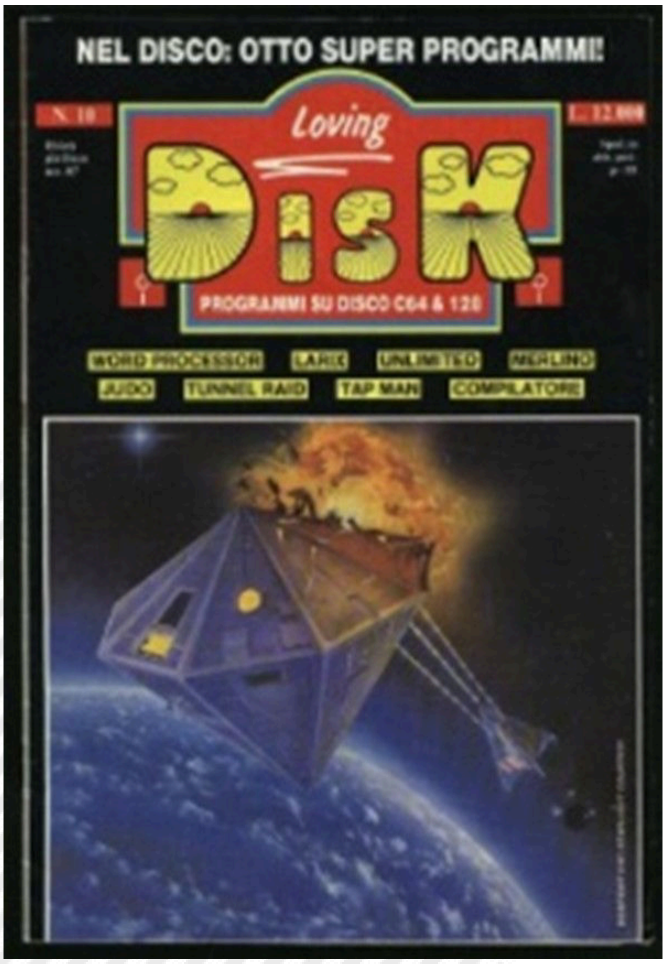
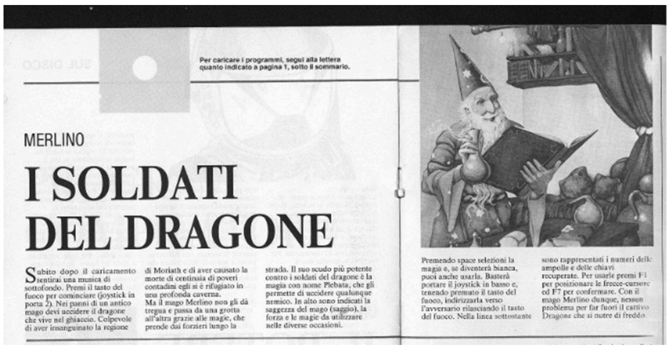


Fig. 1 - The cover of 'Loving Disk' No. 10; we are in 1987

I would often invent a 'probabilistic' game, determine the optimal strategy and put it to the test by having the C64 play thousands of games to see if my strategy turned out to be a winner even ... in the field), I would systematically buy a few magazines and try my hand at some of the games on the enclosed cassette or floppy.

Mind you, even then I was not a kid; I was not particularly interested in most of the games (such as the 'shoot-em-ups') and usually only played a couple of games before giving up, except for ... Merlin!

Inside magazine No. 10, only a few lines were dedicated to the game identified as Doriath/'Merlino':



Per caricare i programmi, segui alla lettera quanto indicato a pagina 1, sotto il sommario.

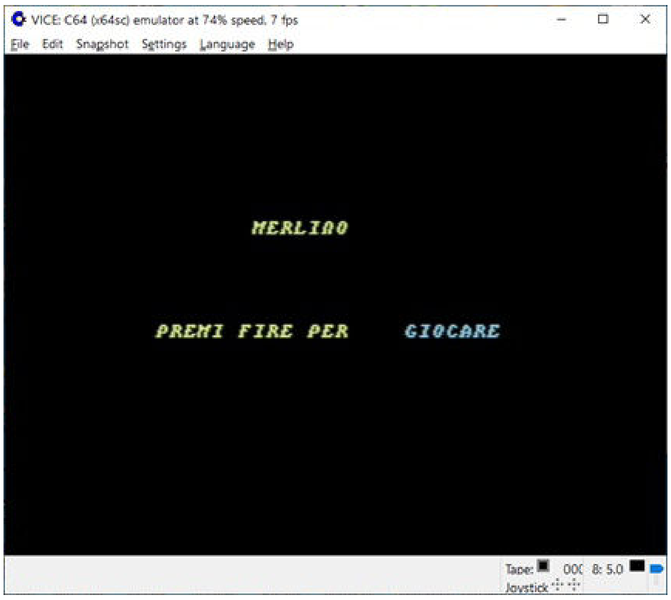
MERLINO

I SOLDATI DEL DRAGONE

Subito dopo il caricamento sentirai una musica di sottofondo. Premi il tasto del fuoco per cominciare (joystick in porta 2). Nei panni di un antico mago devi uccidere il dragone che vive nel ghiaccio. Colpevole di aver insanguinato la regione di Moriath e di aver causato la morte di centinaia di poveri contadini egli si è rifugiato in una profonda caverna.
Ma il mago Merlino non gli dà tregua e passa da una grotta all'altra grazie alle magie, che prende dai forzieri lungo la strada. Il suo scudo più potente contro i soldati del dragone è la magia con nome Plebata, che gli permette di uccidere qualunque nemico. In alto sono indicati la saggezza del mago (saggio), la forza e le magie da utilizzare nelle diverse occasioni.

Premendo space selezioni la magia e, se diventerà bianca, puoi anche usarla. Basterà portare il joystick in basso e, tenendo premuto il tasto del fuoco, indirizzarla verso l'avversario rilasciando il tasto del fuoco. Nella linea sottostante sono rappresentati i numeri delle ampolle e delle chiavi recuperate. Per usarle premi F1 per posizionare le frecce-cursore ed F7 per confermare. Con il mago Merlino dunque, nessun problema per far fuori il cattivo Dragone che si nutre di freddo.

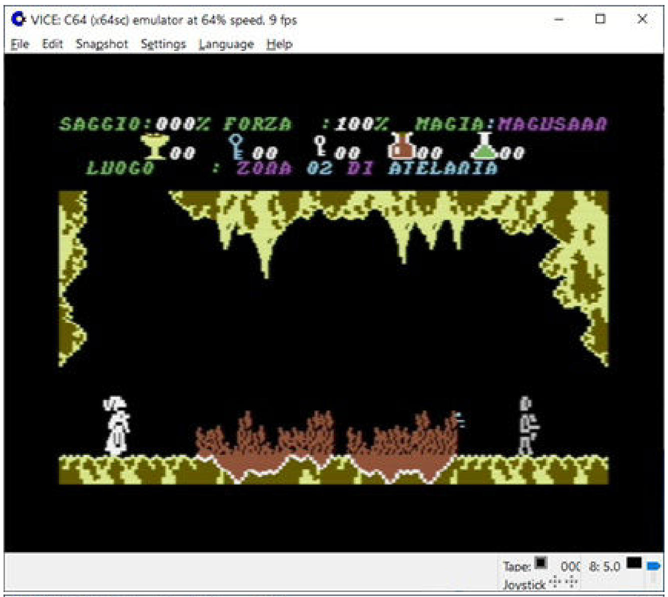
The presentation is very rough and some essential information is missing. Launching the game resulted in::

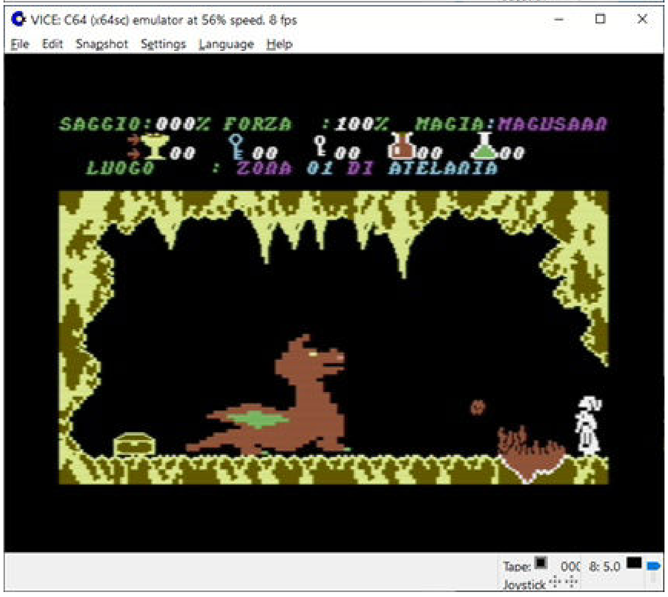

VICE: C64 (x64sc) emulator at 74% speed, 7 fps
File Edit Snapshot Settings Language Help
Tape:
Joystick

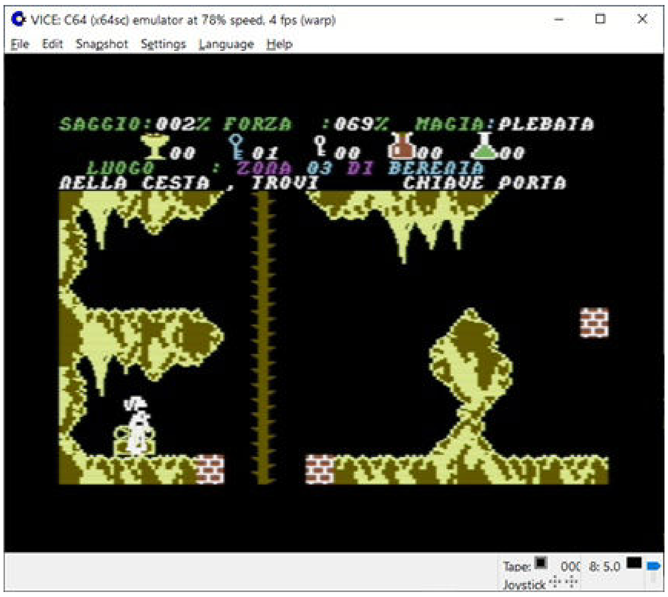
MERLINO

PREMI FIRE PER GIOCARE

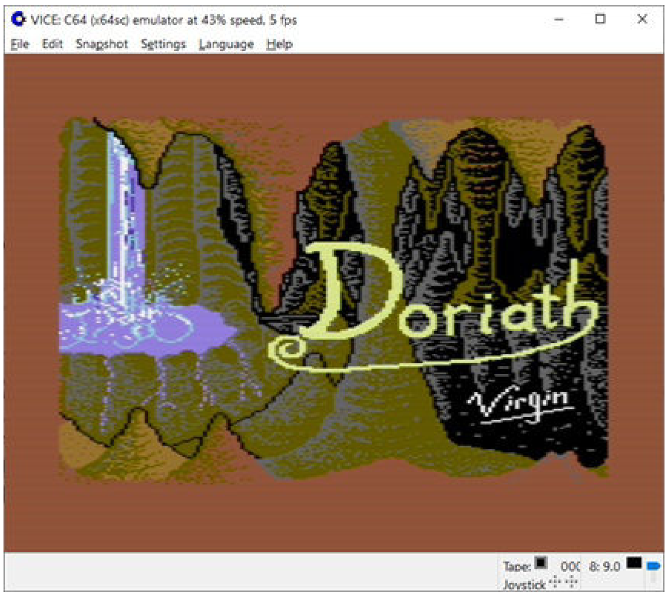
Here are three screens at the start of the game:

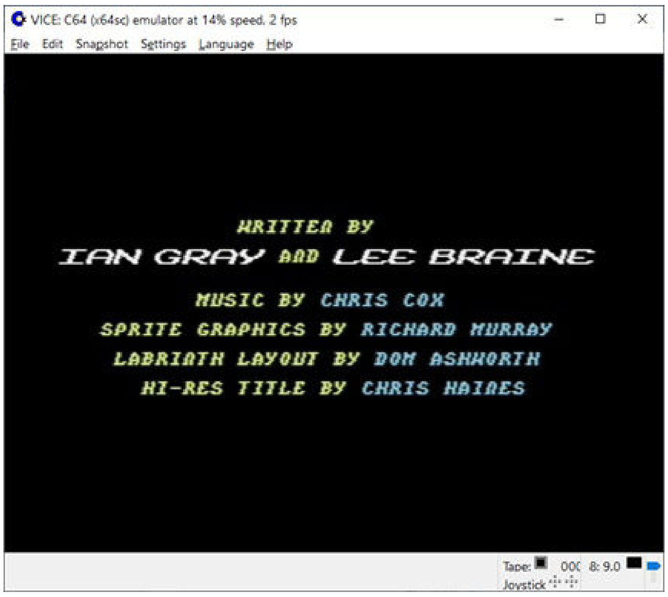

VICE: C64 (x64sc) emulator at 64% speed, 9 fps
File Edit Snapshot Settings Language Help
SAGGIO:000% FORZA :100% MAGIA:MAGUSAAN
00 00 00 00 00
LUOGO : ZONA 02 DI ATELANIA
Tape: 000 8: 5.0
Joystick


VICE: C64 (x64sc) emulator at 56% speed, 8 fps
File Edit Snapshot Settings Language Help
SAGGIO:000% FORZA :100% MAGIA:MAGUSAAN
00 00 00 00 00
LUOGO : ZONA 01 DI ATELANIA
Tape: 000 8: 5.0
Joystick


VICE: C64 (x64sc) emulator at 78% speed, 4 fps (warp)
File Edit Snapshot Settings Language Help
SAGGIO:002% FORZA :069% MAGIA:PLEBATA
00 01 00 00 00
LUOGO : ZONA 03 DI BERENIA
NELLA CESTA , TROVI CHIAVE PORTA
Tape: 000 8: 5.0
Joystick

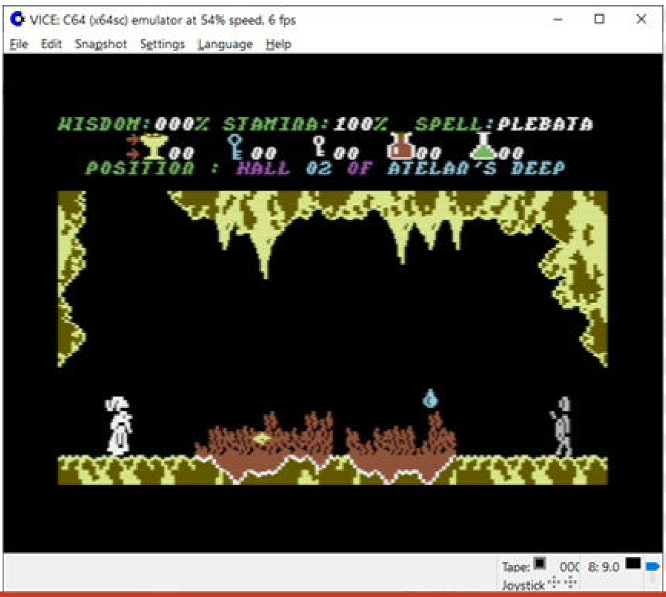
We all know that, in those days, we were in a kind of Wild West: games 'cloned' from games bought abroad appeared in the magazines of the time without the publishers worrying too much about copyrights or the like. Several years later I struggled a bit before I discovered that Melino's original was ... Doriath. The initial screens of Doriath, obviously absent in Merlin, are these:

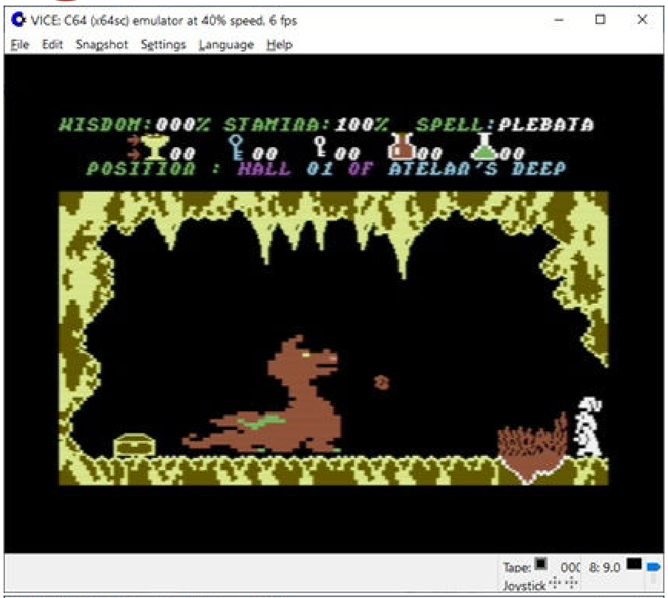

VICE: C64 (x64sc) emulator at 43% speed, 5 fps
File Edit Snapshot Settings Language Help
Doriath
Virgin
Tape: 000 8: 9.0
Joystick

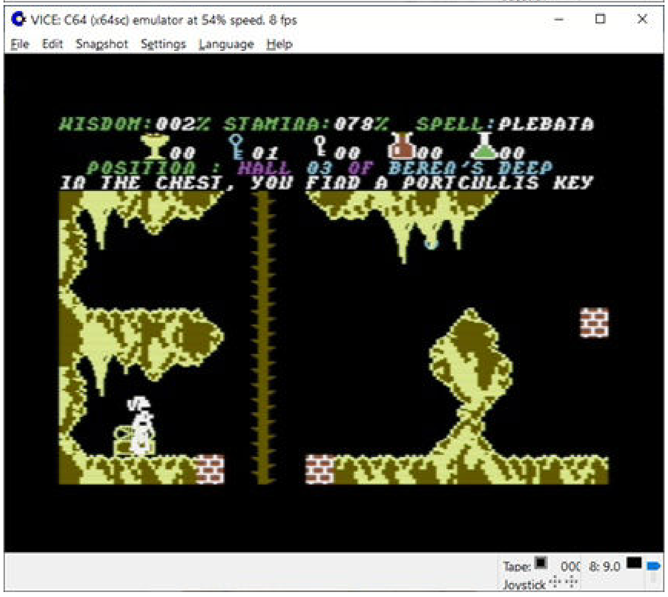

VICE: C64 (x64sc) emulator at 14% speed, 2 fps
File Edit Snapshot Settings Language Help
Tape: 000 8: 9.0
Joystick

WRITTEN BY
IAN GRAY AND LEE BRAINE
MUSIC BY CHRIS COX
SPRITE GRAPHICS BY RICHARD MURRAY
LABRINTH LAYOUT BY DOM ASHWORTH
HI-RES TITLE BY CHRIS HAINES

And here are the three original screenshots at the beginning of the game:


VICE: C64 (x64sc) emulator at 54% speed, 6 fps
File Edit Snapshot Settings Language Help
WISDOM:000% STAMINA:100% SPELL:PLEBATA
00 00 00 00 00
POSITION : HALL 02 OF ATELAN'S DEEP
Tape: 000 8: 9.0
Joystick


VICE: C64 (x64sc) emulator at 40% speed, 6 fps
File Edit Snapshot Settings Language Help
WISDOM:000% STAMINA:100% SPELL:PLEBATA
POSITION : HALL 01 OF ATELAN'S DEEP
Tape: 000 8: 9.0
Joystick


VICE: C64 (x64sc) emulator at 54% speed, 8 fps
File Edit Snapshot Settings Language Help
WISDOM:002% STAMINA:078% SPELL:PLEBATA
POSITION : HALL 03 OF BEREN'S DEEP
IN THE CHEST, YOU FIND A PORTCULLIS KEY
Tape: 000 8: 9.0
Joystick

As can be seen, in Merlino the English lettering has been replaced by Italian in order to maintain the same length.

But back to 'Merlino': the game mechanism was interesting and the soundtrack was ... hypnotic. Me and two friends of mine would get together for what we called a '*merlinata*' (i.e. have a play at Merlino). It was necessary to agree in advance to have a good interval of time because ... it was not possible to 'save the game' and, hear hear hear, there was (and there is) only one life available: one distraction and you had to start again from the beginning, collect all the amulets needed to destroy the various enemies.
In the 'baskets' you retrieve keys, restorative potions, amulets and even mysterious 'scroll fragments' (there is no mention of these scroll fragments in the article 'I **Soldati del Dragone**').
Once, late in the game, in a desperate attempt to find out if there was a way to record the situation on a floppy disk, we pressed the 'S' key for a possible 'Save' command. Surprise! Pressing 'S' brings up a message that, it later turns out, takes shape as we get these blessed fragments. What is the objective of the game? The article associated with the magazine talks about 'killing the dragon that lives in the ice'. Although it is nowhere stated, you should know that you must first collect all the 'scroll fragments' and reach 100 per cent wisdom.

Does Doriath retain its charm today? If you want to give it a try, stop reading and go download the game (you can play 'Merlin' by downloading 'Loving Disk' No. 10 from
https://ready64.org/download/scheda_download.php?id_download=814
or download "Doriath" from
https://csdb.dk/search/?seinsel=releases&search=Doriath
if you choose "Doriath +15", you also have the option of ... cheating);
draw the map yourself as you progress through the game. The joystick goes to port 2. To switch from one 'spell' to another (marked in the top right) you use the space bar. The available spells are written in white; at the beginning there is only the one called 'Plebata'.

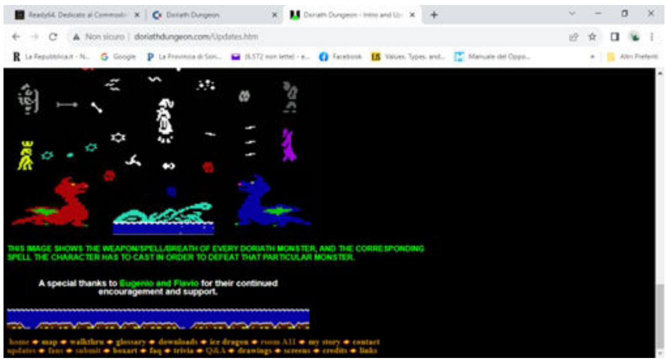
To use an available spell you move the joystick down by pressing 'Fire': the spell appears and, keeping the Fire active, you place it on the opponent (opponents of different types require different spells) then release the Fire.

The double red arrow indicates the usable element among those marked in the second row; you switch from one to the other by pressing F1 and use it by pressing F7.

The first of the five elements, the chalice, indicates the restorative potions available to you: drinking one of them will return your strength to 100%. If your strength drops to 0 %, the game ends and you will have to start from scratch. So watch out for this percentage! The restorative potions scattered around are numerous: don't skimp. As mentioned before, pressing S will read the scroll fragments found up to that point.

Actually, there are also a few ... mysteries ("no Quasilin Amulet?"; "how do you access the right side of the map?"; "room A11 unreachable?"), but first-time players have come to know about them ... while playing.
In any case, you absolutely must visit at some point:
http://www.doriathdungeon.com/
where you will find everything, but absolutely everything, about Doriath.

There is also a 'fan submission' section with contributions from fans from various parts of the world; there is also something from me and my son Flavio (the real Doriath expert in the family):


THIS IMAGE SHOWS THE WEAPON/SPELL/BREATH OF EVERY DORIATH MONSTER, AND THE CORRESPONDING SPELL THE CHARACTER HAS TO CAST IN ORDER TO DEFEAT THAT PARTICULAR MONSTER.

A special thanks to Eugenio and Flavio for their continued encouragement and support.

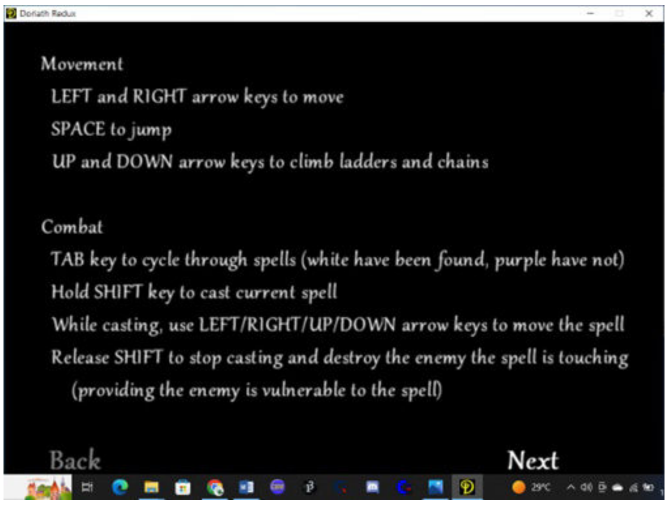
doriathdungeon.com/Updates.htm
home • map • walkthru • glossary • downloads • ice dragon • room A11 • my story • contact
updates • fans • submit • boxart • faq • trivia • Q&A • drawings • screens • credits • links

At https://www.lemon64.com/game/doriath you can find articles about Doriath that appeared in magazines of the time, listen to and download the soundtrack, read commentaries and much more.

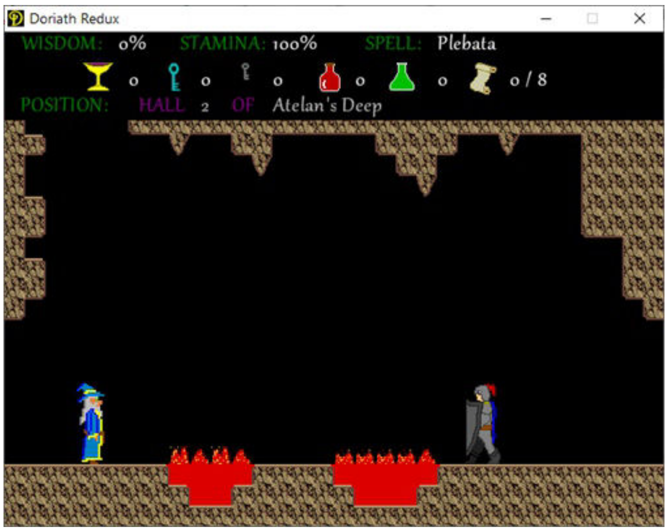
A longPlay of over two hours is visible on YouTube (but Doriath can be solved in less time):
https://www.youtube.com/watch?v=RPMYlXvsaJM&t=339s

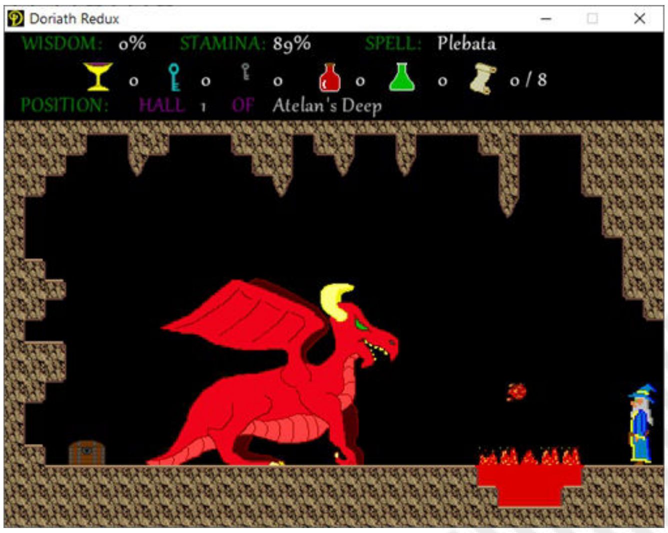
A porting of Doriath for PC is Doriath Redux easily retrievable from the Internet: https://gamejolt.com/games/doriath-redux/87317

Here are some screenshots of Doriath Redux:


Movement
LEFT and RIGHT arrow keys to move
SPACE to jump
UP and DOWN arrow keys to climb ladders and chains

Combat
TAB key to cycle through spells (white have been found, purple have not)
Hold SHIFT key to cast current spell
While casting, use LEFT/RIGHT/UP/DOWN arrow keys to move the spell
Release SHIFT to stop casting and destroy the enemy the spell is touching
(providing the enemy is vulnerable to the spell)





I cannot conclude this homage to Doriath without offering you the map made, back in the day, by Luigi, one of the three participants in the 'merlinate'.
Isn't it a masterpiece?

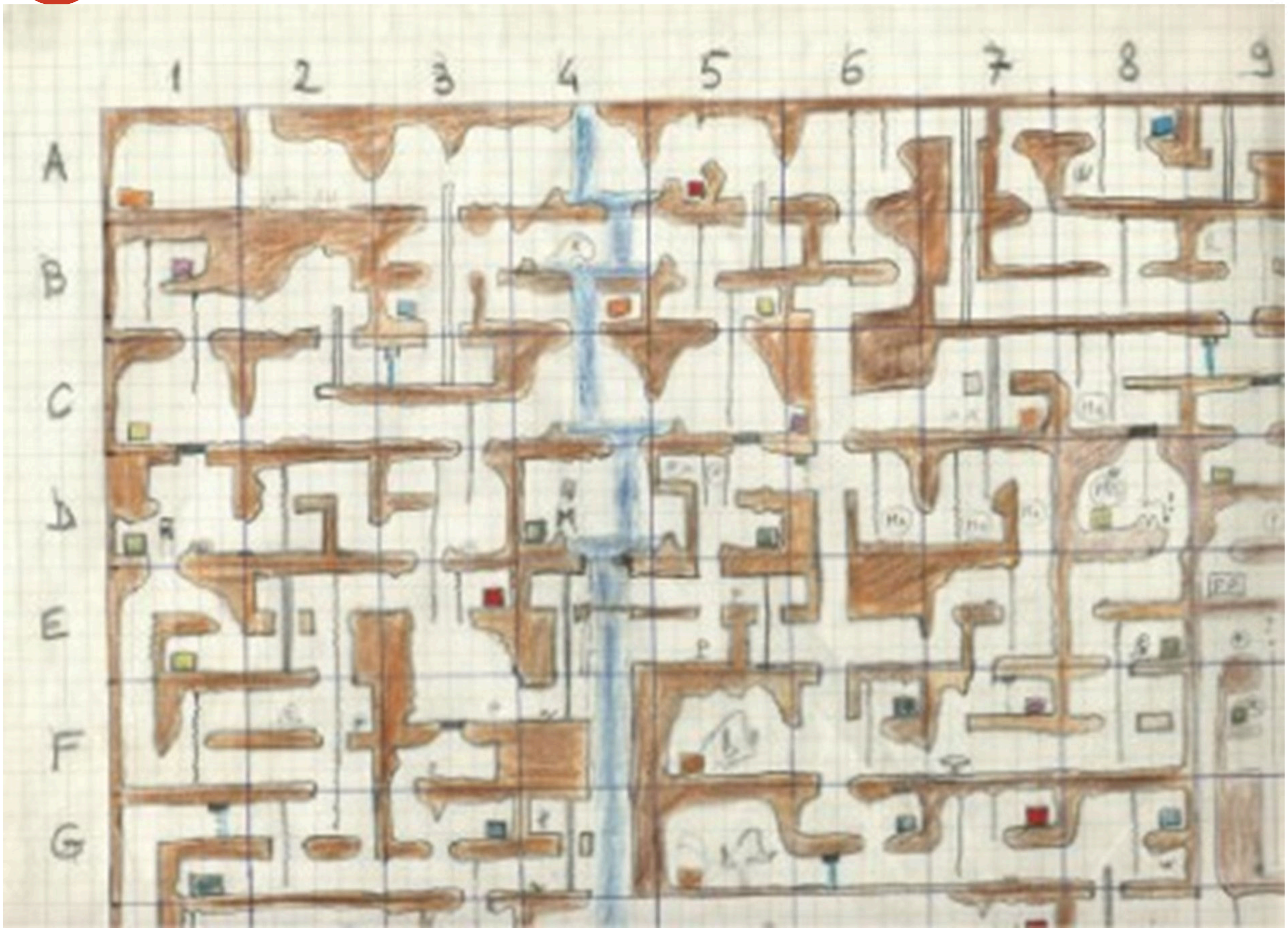


Fig. 2 - Upper left-hand corner of the map

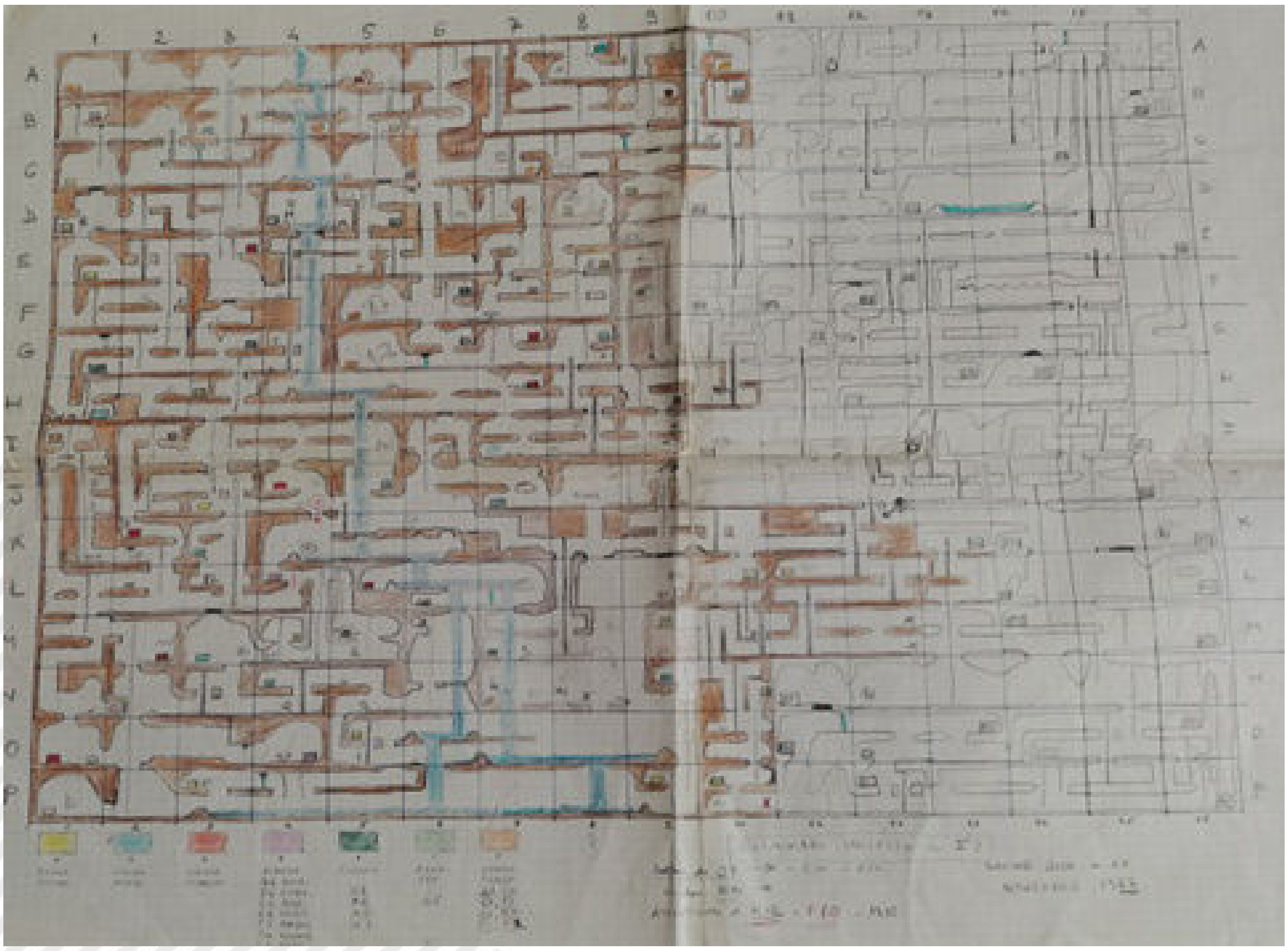


Fig. 3 - The complete map made by Luigi (that's 16*16=256 environments!)

NEW GAME

DRACULA X: NOCTURNO IN THE MOONLIGHT – ULTIMATE

Year: 2024
Editor.: Konami/Meduza Team
Genre: Metroidvania
Platform: Sega Saturn
Website: http://meduza-team.ucoz.net/news/dracula_x_nocturne_in_the_moonlight_ultimate_sega_saturn/2024-01-01-48

Castlevania: Symphony of the Night is one of the jewels in the catalogue of the first Playstation.

Released in 1997, it took the gameplay of the Castlevania saga and cleverly combined it with the non-linear exploration of Super Metroid by adding typical role-playing game elements complete with experience points and equipment.

The end result was a revolution that changed a genre, or rather invented a new one, 'Metroidvania'.

But it was not exclusive to the Sony system; Konami also released it for the Sega Saturn, complete with everything and some additional levels, but the console's poor sales in the West meant that it was never released outside Japan.

In a way, it was a blessing of sorts as the Saturn version had some flaws and shortcomings compared to the Playstation version, due to a porting handled by Konami's Nagoya studio, which was done in a hurry.

But now it is here among us! We have an English language version with some really excellent improvements. The retrogaming community has finally found a way to make this game usable for those who don't know Japanese, with a nice big patch that not only translates the text, but adds some much-needed improvements.

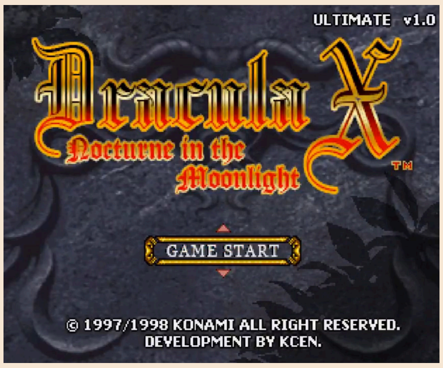
The translation project took a long time, but the 'final' version is worth waiting for and testing.

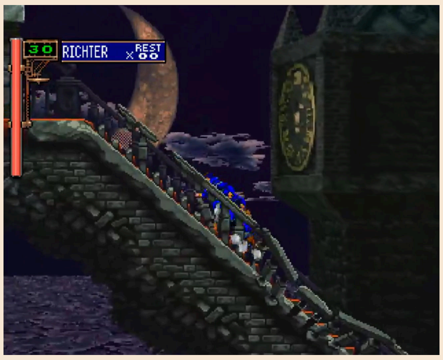
Using the PSP version as a template, this patch for Dracula X: Nocturn in the Moonlight (this is the game's Japanese title) reduces loading times, fixes some glaring bugs and even allows the Saturn's 4 MB RAM expansion cartridge to be used.

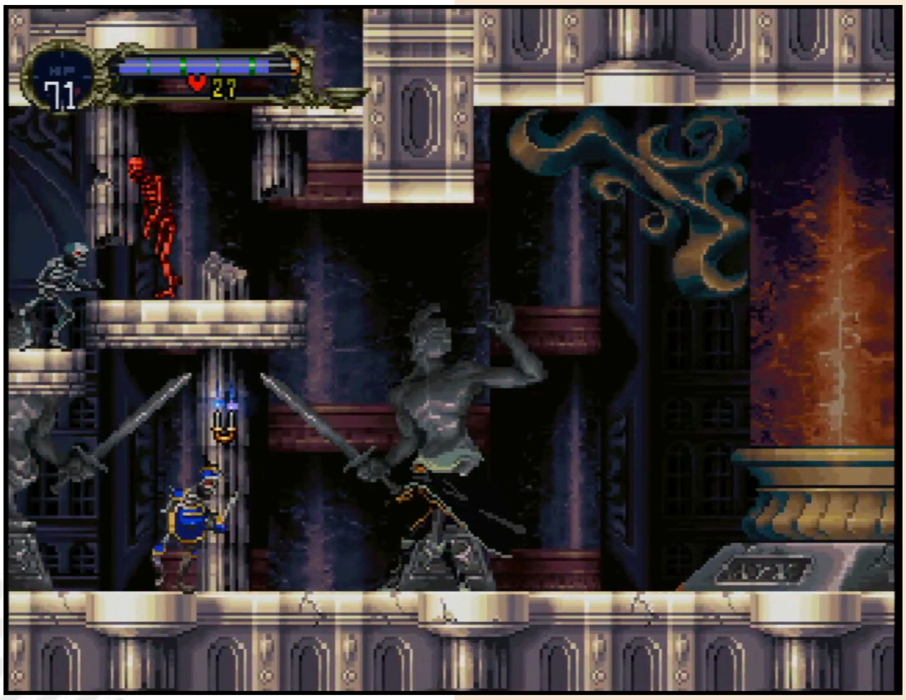
All the transparency effects that were missing in the original port were added, the sprites of the protagonists were resized and the animations were fixed.

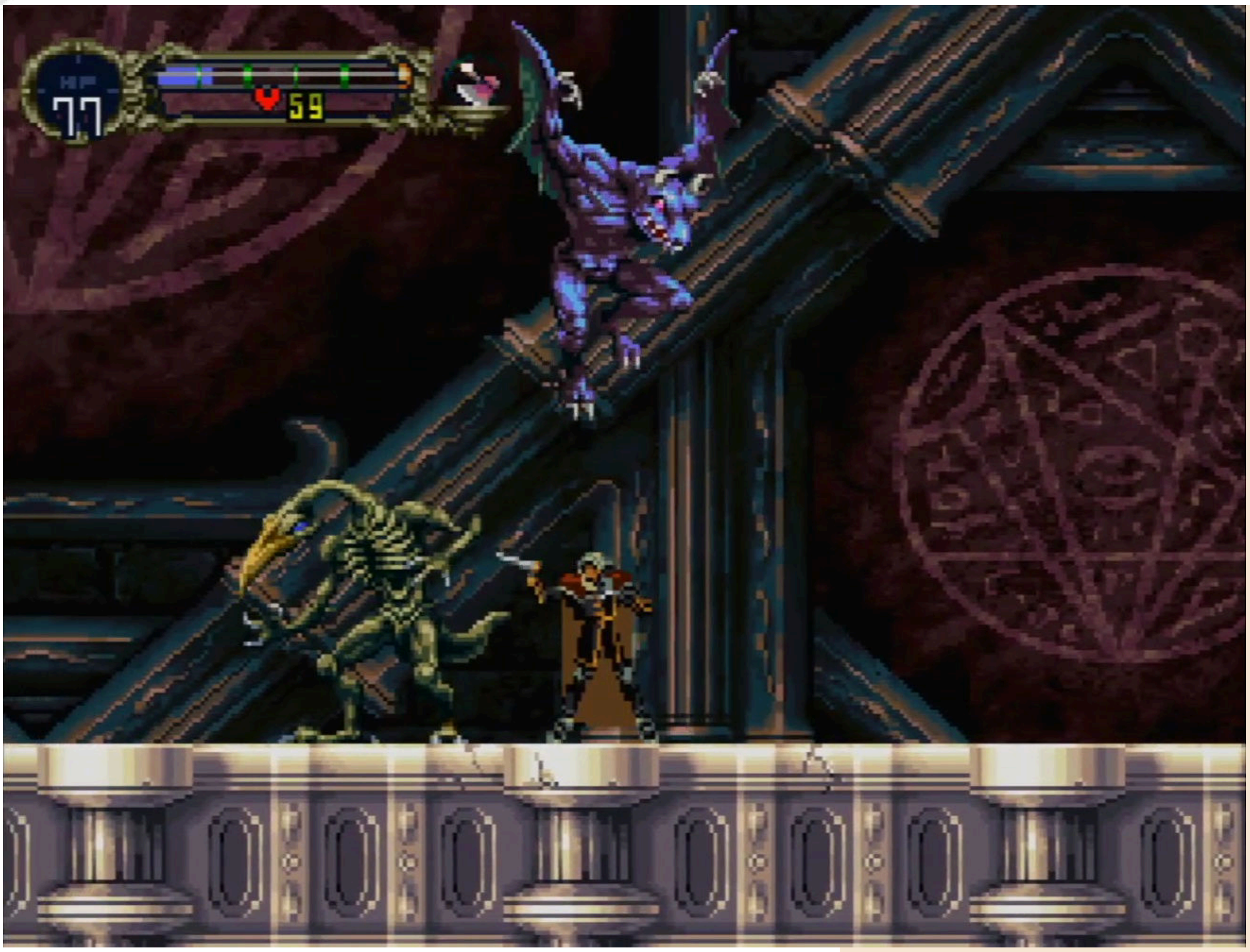
The game is always the same. It uses two-dimensional side-scrolling gameplay. The objective is to explore Dracula's castle to take down the entity called Shaft that controls the body of Richter Belmont, self-proclaimed lord of the castle and absolute hero.

Once defeated, it is possible to enter the second portion of the game, the


ULTIMATE v1.0
Dracula X
Nocturne in the Moonlight
GAME START
© 1997/1998 KONAMI ALL RIGHT RESERVED.
DEVELOPMENT BY KCEN.







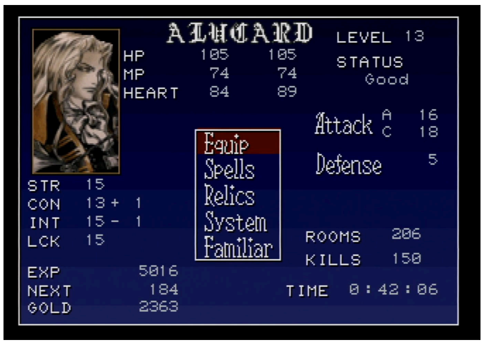
Inverted Castle, an inverted version with new enemies and bosses.

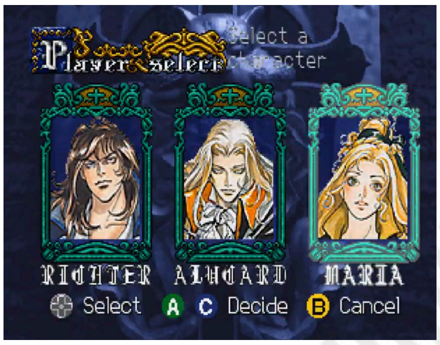
The non-linear gameplay, with most of the castle inaccessible until the various items and abilities are collected, is still addictive.
A title to rediscover if you are a Saturn owner or want to try it out in emulation.

The patch must be applied to a rom of the Japanese game and possession of the original title is obviously recommended.

Now we just have to wait for the Sega Megadrive demake/port of this gem, but in the meantime let's enjoy this super work.

by **Carlo Nithaiah Del Mar Pirazzini**





OUR FINAL SCORE

» Gameplay 90%
A title that opened up a genre that can now also be played on the Saturn in a nice, understandable and improved form.

» Longevity 95%
Difficult but rewarding gaming experience.

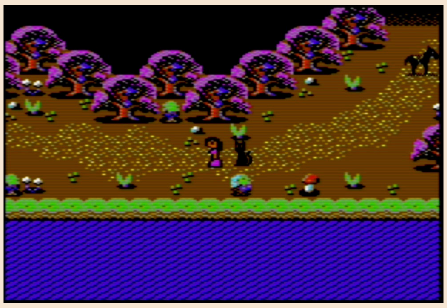
NEW GAME

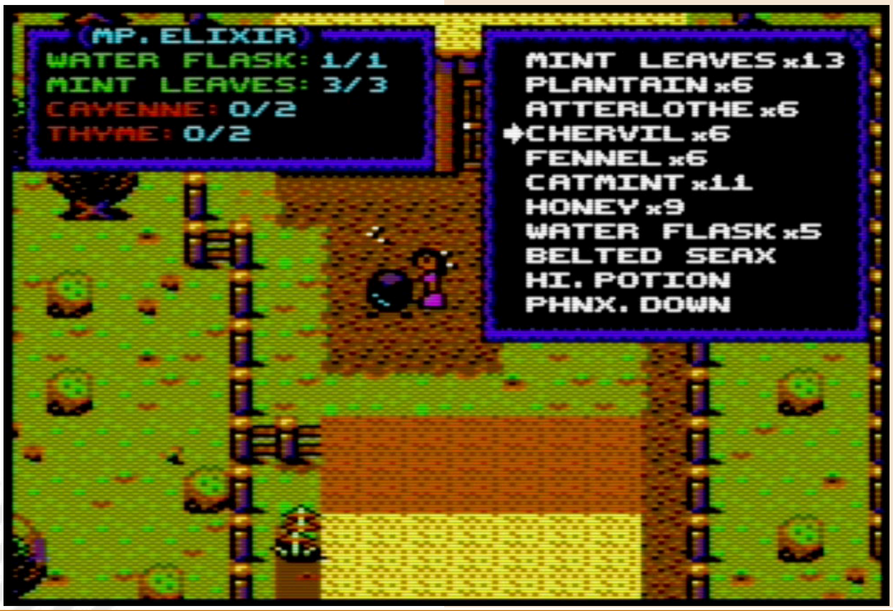
BRILEY WITCH CHRONICLES 2

Year: 2023
Editor/Developer: Sarah Jane Avory
Genre: JRPG
Platform: Commodore 64
Website: https://sarahjaneavory.itch.io/briley-witch-chronicles-2-c64

The saga of the little witch Briley returns and the characters from the village of Meapole return.
Something ominous is about to happen in the young witch's life and unfortunately her destiny will be changed forever.
Unveiled on 1 December, Briley Witch Chronicles 2 is a direct sequel to the award-winning first chapter by Sarah Jane Avory in 2021.
The video game saga is based on the Briley Witch Chronicles Book series, and in the specific case of this second episode, volumes 5 and 6.
In this new adventure we find Briley and his friend Smokey much grown up. Our heroine is now an expert witch.
The game actually starts with quite a few more experience points (a good 55000) than in Chapter 1, which saw us as young neophytes, and we will have a grimoire full of spells and 160 magic points at our disposal. A nice amount of mana that will allow us to deal with the terrible charge that will be placed in front of us as the story unfolds.
The game starts with a well-crafted and rather long introduction that explains what is happening and immediately puts us in play as... a baker!
Yes! We start by delivering bread through the village until we reach the climax of the story.
All this while finding ourselves fighting spiders, snakes and monsters of various kinds.
Gameplay in pure JRPG style, like its predecessor, with fights where we can select the type of attack, defence and magic reminding us so much of the very first Final Fantasy.

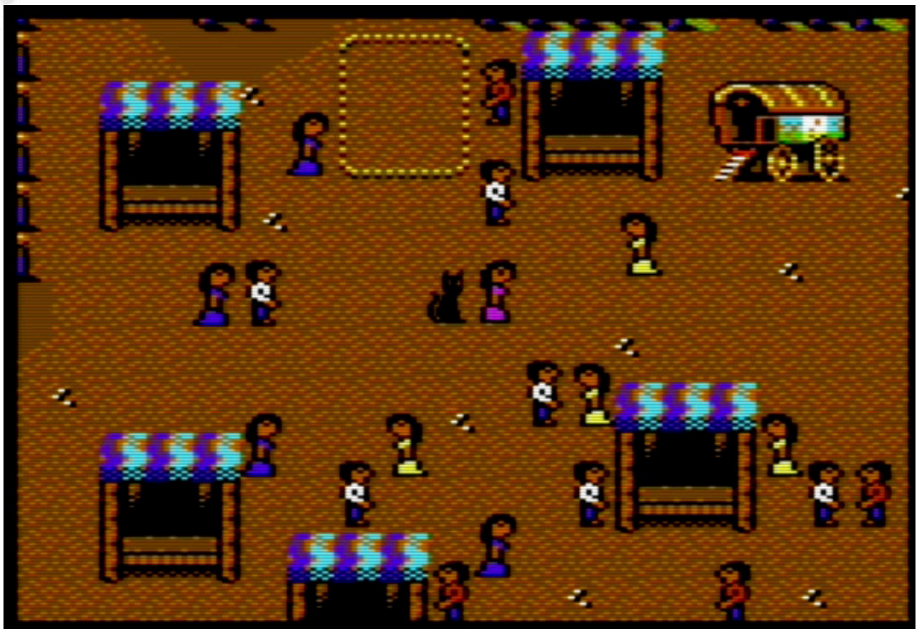
As in any self-respecting title in the genre, there are classic aspects such as equipment and magic shops and the inevitable side quests.
The title engine is the same as the previous one but with some improvements such as the second button to open the menu (also





```
MP. ELIXIR
WATER FLASK: 1/1
MINT LEAVES: 3/3
CAYENNE: 0/2
THYME: 0/2
```

```
 MINT LEAVES x13
 PLANTAIN x6
 ATTERLOTHE x6
➧CHERVIL x6
 FENNEL x6
 CATMINT x11
 HONEY x9
 WATER FLASK x5
 BELTED SEAX
 HI. POTION
 PHNX. DOWN
```



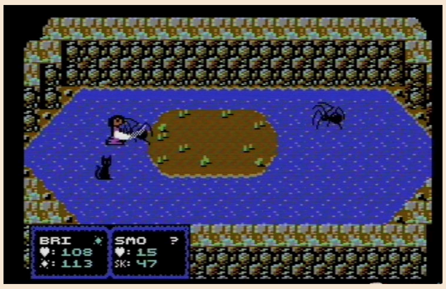
implemented in the first chapter in an update).
The graphical aspect is always pleasant and well animated, as is the soundtrack, which is well suited to the game context.
I forgot... it is Pal and NTSC compatible and runs on all possible types of configuration for Commodore 64 without a hitch.
Compared to the first chapter, the difficulty level is slightly higher. This is also due to how our beloved little witch starts at the beginning of the game.
It is all proportional to the initial power level.
A title in crt (cartridge) format that does not deviate much from its predecessor but offers a few hours of entertainment and adventure.
A really well-made title that makes you realise how skilled (for those who still don't remember) the valiant Sarah Jane Avory is behind a keyboard.
A must buy!

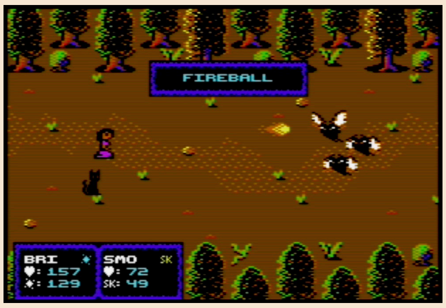
by **Giampaolo Moraschi**

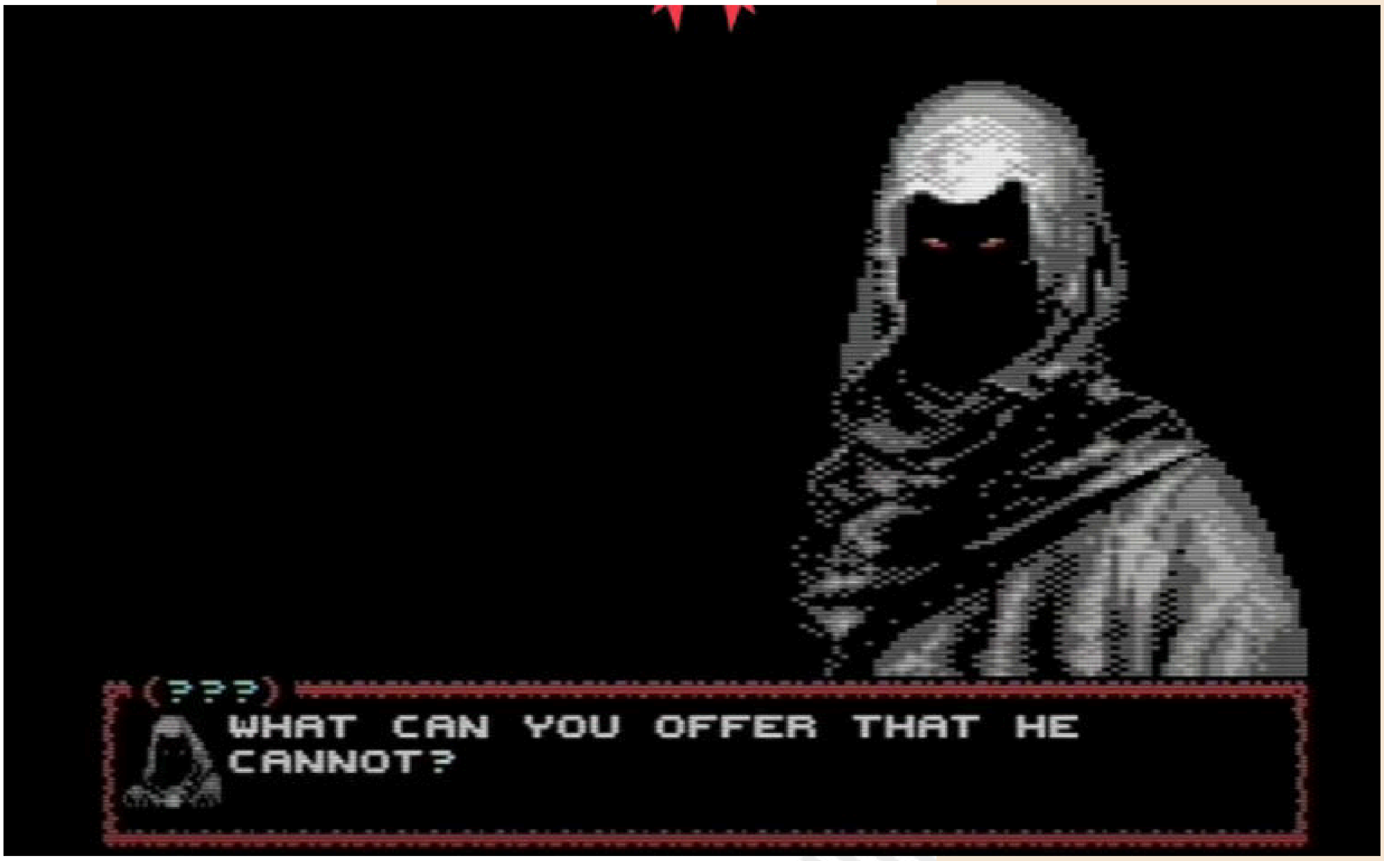
OUR FINAL SCORE

» Gameplay 91%
Same gameplay as the first chapter and a decidedly intriguing story.

» Longevity 90%
More difficult than the first episode but with a balanced degree of challenge.







NEW GAME

YETI MOUNTAIN

Year: 2024
Editor/Developer: Protovision
Genre: Action Multi-Genere
Platform: Commodore 64
Website: https://protovision.itch.io/yeti-mountain

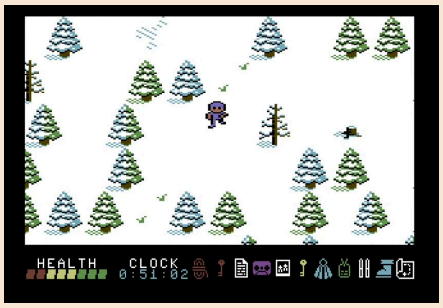
Yeti Mountain is a multi-genre adventure and exploration game, like so many titles from the golden age, developed by a talented team led by Russel Mills.
In three separate episodes we have to investigate the disappearance of our best friend and other skiers.
The first episode is a mission-based RPG style. Lots of character interaction, participation in skiing events and exploration.
The ski village created is very interesting and is developed with a lot of attention to detail.

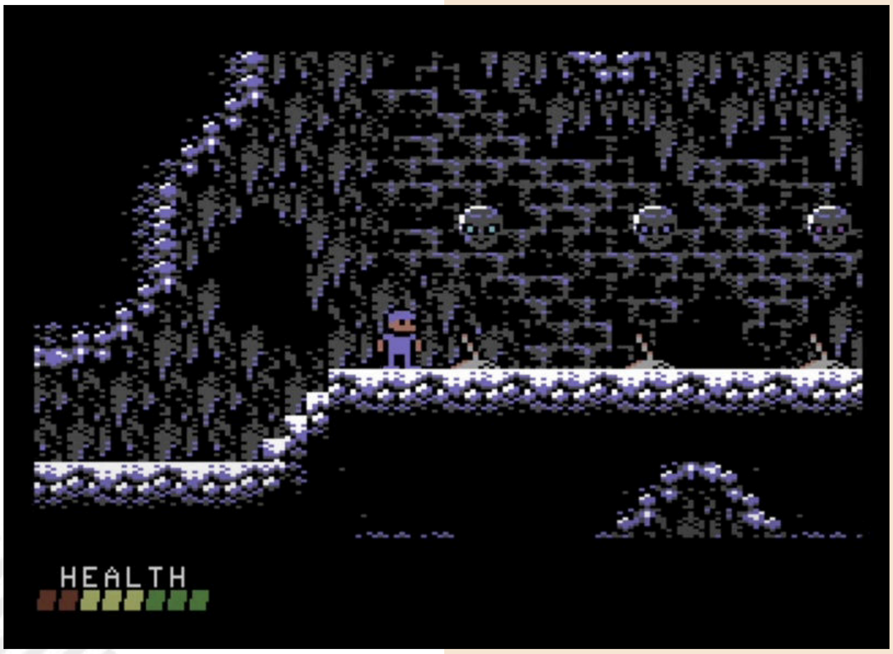
In the second episode, we will be engaged in an action/platform game with multiple scrolling. We will enter the Yeti's lair and explore different environments. Traps and hostile fauna galore!

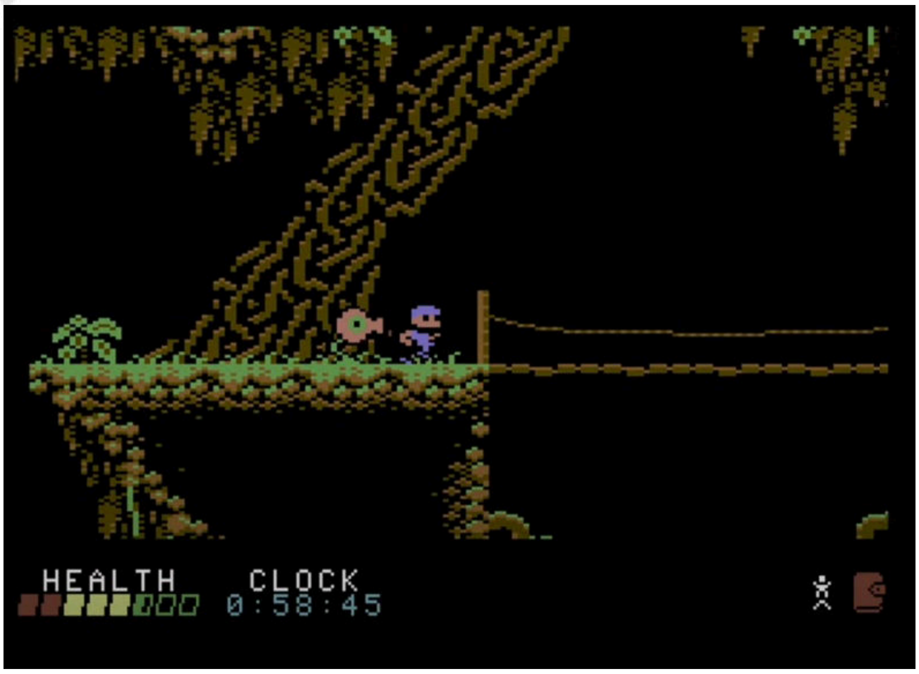
The final episode, on the other hand, will take us down the ski slopes to the foot of the Yeti's mountain. During the descent we will be involved in various puzzles and explorations.
There is a bit of everything in this title: mini-games on skis, platforms, exploration, puzzles, monsters, traps, etc. etc.

The narration reminds me of some Lovacraftian stories. It is very well written and flows smoothly and enjoyably through numerous well-crafted interlude scenes.

The file is in CRT format and includes an in-game save option. It is PAL and NTSC compatible and costs about $9 in the store.
It is money well spent because Yeti Mountaian is a fun title. I enjoyed the narrative and the style of gameplay. I don't really like multi-genres, but there is quality here.
Quality in the story, quality in the







OUR FINAL SCORE

» Gameplay 90%
Three different approach scenarios and lots of gameplay in one title.
Beautiful exploration sections.

» Longevity 90%
It is a challenging but never nerve-wracking title. The narration is really excellent.

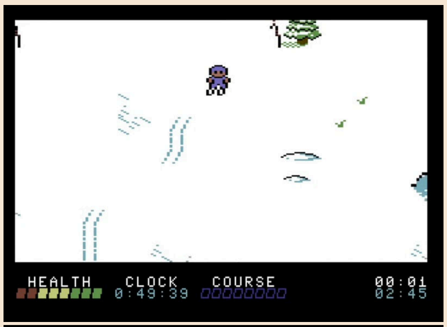
graphics with good use of colours and animations, and a lot of quality in the soundtrack with 21 beautifully crafted tracks.

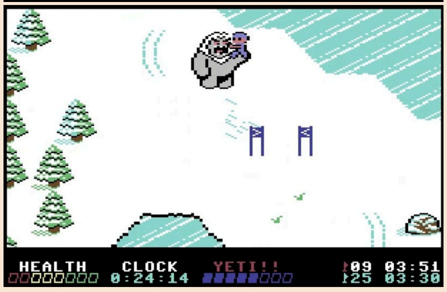
The side quests keep the interest high throughout the game, another plus point.

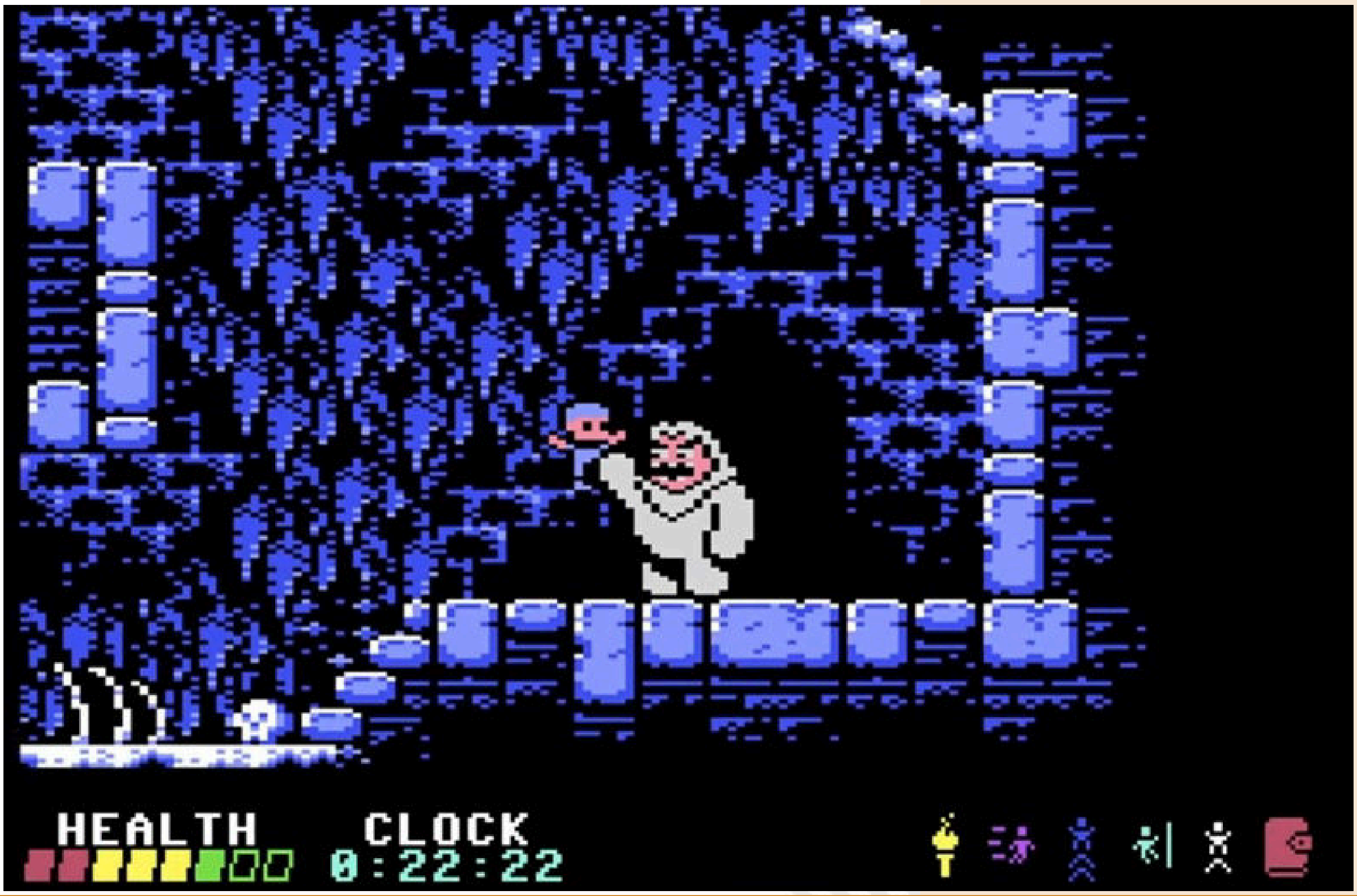
There is variety in the style of play. For example, skiing downhill, accumulating points through complicated jumps and stunts, or actively participating in competitions and events.

In short, this is a fine product that deserves to be purchased in digital format and, when it comes out, also in cartridge format.

by **Carlo Nithaiah Del Mar Pirazzini**







NEW GAME

CORESCAPE

Year: 2024
Developer: drMortalWombat
Genre: Shoot em up
Platform: Commodore 64
Website: https://drmortalwombat.itch.io/corescape

Warning: this game is not easy, it does not make you happy. It's challenging but it's damn frustrating. You can't tackle it without knowing every single wave by heart. You need quick reflexes, manual dexterity... and the use of some trainers specially included in the title.

Corescape is a shooter for hardcore gamers of the first guard. Get used to space battles with immense difficulty. Inserting tokens upon tokens into the cabinets in order to see the next levels.

It's so damn difficult... to be beautiful!

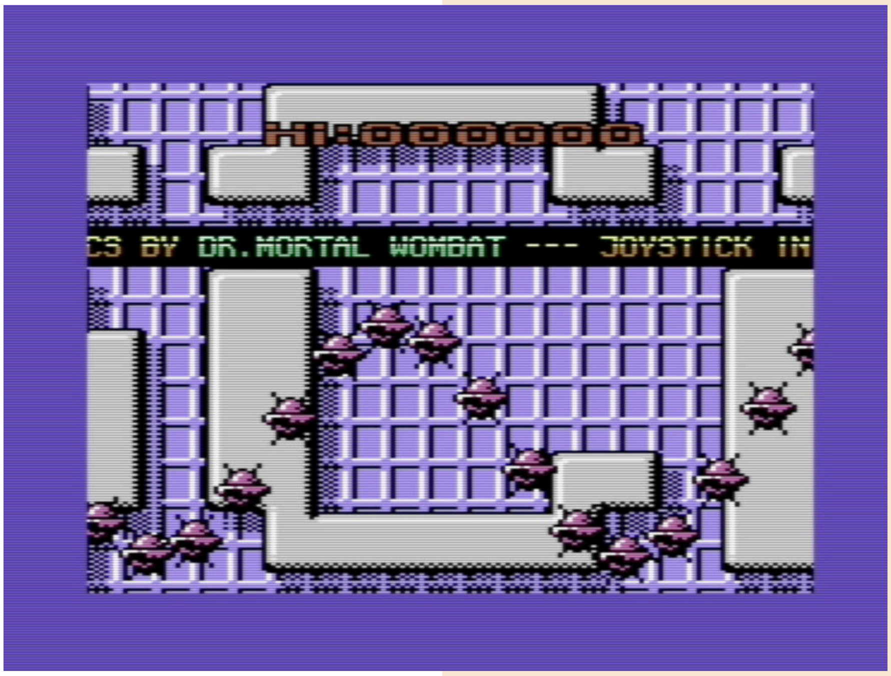
It is not the prettiest but its level design intrigues and makes you play and replay it often.
The real problem with this game is not the enemies, but the barriers and background elements.

They are indestructible and require a strong visual memory and great cold blood to overcome.
You cannot imagine the expletives I released every time I died badly against a level element.

There are eight levels and they are all a great feat to complete.

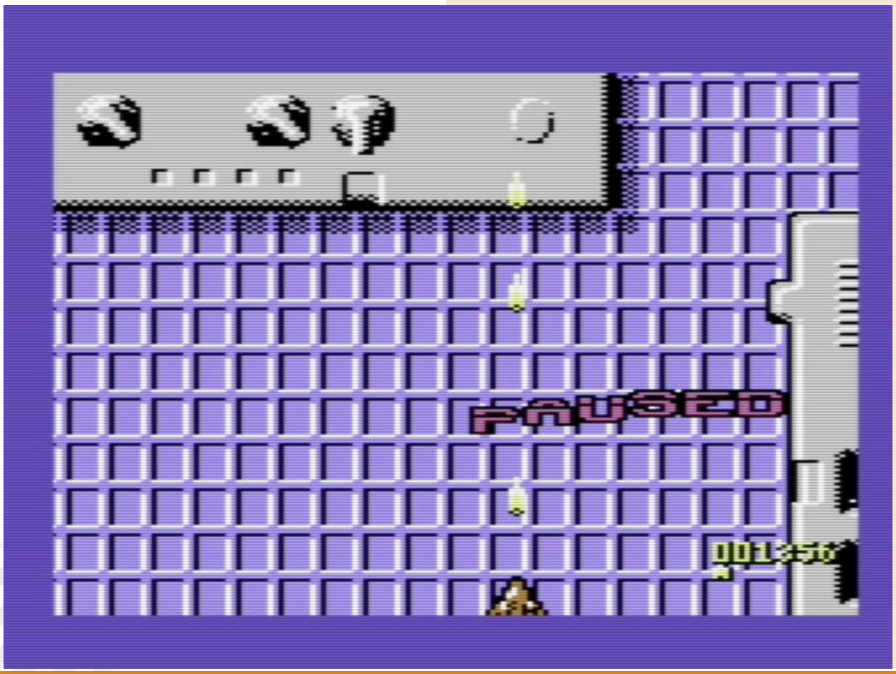
If you like 'almost' impossible challenges Corescape is the perfect game for you.

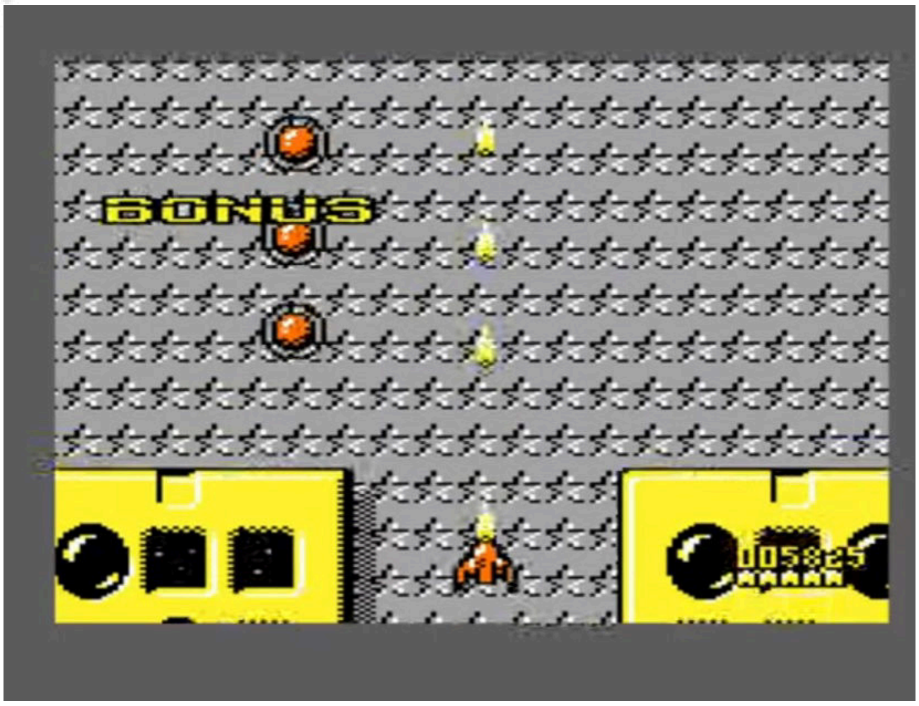
I appreciated not only the brutal style



of the game design, but also the technical aspect.

Fast, well animated with a 'metallic' graphic look that reminded me of some classics of the past.





The levels are not varied, but they are beautiful to look at and well-kept.

Sid sounds great, pumping out a punchy and catchy soundtrack peppered with good sound effects.

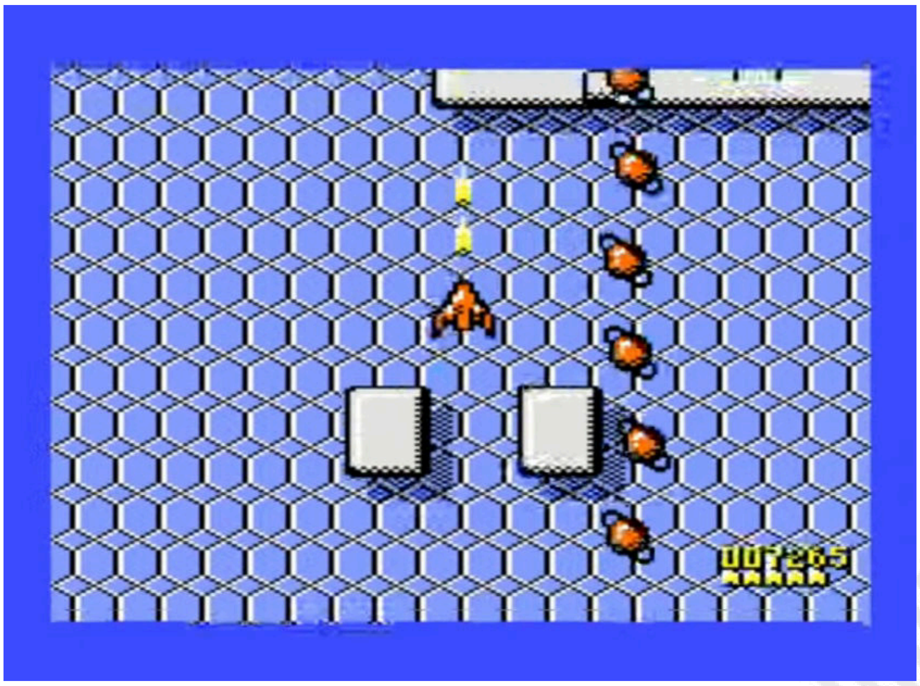
The file is in prg format compatible on all possible systems and is designed and tested PAL, but runs slightly faster on Commodore NTSC.

It shoots a lot, there are no power-ups and it is full of things on the screen. It never slows down.

A clear example of good work.

I repeat the initial warning: this is not a game for everyone. It takes guts and courage to complete it. Approved!

by **Giampaolo Moraschi**



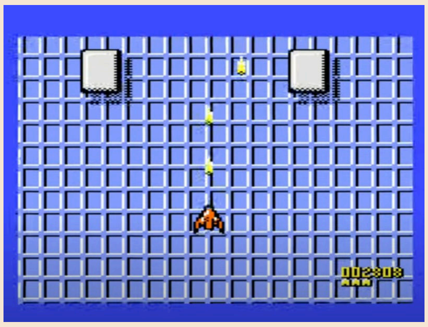
OUR FINAL SCORE

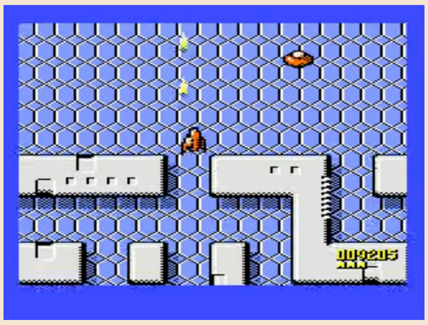
» Gameplay 80%

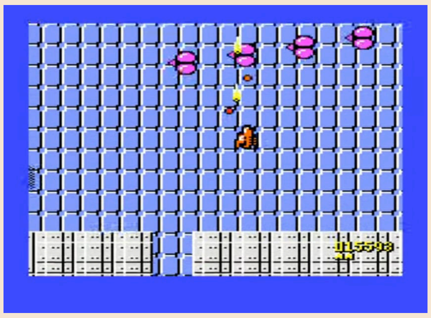
The level structure is intriguing and the control system perfect.

» Longevity 65%

Personally I would have given it 90%, but I have to be fair. It is a really difficult game and suitable for veterans of impossible challenges.







MAGICAL POP'N

Year: 1995
Editor/Developer: Pack-in-video/Polestar
Genre: Platform/Metroidvania
Platform: Super Famicom/Super Nintendo

I love scouring the import libraries for old consoles, I am constantly amazed by the sheer number of hidden gems most people have never heard of. Licensed Gundam titles, a thousand publications of never-localised role-playing games, bizarre sports games. But it is the publications of smaller publishers that are interesting.

Pack-In-Video is certainly not a famous name that I associate with high-end software. So it is ironic that they published one of the best games for the 16-bit Nintendo of all time.

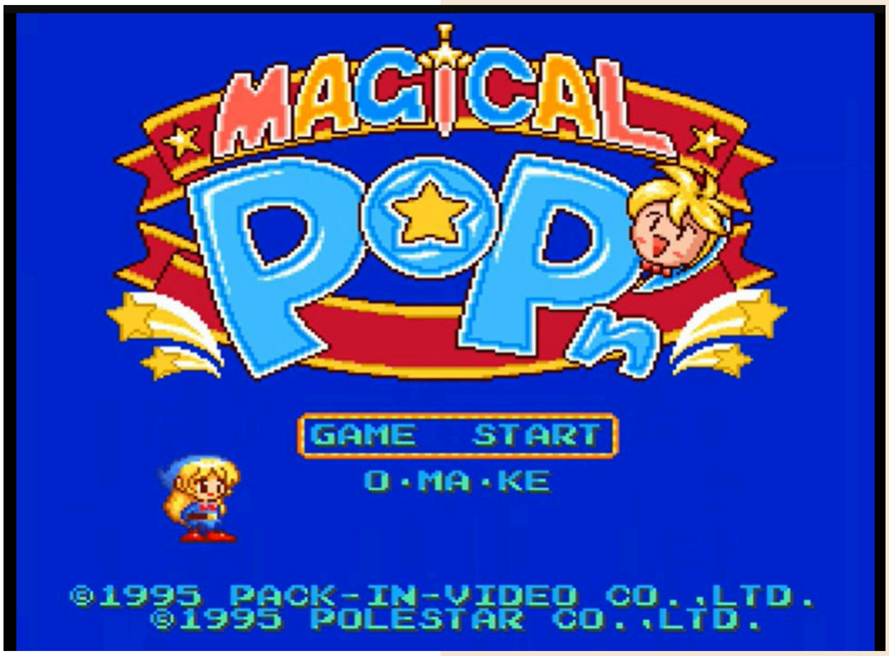
Magical Pop'n is a phenomenal title that deserves to be in the same cabinet as Super Mario World or Metroid and is a must-have title.

The best way to describe Magical Pop'n is to associate it with a strange combination of Mario, Metroid and Zelda, in short a Metroidvania.

It is a legendary mix and the game manages to extract the right elements from each to create something unique.

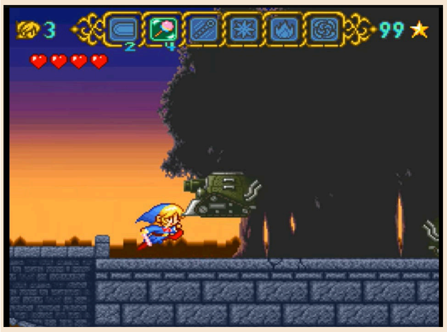
The protagonist, a cuddly, chubby princess, is an extremely agile character who moves at a fast pace. She is armed with a sword that she can use in almost any direction.

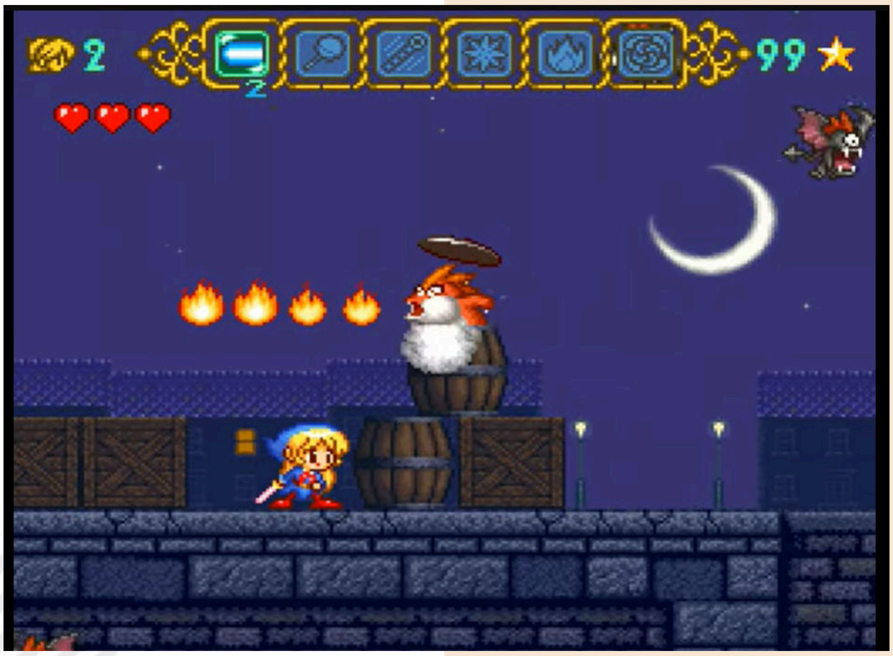
As in Zelda, here too we have an inventory of secondary weapons and spells to use. As in any good action game, you start with a simple weapon but can upgrade everything to the max: attacks, bombs, climbing, Sonic-style spin attacks... In short, there is a lot of variety.

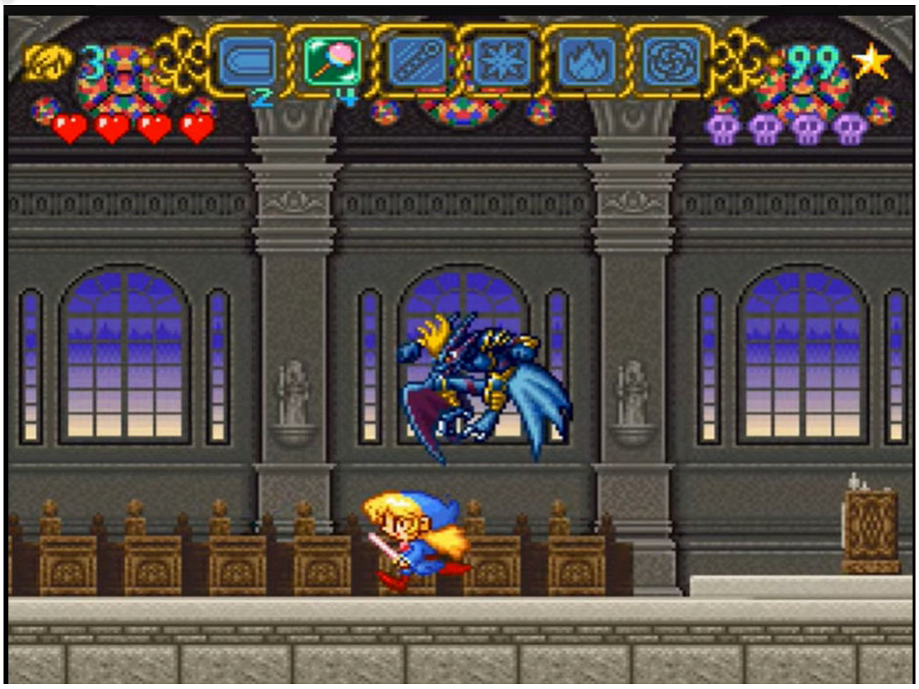


The Metroid connection is firmly rooted in the structure of the game. Although it is divided into six stages with a middle and final boss, each one is rather large with secrets and items to be found at any given time.

Each level contains a new spell or item needed to overcome blocks and/







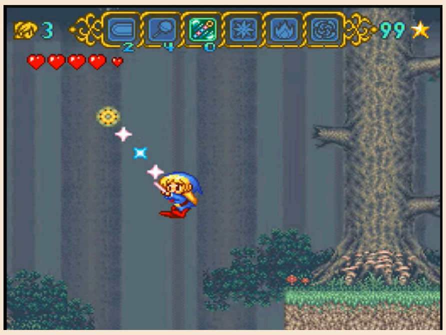
or further power up. Unlike in the Zelda series, these abilities are not crucial for defeating the boss of that particular level, but are useful for dealing with it more confidently. There are no time limits and the stages are large enough to provide a sense of 'open world', but not so large as to require a map.
It may sound strange, but it reminded me much more of a Sega Megadrive title than a classic SNES game. It moves fast. The pace of gameplay is excellent and the excellent controls make switching between skills a breeze in the midst of combat.
For those who want some replayability, there are usually multiple paths to the end of a given level.
All this leads to a spectacular finale that leaves you speechless.
The six game stages are of moderate length and I personally appreciate this. There are no passwords or battery-saving features because they are unnecessary.
Each stage is perfectly calibrated with what you have to do and deal with.
To be honest, the only real flaw is its lack of distribution in the West. The title was only launched for the Japanese market.
The language does not affect the playability in any way.
A phenomenal title.

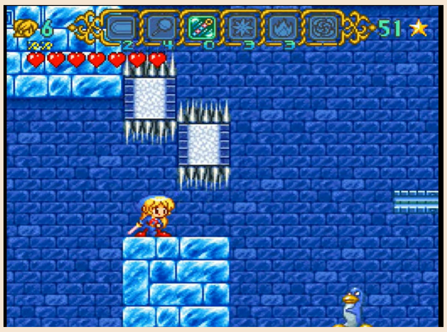
by **Roberto Del Mar Pirazzini**

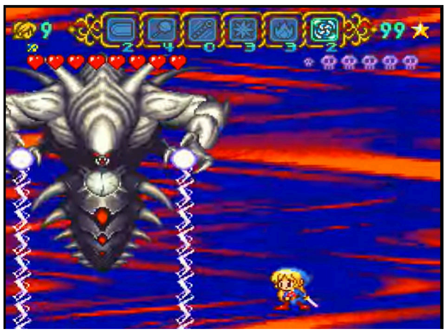
OUR FINAL SCORE

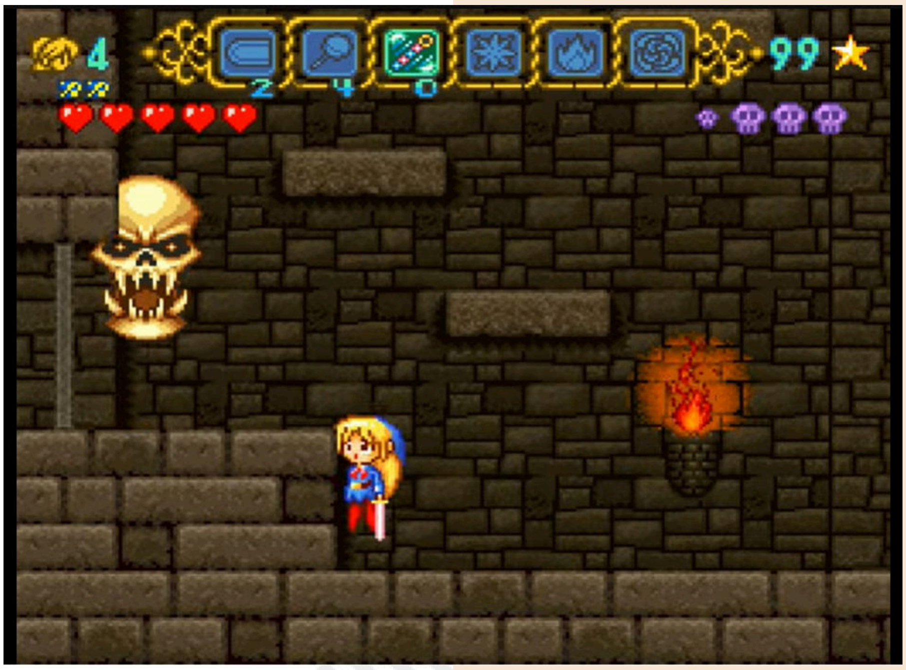
» Gameplay 95%
Varied, elegant, graphically beautiful.

» Longevity 90%
Long enough to be best appreciated.









THE ADVENTURE OF LITTLE RALPH

Year: 1999/2007
Editor/Developer: Sony/New Corporation
Genre: Platform
Platform: Playstation (versione recensita), Playstation 3 e PSP

In the late 1990s, the 2D platformer was a genre that was going out of fashion.

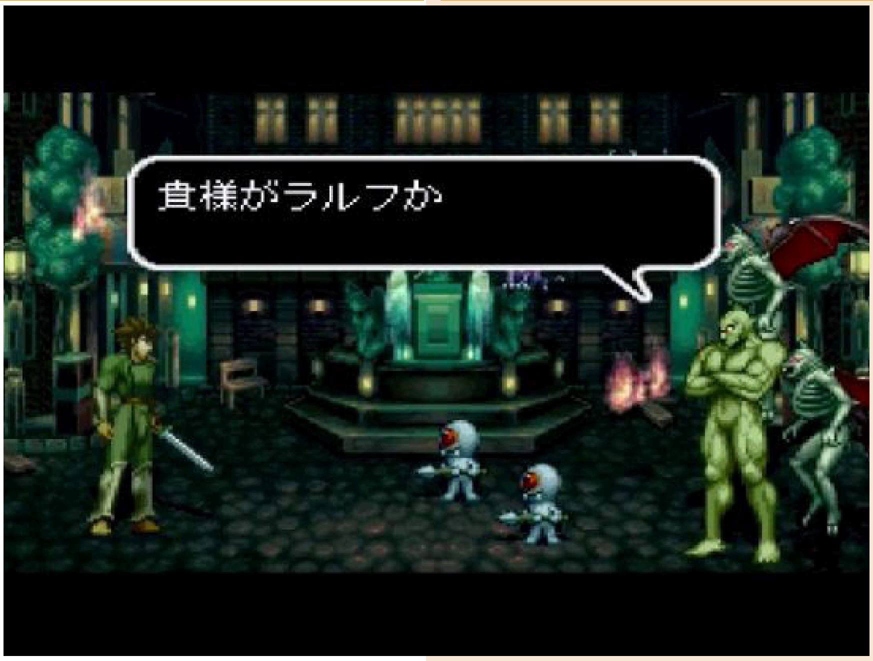
Super Mario 64 had made 3D mainstream and it didn't help that Sony preferred titles that pushed 3D games over titles that still relied on sprites. Often, some games were denied western localisation if they were based on 2D sprites.

The Adventure of Little Ralph is one of those titles that the West missed out on.

Ralph has a problem. His girlfriend is super hot and has been kidnapped by the evil Valgo; a crazy, horny villain. As if that were not enough, this giant yellow Adonis has also turned our hero into a child. Not only does Ralph have to face his evil rival in childlike form, but he has to prove that he is a pretty tough guy by surviving eight hellish levels that have various branch points.

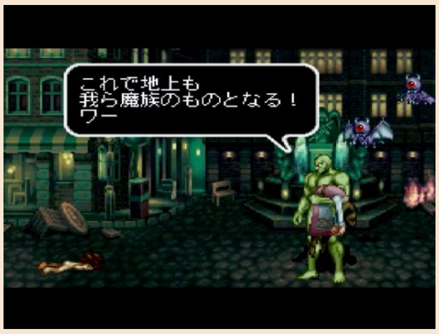
Sword in hand, Ralph will fight his way through robots, devils, satanic pigs and polar bears...

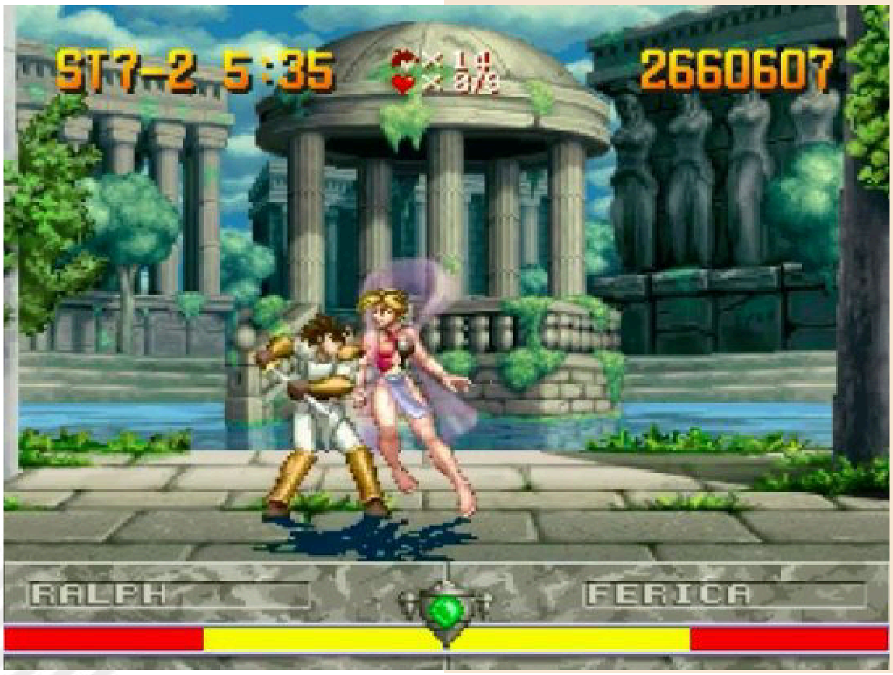
In a series of colourful and whimsical game worlds, our hero will find himself engaged in the most classic of platform games, but with a refined 2D twist.

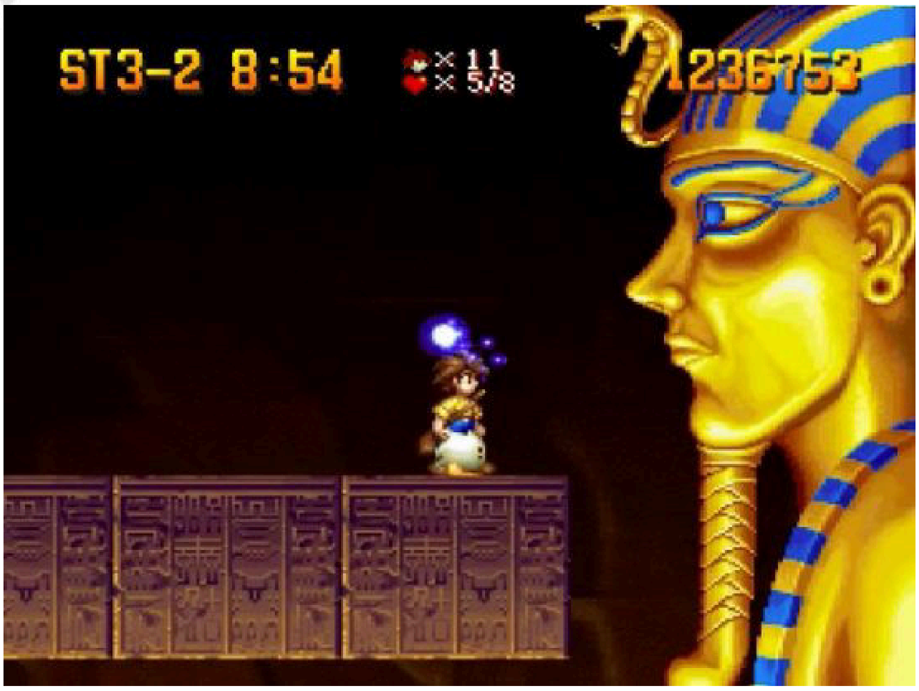


A title that went unnoticed in the West in 1999 but retains a wonderful technical charm.

The guys behind The Adventure of Little Ralph paid a lot of attention to pixel art and animation. The characters







are detailed, the backgrounds inspired by a style that reminded me of the late Shotaro Ishinomori, and there is a masterful use of colours.

The aesthetics really come into their own with a masterful look.

In 2007 it was brought back to PS3 and PSP but went unnoticed once again.

At this point, I can only recommend it to you in this 2024.

If you can find it in a physical version at an acceptable cost, don't miss it.

Truly a gem of graphics, sound and gameplay.

by **Marta Rossman**

OUR FINAL SCORE

» Gameplay 90%

There's everything a good platform game needs. Responsive controls and plenty to see, do and play.

» Longevity 85%

Tough title with some challenges at the limit of human endurance, but it glues you to the pad.

NEW GAME

HUNTER GIRLS

Year: 2023
Editor/Developer: PSCD Games
Genre: Running Game
Platform: Sega Megadrive
Website: https://pscdgames.com/index.php?route=product/product&product_id=15

Girl power!!! Three women with special abilities throw themselves into adventure in a next-generation running game for Sega Megadrive.
A fantasy adventure with lots of platforms to jump, skills to use and monsters to kill and avoid.

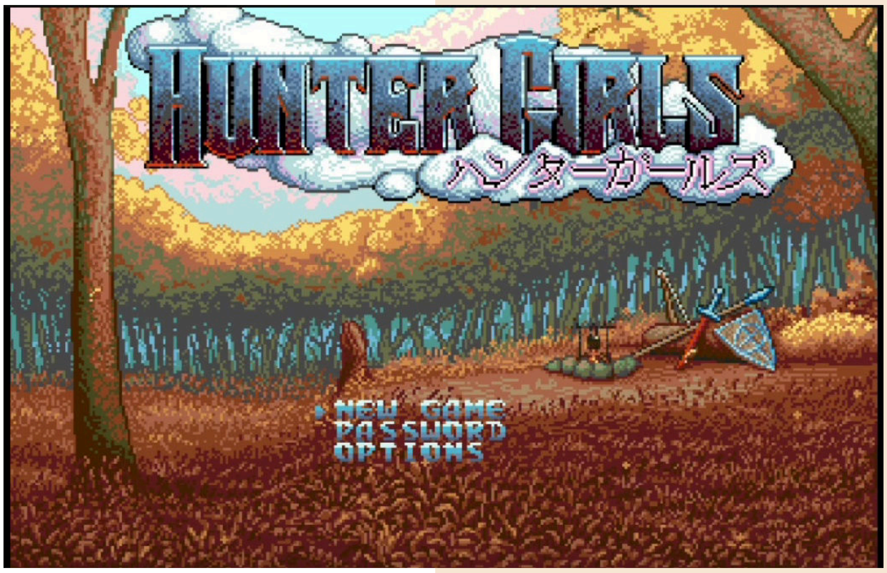
Yeeeeeh!!! Ok, no... actually Hunter Girls is a title that created so much expectation for me and in the end it turned out to be rather dull and technically mediocre.

Yet the PSCD team is a good group of developers, but this time it left me with a bitter taste in my mouth.
Dull in story and dull in mechanics.

A flat running game that we have already seen in all sauces.
Sure, the introductory films are nice, as is the accompanying music. I can admit that even the menus and the ability to set many languages are welcome and well-done things, but it is a flat game with a high degree of difficulty due to less than perfect handling of jumps and collisions.
Agnes, Kim and Flora all have different abilities. Some are better at defence and some are more skilled at jumping.

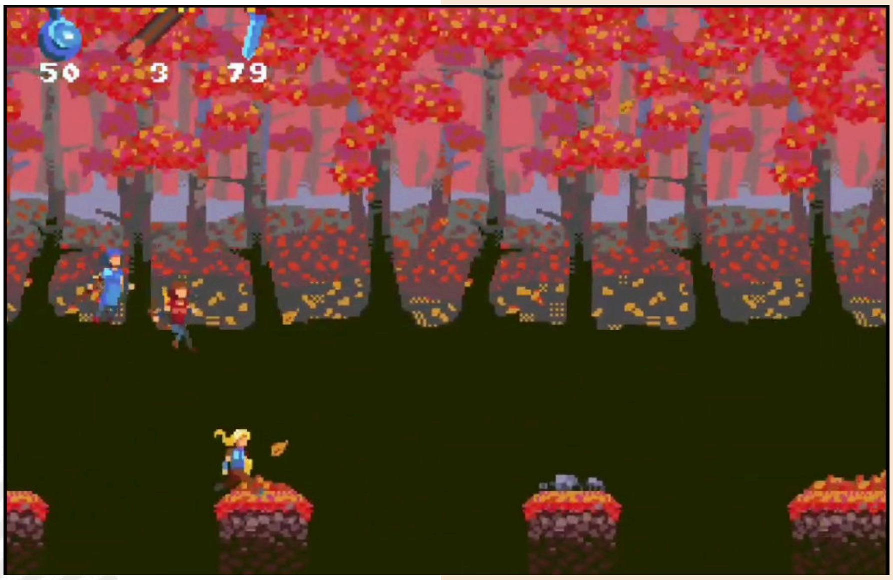
At the top, during the game, we have three indicators: the first shows us the mana that one of the girls possesses at the beginning, the second the number of bullets in another girl's bow, and the third the defence capacity.

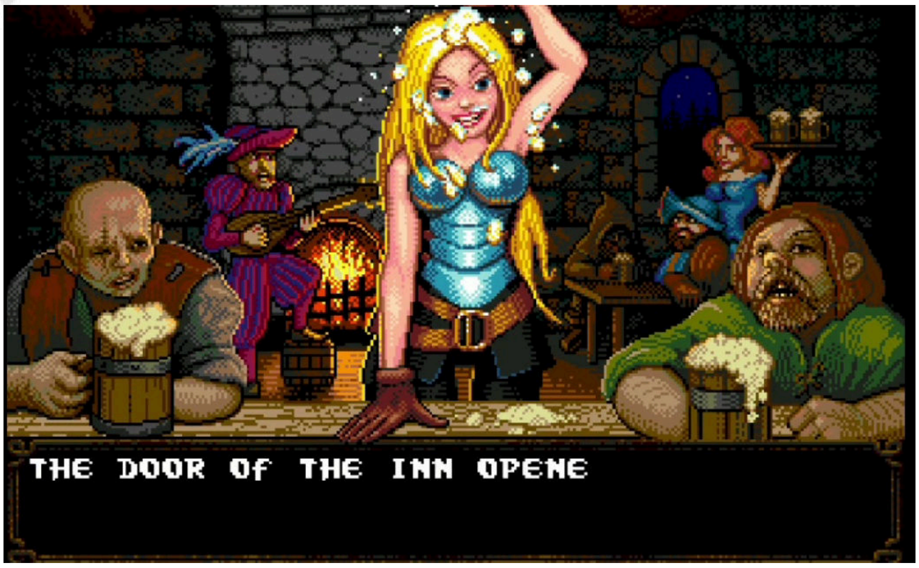
All of these skills are exhausted when



they are used, so it is necessary to do so sparingly.

The title actually lasts about thirty minutes. It is completed with only one condition: that of memorising each route perfectly.
Any mistake in positioning, jumping or misuse of skills leads to certain death. Definitely too limiting and frustrating.





Add to this a less than excellent technical compartment. The sprites animate well but lack detail and above all the game action is 'sparse'. There is little to see. One jumps over goblins, parries arrows, takes time to zap over platforms to avoid cliffs.
Simple backgrounds with very bland autumn colour themes.

In short, I did not enjoy myself and would have preferred to play some other PSCD title.

The game is available for purchase in 'cartridge' format for $55 on the site I linked in the description.

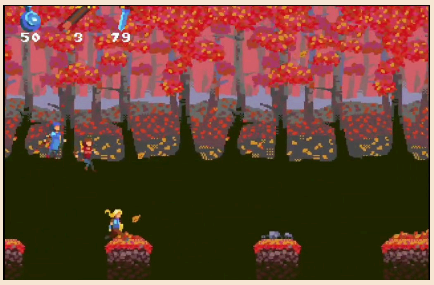
Fifty-five dollars is definitely a lot for so little substance.

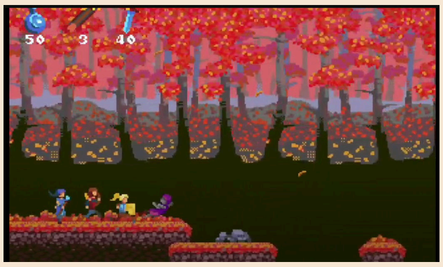
by **Marta Rossmann**

OUR FINAL SCORE

» Gameplay 50%
A repetitive running game with some design problems.

» Longevity 30%
Playable in thirty minutes after madly memorising every single jump.
Definitely insane!





"YES, I'M AGNES AND I'M ON MY WAY TO RESCUE PRINCE ALEX."
"I HEARD ABOUT HIS ABDUCTION. IT WAS KELTREM, THE MONSTER THAT KILLED MY

NEW GAME

SERAPHIMA

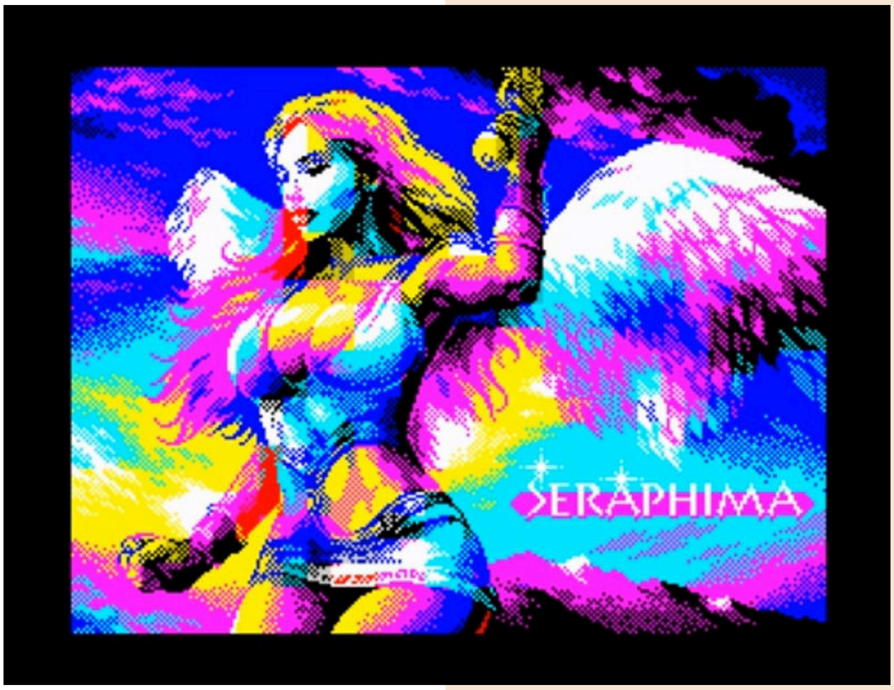
Year: 2023
Editor/Developer: Zosya Entertainment
Genre: Metroidvania
Platform: ZX Spectrum 128k
Website: https://www.zosya.net/2023/11/03/new-game-seraphima/

Is this the best Spectrum game ever? I honestly don't know.
What I do know is that Seraphima is one of the best homebrew for this machine in the last 10 years and the best released in 2023.

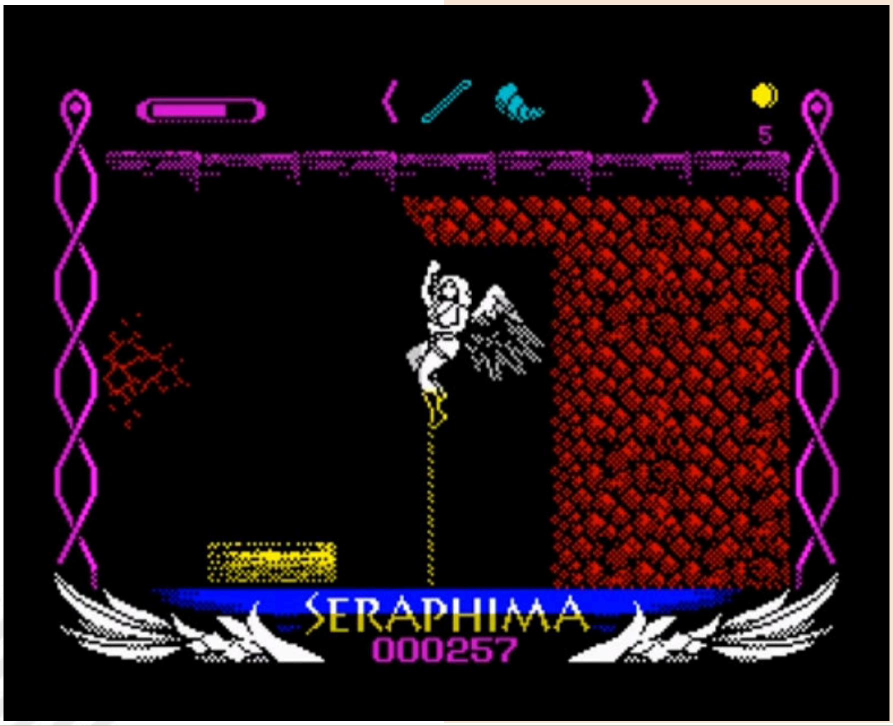
An over-the-top title, well-developed and enjoyable to play.
Guardians and marauders have coexisted secretly on Earth. The guardians (or Grigori) protect life, while the marauders hunt it down.
Their eternal struggle maintains the balance. However, when Seraphima emerges as the last guardian, the fate of the Earth becomes uncertain.

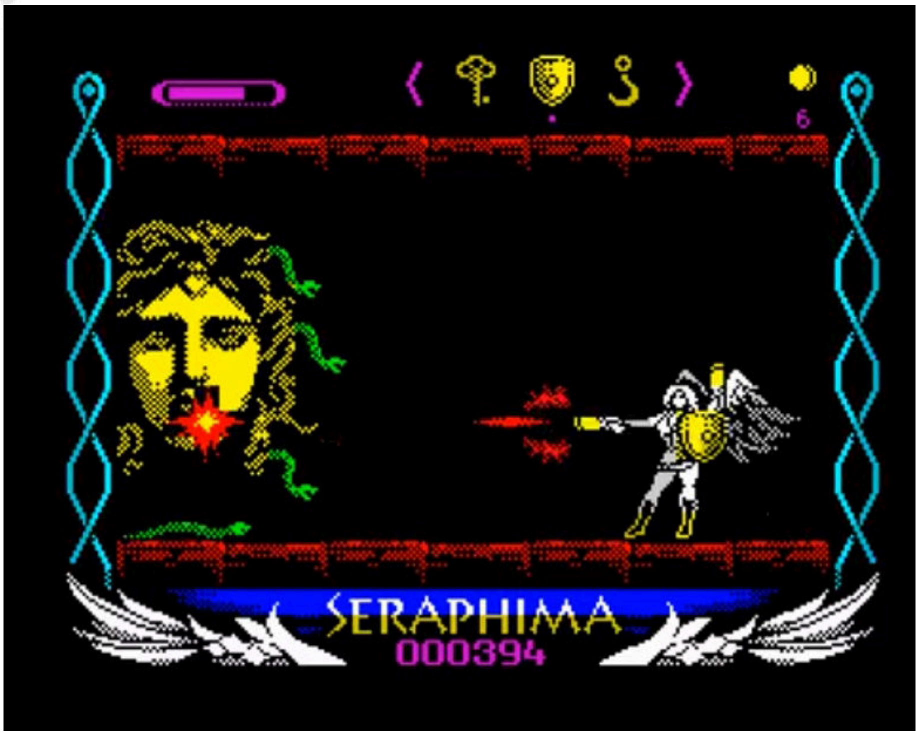
The Marauders are eliminating all guardians.
The task of our winged and prosperous heroine is to preserve the balance of the ages and fight bravely to ensure peace and salvation.
(Almost) predictable plot aside, Seraphima is a Metroidvania that entices us to explore and experience.
Our heroine initially moves through a kind of tutorial, guided by the Queen Mother. In this training camp she learns to shoot with her double magic pistols, to use objects and keys to open doors, and to clash with the warlike worlds she explores.

Each world is connected through doors and each world is different in terms of impact and difficulty.



A title that has it all. There are beautiful graphics. Animated well, coloured well and with the right touch. You can see the hand of an experienced team like Zosya's.
The sprite of our 'poppet' heroine is





OUR FINAL SCORE

» Gameplay 95%
Balanced, enjoyable, great tutorial and good controls (joystick recommended of course). There is some concept in the level development and a good story.

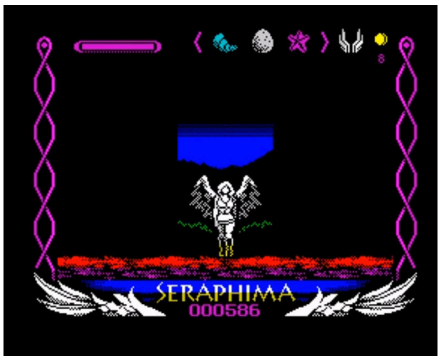
» Longevity 95%
It's a fun title, not too punishing but one that requires attention to be enjoyed to the fullest. It lasts as long as a title of this genre should and does not tire.

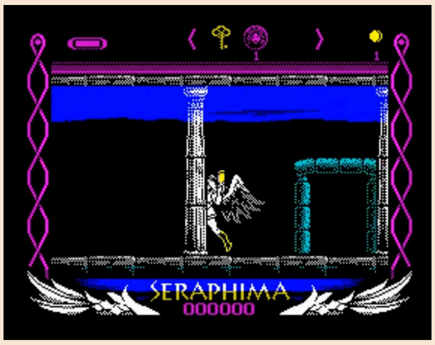
marvellous in its movements and elegance.
The same goes for the sound, music and effects compartment.
But the strength of the game is its incredible ability to keep you glued to the action.

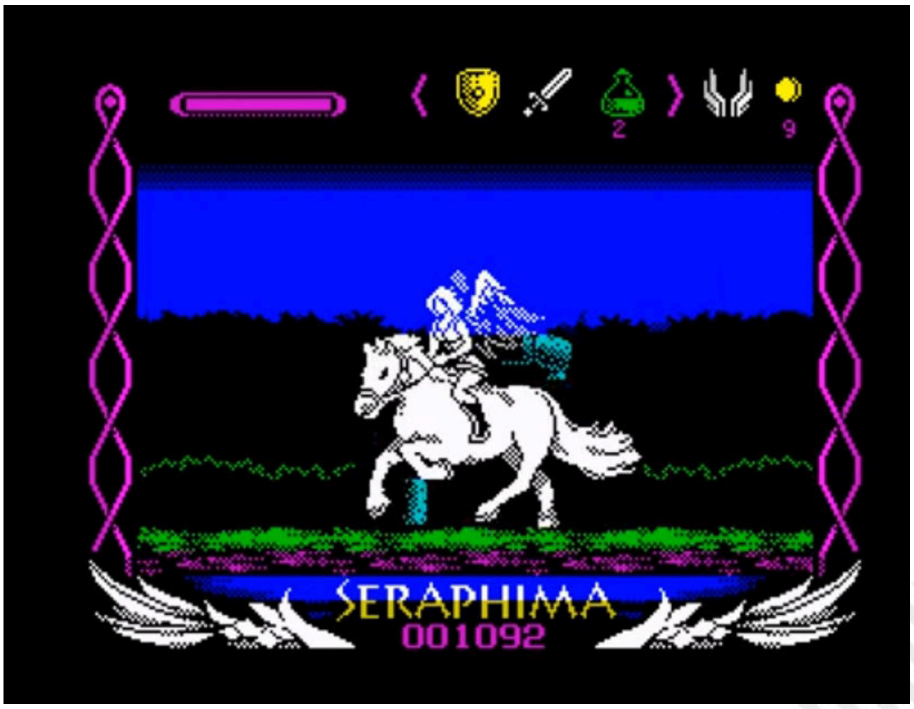
Seraphima is beautiful to play, to explore and to enjoy.
It is not very long but it is enjoyable.
A piece of work that deserves to be collected.

The title runs on all ZX Spectrum 128k and of course in emulation.
Masterpiece!

by **Carlo Nithaiah Del Mar Pirazzini**







NEW GAME

HERMANO

Year: 2023
Developer: PAT MORITA Team
Genre: Platform
Platform: Game Boy
Website: https://patmorita.itch.io/hermano-game-boy

During El Dia de los Muertos, little Mano went as usual to the grave of his brother Nano.

It has always been said that during this night, the gates of hell and heaven open so that the departed can be reunited with the living.

While his little brother was talking at Nano's grave, a terrible demon came out of nowhere and abducted him.

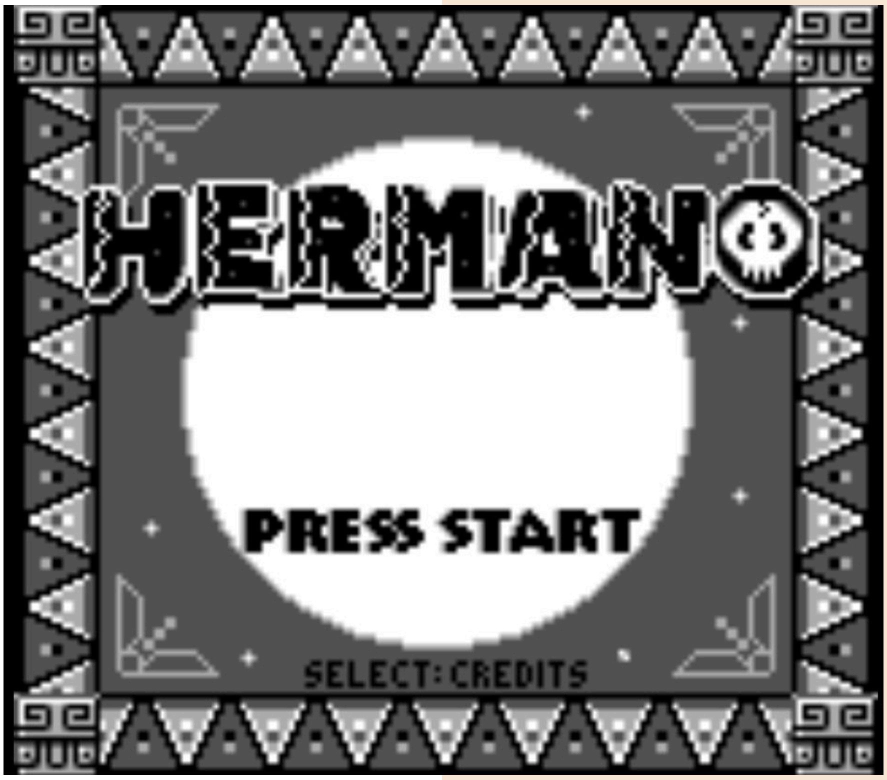
Just as Nano was returning to the surface.

This is the story behind Hermano, the new platform game for Game Boy that I just downloaded from the official website.

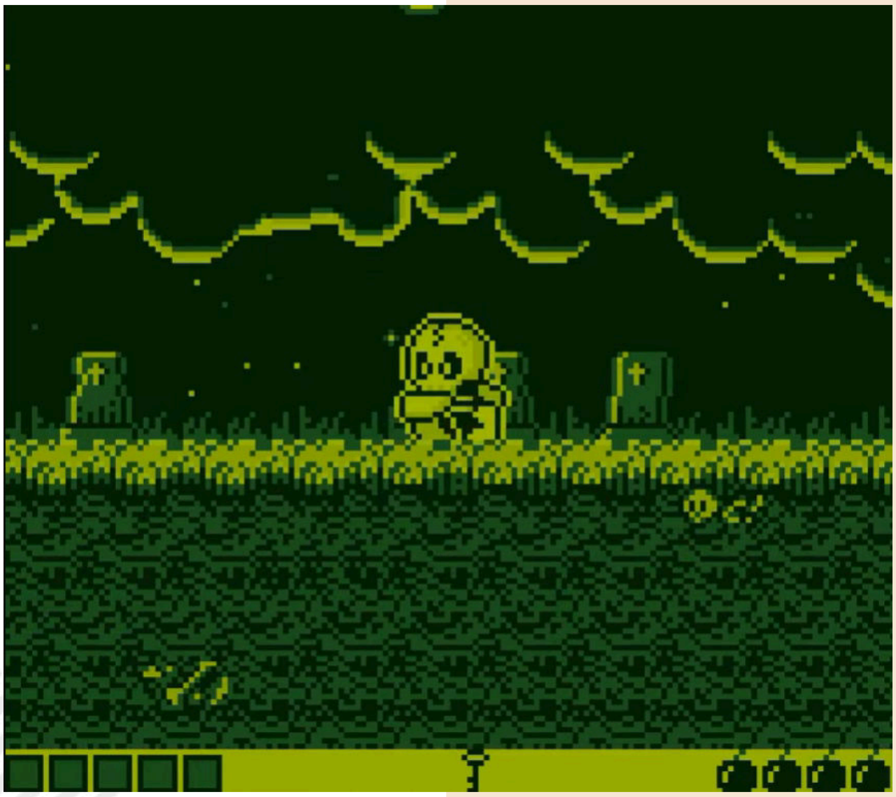
We will be at the helm of Nano. For a dead guy, he doesn't do too badly at all. He fights head-on against enemies and can rely on some bombs he finds around the levels.
It's a classic title that mixes Ghost n' Goblins elements with Super Mario and does it well.

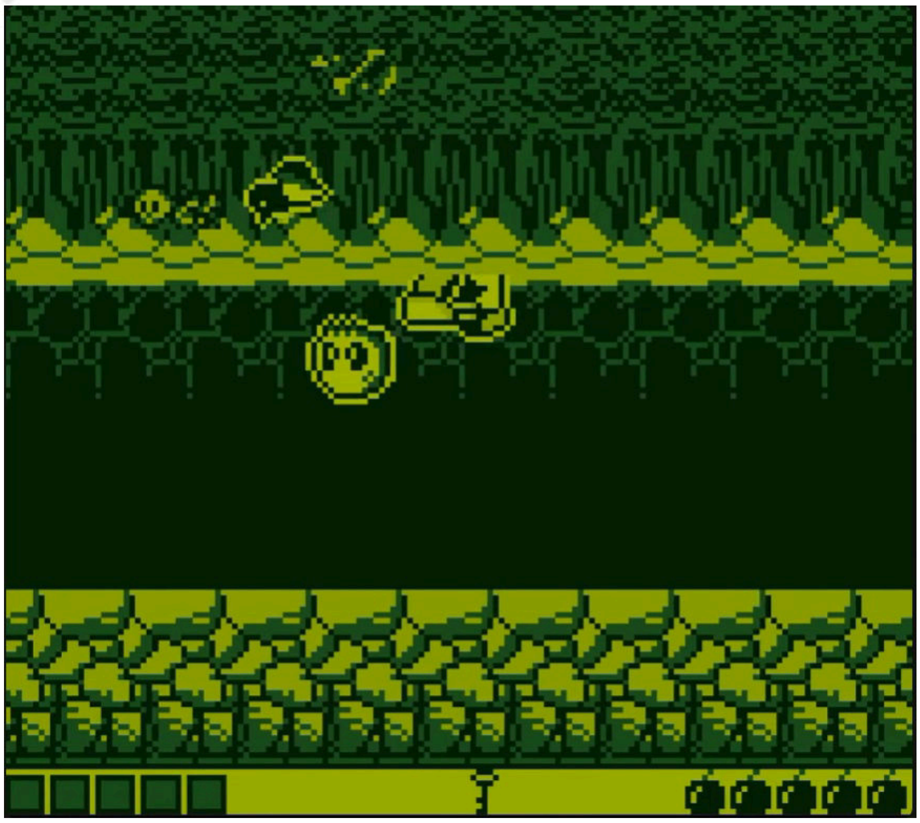
Pleasant to control and command and very pretty to look at.

I tried it both on my GB and in emulation and I must admit that it renders great in both cases.
The title also features some nice gameplay ideas such as the possibility of playing in 'inverted gravity' underground.
Several game levels with numerous



settings (cemetery, abandoned city, swamps, pyramids) and many lore





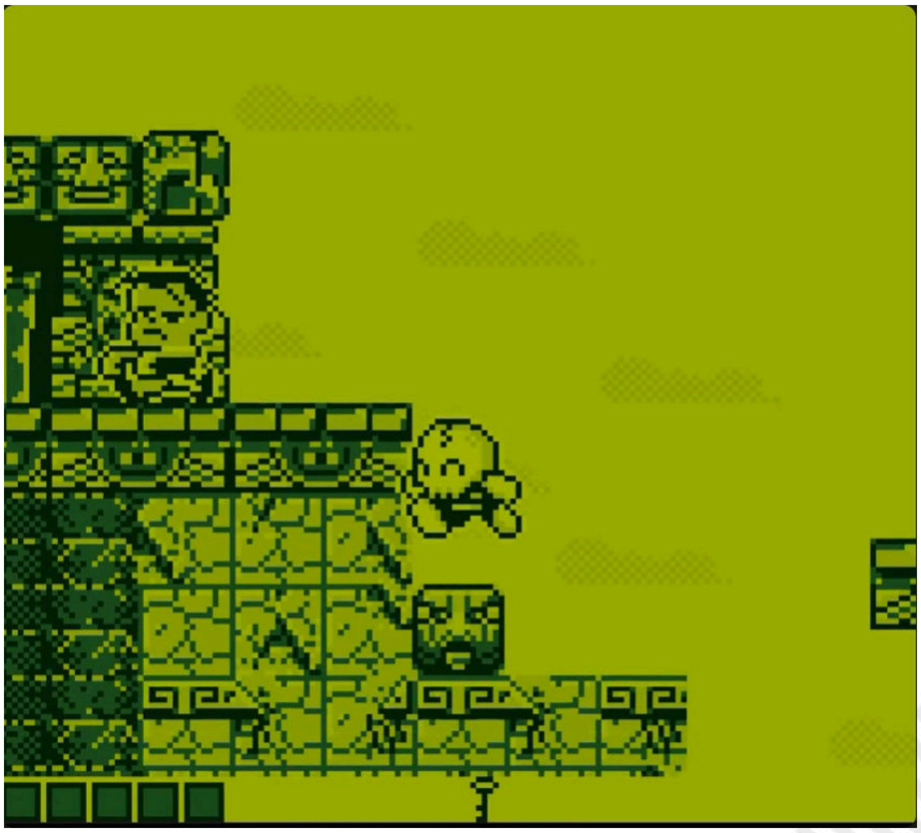
monsters to defeat.
The aim is to find the key to get to the next level.

I read on the website that there is also a physical version on cartridge that I will soon make mine.

Super clean and well animated graphics and really beautiful sound. I liked the music a lot.

Another Game Boy success.

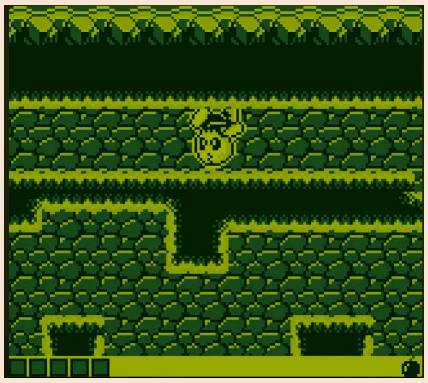
by **Marta Rossman**

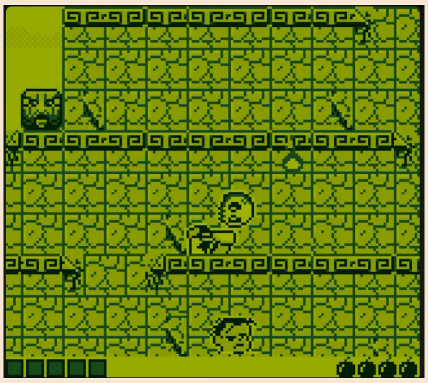


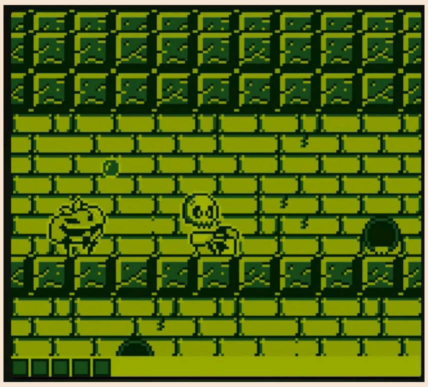
OUR FINAL SCORE

» Gameplay 85%
Fun and well structured. The idea of playing 'upside down' is intriguing.

» Longevity 85%
Several well-characterised levels that will take some time to play. Enjoyable.







NEW GAME

TETRIS 1200

Year: 2024
Editor/Developer: Gamtec, Super Fighter Team
Genre: Puzzle Game
Platform: Amiga 1200 o superiore
Website: https://jotd666.itch.io/tetris1200

Developer JOTD666 is getting us used to it! I would say very well.
He has been giving us conversions of complex titles from the past for a while now.

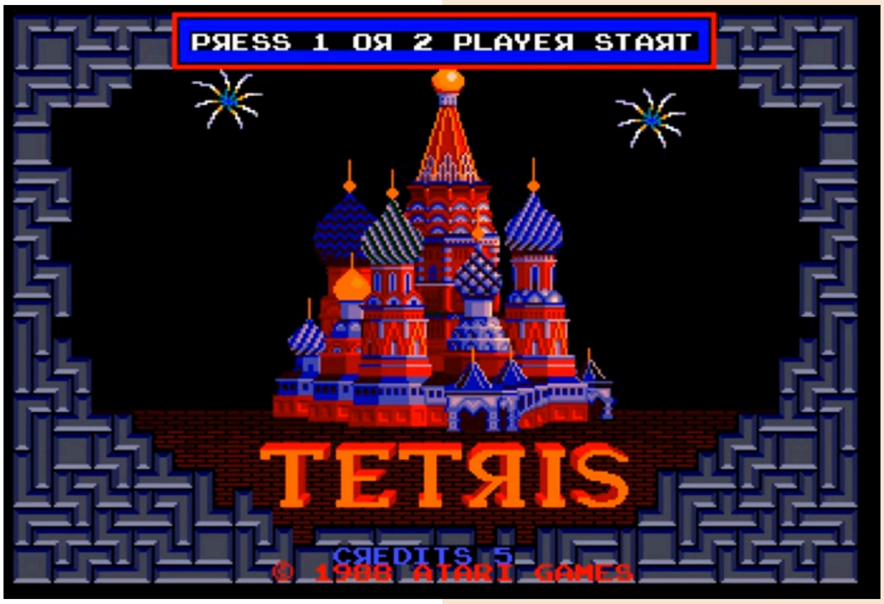
Arcade like Xevious or Super Bagman and more. All titles ported perfectly to our Amiga.
Even in the case of Tetris we are faced with a 1:1 porting of Aleksej Leonidovic Pazitnov's 1984 title.
In fact we are in front of the best known version for us fans, the Atari version that we found in the arcade in 1988.

A version that came about during a mega diatribe between Atari and Nintendo.
While the Nintendo one was more similar to the version made by Pazitnov on Electronica-60, the Atari one brought some changes.

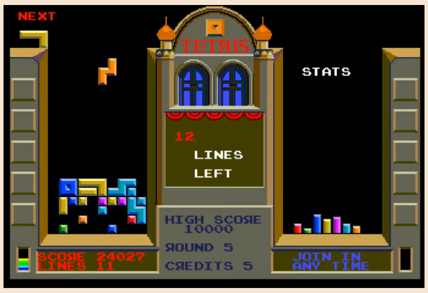
The game is based on rounds where the player advances by clearing the greatest number of target lines for that particular level. At the end of the round, the field is cleared and the next level begins with greater difficulty.

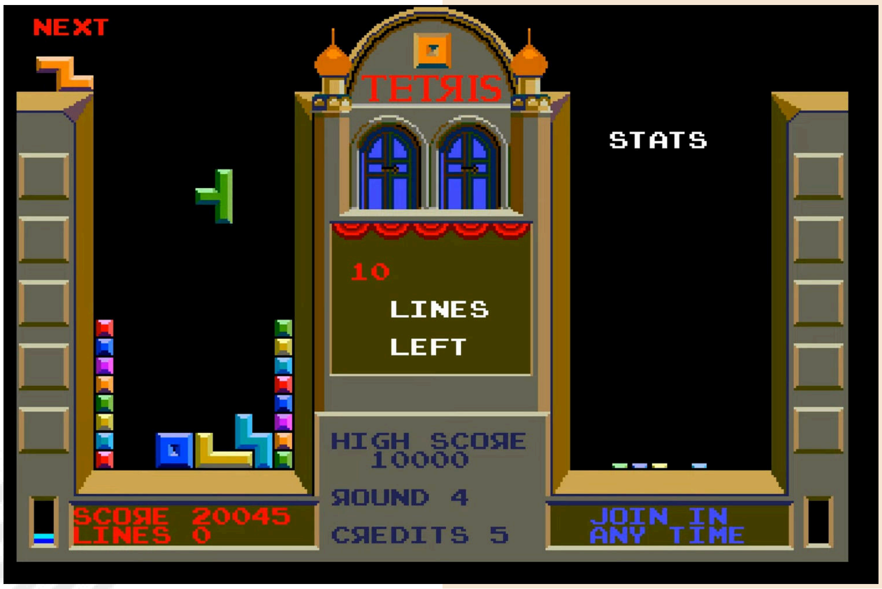
Next to the player's traditional score is a bar called the rainbow meter. Every four lines cleared adds 1 meter to the bar.

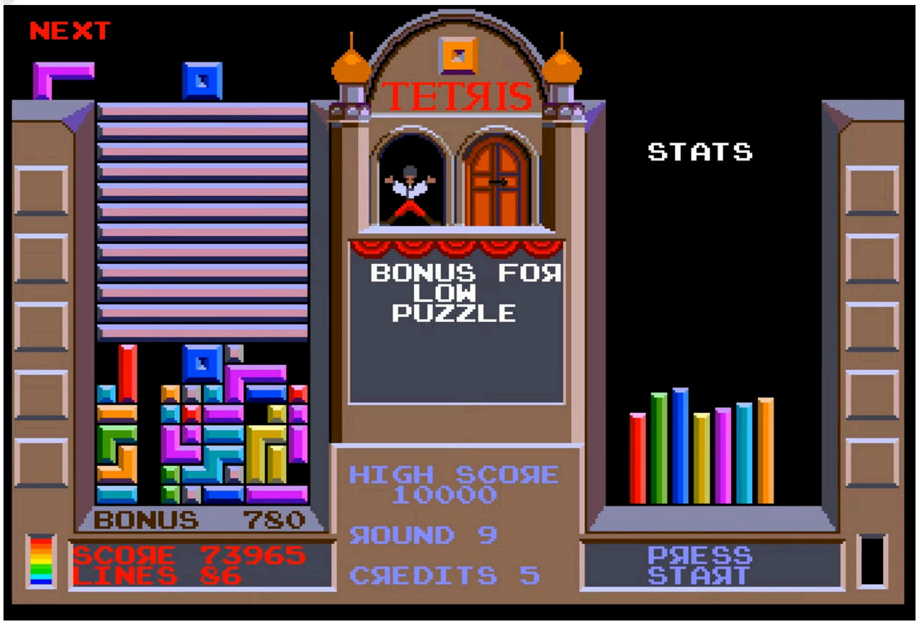
The score for each tetramino placed by the player is d *min(r*(r+h),250) where:
d = 1 for normal gravity or 2 for soft fall when the tetramino lands
r = the number of bars on the rainbow meter plus 1



h = the line on which the tetramino was placed, minus 1.
After finishing each round the game awards a "Bonus" of up to 2100 points based on the number of empty rows above the highest block. The formula is 5* e *(e + 1), where e is the number of empty rows, which corresponds to 10 points for the first row, 20 points for the second row, 30 points for the third row and so on, with 10 points more for each row.







OUR FINAL SCORE

» Gameplay 95%
I don't think much more needs to be said. Perfect in dynamics and gameplay.

» Longevity 99%
Infinite. Literally infinite.

Curiosity about the Atari title... it did not support sprites. The entire game ran in 'text' mode, just like the early versions released on computers did.

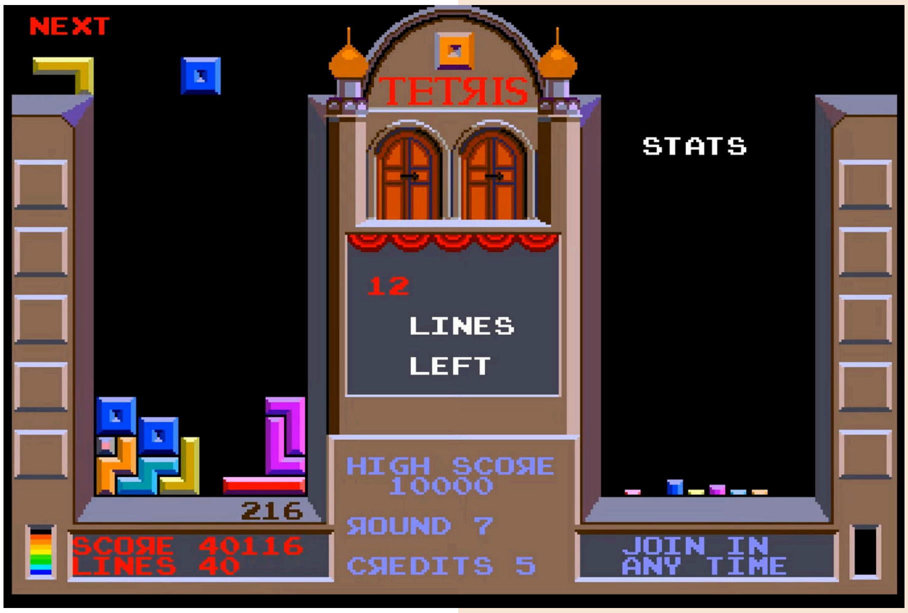
This Amiga version is a port of the 6502 to 68000 code and took quite some time and numerous betas to get perfect.

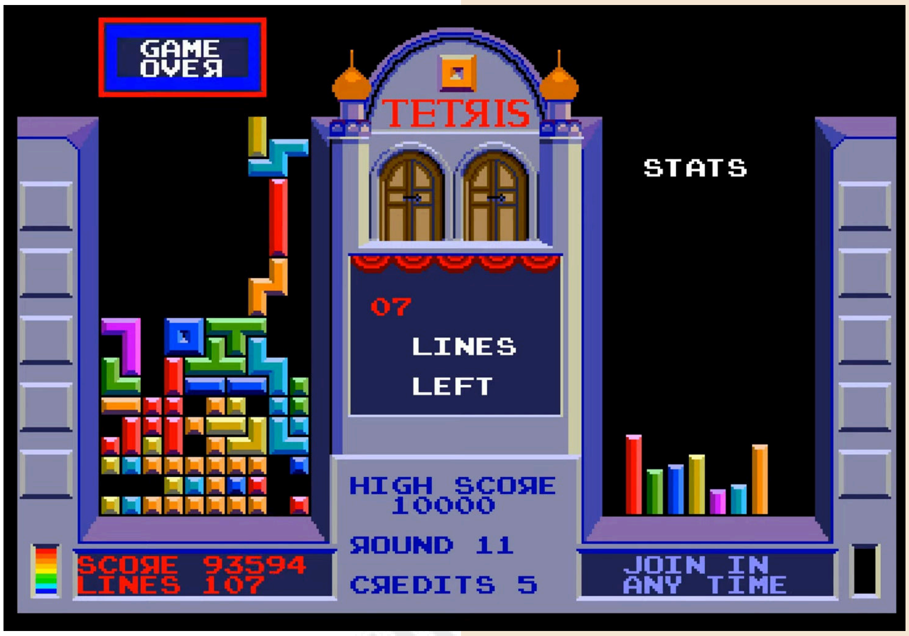
Yes, because we are dealing with the best version of this game and the best transposition of Tetris on the Amiga.

It requires a 68020 processor and the AGA chipset, so an Amiga 1200 is needed as a basic configuration.
It works perfectly in emulation and is available in ADF or WHDLoad format.

It only remains for me to recommend that you download the title and enjoy it. Tetris belongs to those games that are literally endless in terms of playability. Immortal!

by **Carlo Nithaiah Del Mar Pirazzini**





NEW GAME

BREAKTHRU

Year: 2024
Editor/Developer: Acidbottle
Genre: Shoot em up
Platform: Amiga
Website: https://acidbottle.itch.io/breakthru

Another historical title of the arcade era arrives on the Amiga and does so thanks to the golden little hands of Acidbottle, who had long ago delighted us with Wonderboy.

The game is a horizontally scrolling shooter in which the player controls an armed vehicle and is tasked with recovering a powerful fighter jet called PK430 stolen by some bloodthirsty terrorists.

To complete the mission, you have to 'cross' five different enemy strongholds located in different scenarios.

The opposing forces consist of soldiers, tanks, helicopters, jeeps and more.

The game, known in Japan as Forced Breakthrough, was released in 1986 by Data East and was quite successful. In the same year, home computer versions for the ZX Spectrum, Amstrad CPC and Commodore 64 were released under US Gold licence.

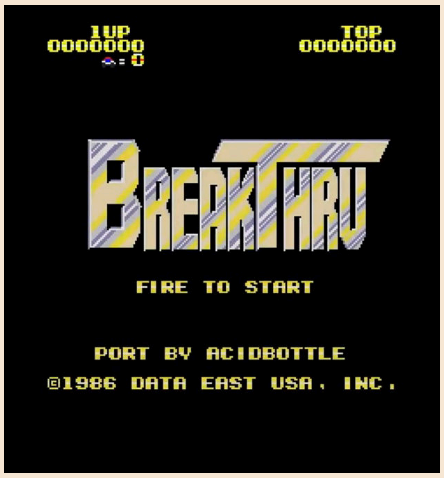
This Amiga version perfectly replicates the spirit, structure and technical capabilities of its arcade counterpart.

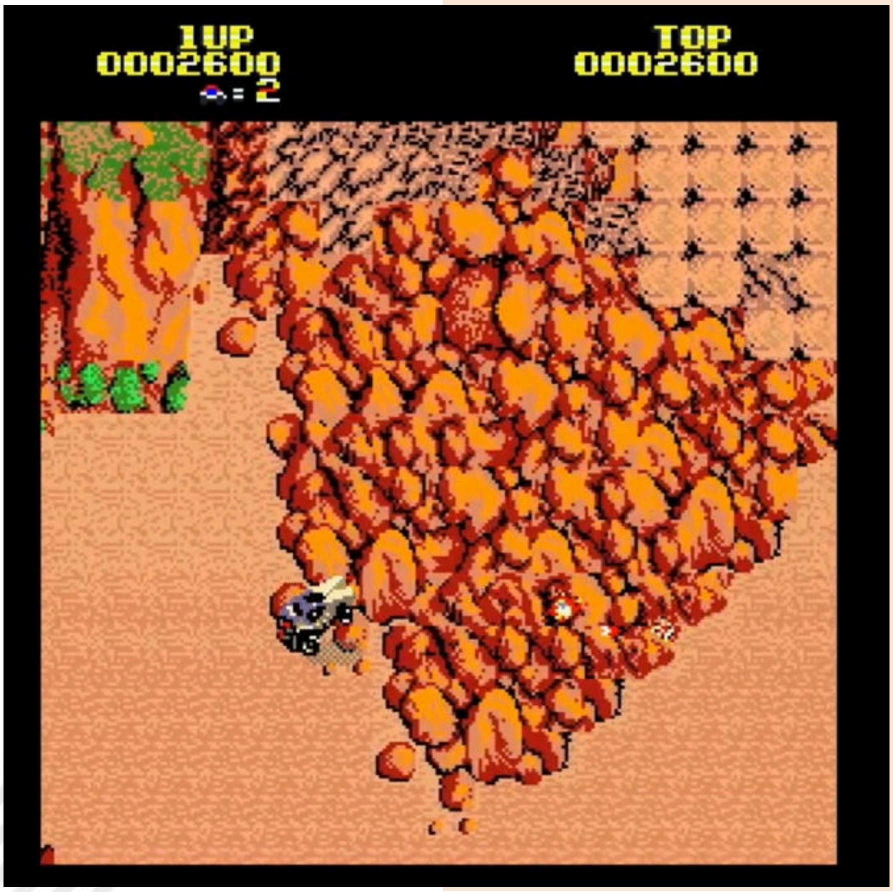
There is the whole game, including the use of the double button (one button fires and the second allows you to jump with the bike).

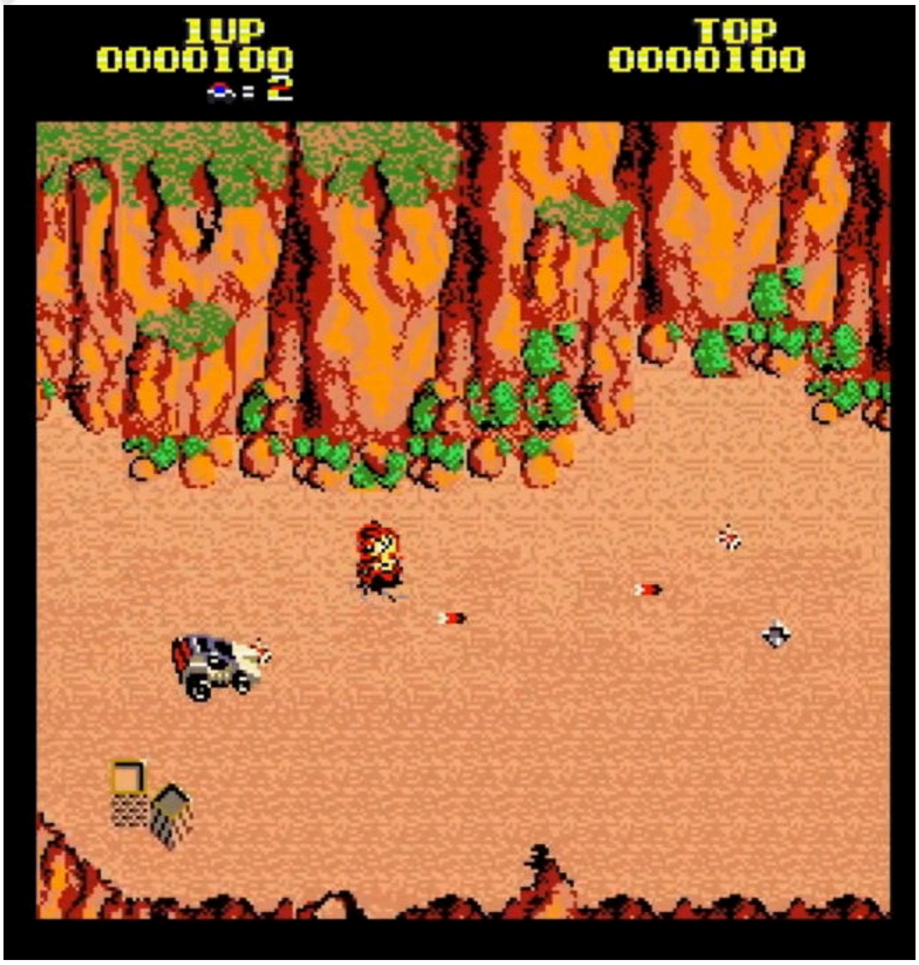
There are all available paintings, enemies and bonuses.

The current version (1.0) still lacks the sound effects that will be implemented, according to Acidbottle, with the next two releases, but there is still the well realised soundtrack.

It is a title from the past. A coin-op classic of the era with a high level of difficulty. The levels have to be memorised as much as possible and are definitely tough from the very first one.
This, however, does not detract from the game's charm and the desire to click on the 'continue' buttons to see how the game will end.



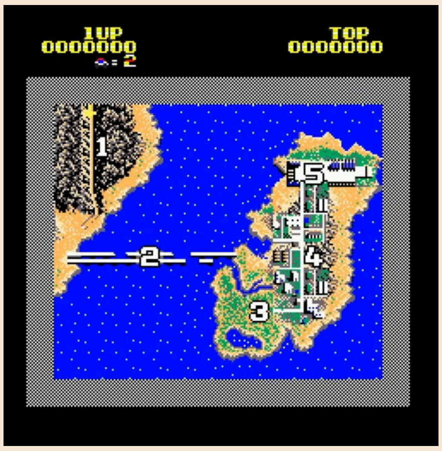




OUR FINAL SCORE

» Gameplay 85%
Data East's title through and through, difficulty included.

» Longevity 75%
Tough game that was born with the objective of 'eating pennies'. The Amiga version is true to line.

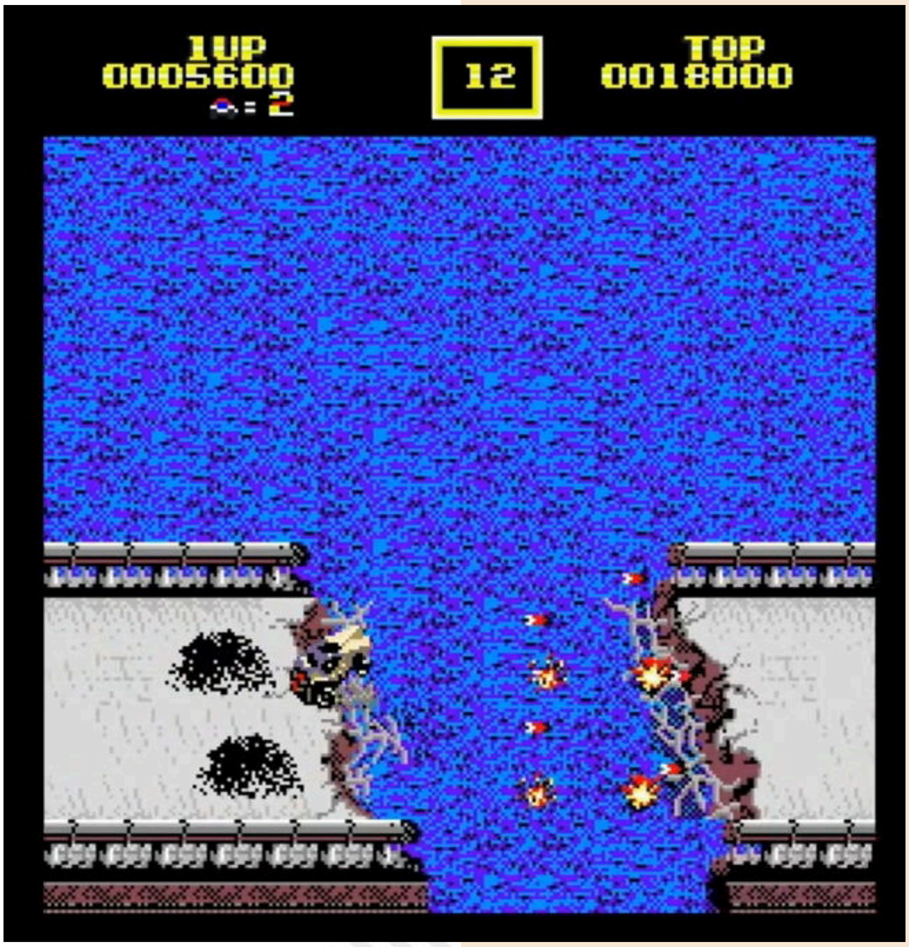


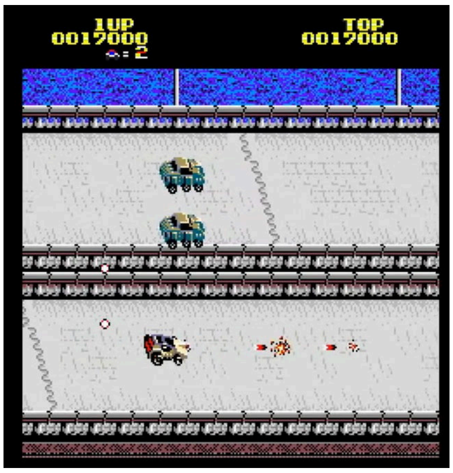
The game has been released in ADF format and requires 1Mb of Chip Ram and 1Mb of additional Ram.

A WHDLoad version and a version for ACD32 are planned.

Amiga is reviving a second life as far as conversions of arcade titles are concerned and this can only be a good thing.

by **Giampaolo Moraschiby**





NEW GAME

FINAL FIGHT ENHANCED

Year: 2024
Editor/Developer: Brick Nash/ Prototron
Genre: Beat 'em Up
Platform: Amiga
Website: https://drive.google.com/drive/folders/1GydNnHlY93p00ePcHNwD7xtn-zmtPPe0

Final Fight has that mythological 'something-or-other' about it. A title that made us love arcades. A game that made us feel like heroes.

Liberating a maiden from the hands of the terrible criminal organisation Mad Gear was about as 'heroic' as a child of the early 90s could get.

Developer Yoshiki Okamoto made a great product in 1989 for Capcom. Conceived as a possible sequel to the questionable Street Fighter, the game took on a life of its own thanks to its dynamic gameplay and story and spawned several sequels and similar titles (like Captain Commando or Cadillacs and Dinosaurs for example).

Like many games of the time it was converted for numerous home systems. Who can fail to remember the terrifying conversion of US Gold for C64, slaughtered by Zzap! with a resounding 22%? Or the lumbering Amiga version, which looked perfect at a standstill but was delirious on the move?

Unfortunately, home computers did not boast direct development like consoles (Capcom personally took care of development on Nintendo and Sega platforms with two decent conversions), but relied on the (not too experienced) hands of US Gold teams.

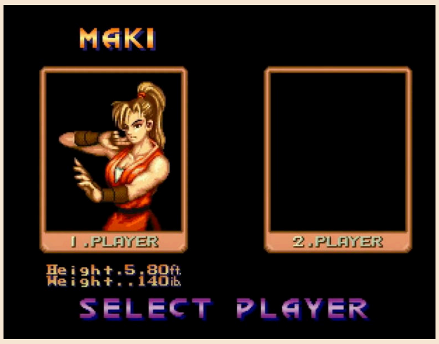
Years have passed since the wacky Amiga version and here we are, in front of our 16-bit Commodore to once again talk about the story of Cody, Guy and Haggar.

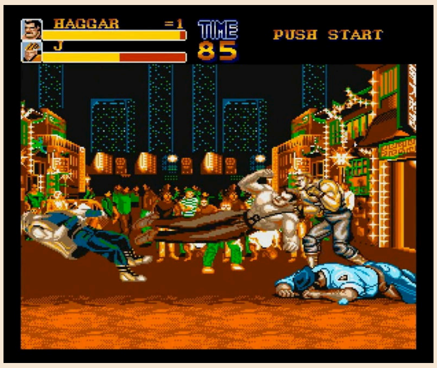
What does this Enhanced version bring new to what we have already seen?

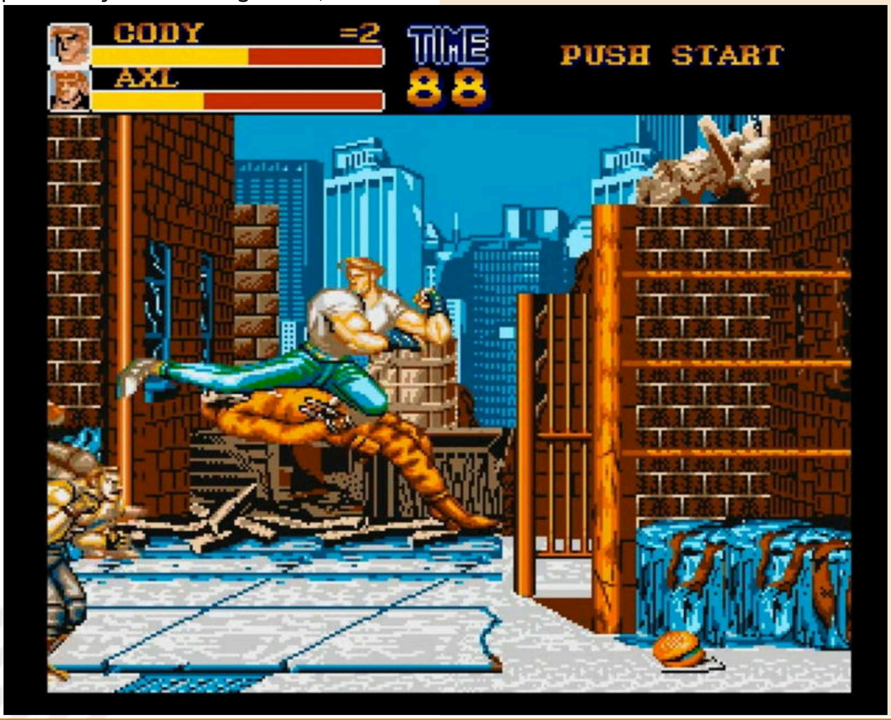
Firstly, it brings, shall we say, strange resources. For one thing it runs on Amiga ECS and does not require higher chipsets or RTGs. Good news for all basic machine owners right? Unfortunately not.

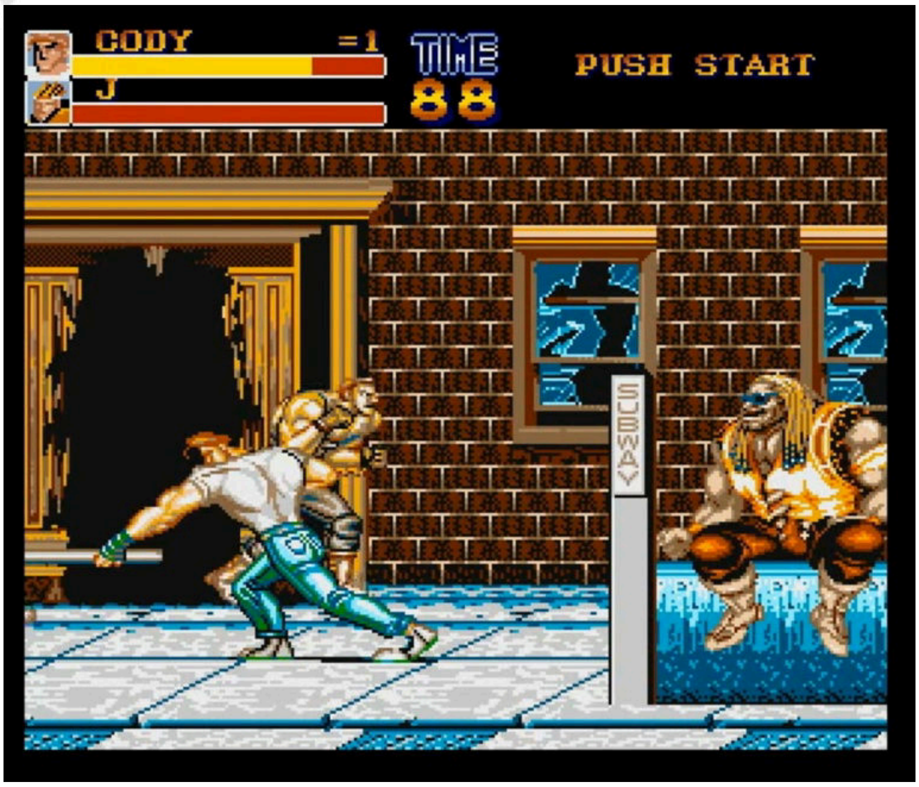
To run properly it needs 2Mb of Chip Ram and an additional 512k of other Ram (fast or slow). A bit much for basic systems but perfect on advanced systems.

Obviously, all this on WinUAE has no problem, but already on an A1200 the product has discrete difficulties. Among the new features is the possibility of selecting MAKI, a female









OUR FINAL SCORE

» Gameplay 80%

The title always has a certain charm and the possibility to play with Maki with the addition of the second button are a really good thing.
Pity about the 'anomalous' configuration to be played on real hardware.

» Longevity 85%

A great game even today despite the fact that years have passed and hundreds of thousands of beat em ups have come out. Always nice to play.

character from Final Fight 2. An interesting character to play for her speed and variety of blows.

There is a greater speed of reaction in the game compared to the 'woodenness' of the original title.

The title was developed in assembly 68000 by the author with the aim of learning this language.

I must admit that it is a good piece of work and all in all successful. We have no uncertainties in the fluidity of the levels and, despite small bugs here and there, everything runs smoothly and enjoyably.

The implementation of the two game buttons is good, making the experience certainly more similar to the arcade original.

Personally, I would have preferred a version optimised for AGA platforms or higher. With such high demands in terms of memory, better to go for a more 'important' palette similar to the arcade version. But this is simply a matter of personal taste.

US Gold's original was a sad product, this one is definitely better.

The title is available in two formats: the first in adf on two disk files and the second in WHDLoad format.

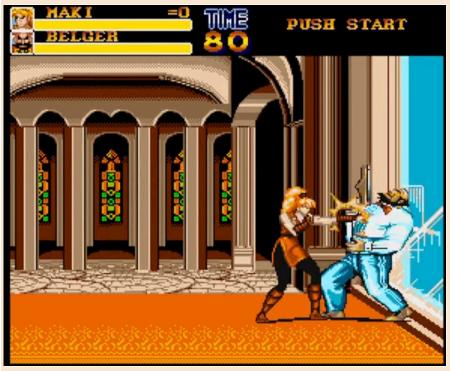
The project is always in the process of 'renewal' according to the author. He speaks of future corrections and this can only be welcomed.

Drawing the proverbial sums, we can say that this version of Final Fight is promoted despite some flaws and that we can forget the horror of US GOLD.

by **Giampaolo Moraschi**

Don't hang up! We have a little business proposition for you...

Your daughter for your cooperation!



Oh Dad!
I was so scared...

I'm so glad to see they didn't hurt you.
I'm so sorry Jessica. I thought I'd lost you like I lost your mother.
I'll never let anything bad happen to you again.

I love you Dad!

NEW GAME

DOTTIE FLOWERS

Year: 2024

Editor/Developer: Goldlocke

Genre: Platform

Platform: Super Nintendo

Website: https://goldlocke.itch.io/dottie-flowers

The little witch Dottie returns on our Super Nintendo after her adventures in Dottie dreads nought.

Goldlocke's new platformer is beautiful, colourful and bizarre.

Bizarre in the way it is distributed. In order to play it in full, you have to finish the demo on the website and read the instructions carefully.

The young witch Dottie finds herself embroiled in a serious predicament; the invasion of planet earth by a robotic alien civilisation.

After a cute animated intro, you embark on the adventure through six game phases.

The first phase is a kind of tutorial showing us the capabilities of little Dottie. We are shown how to jump, how to use the magic broom, the super spin and much more.

The structure of the game is that of all classic platformers. A beautiful map and levels to tackle in pure Super Mario World style.

As I said, it is super colourful, animated well and with graphics reminiscent of the previous chapter and the 'kawai' graphic style of the Goldlocke team. I really liked the colour choice and the soundtrack.

As specified on the website, it is not a commercial project and the physical version is only available by finishing the demo.

A simple but enjoyable title that is finished in about 30/60 minutes.

I would definitely have liked to see more action, but considering the developers' experiment, it's OK.

by Ingrid Poggiali

OUR FINAL SCORE

» Gameplay 80%
A cute platform game that is fun to discover and play.

» Longevity 70%
It ends too quickly.

31
32 Hit
Use it to drill through walls:
Hold A to charge, then release.

NEW GAME

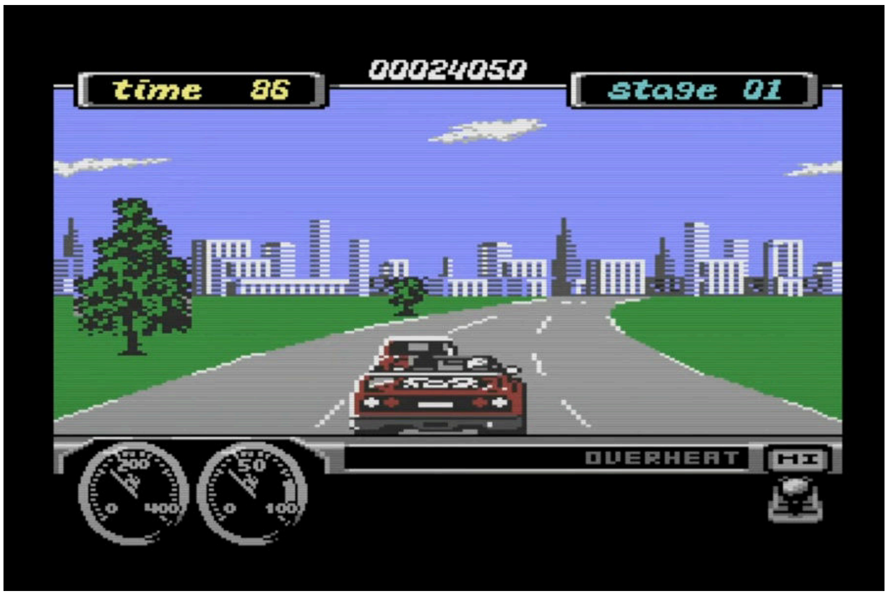
TURBO OUTRUN

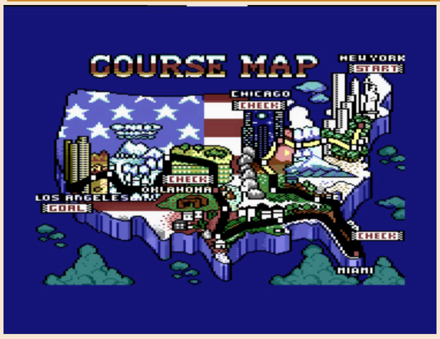
Year: 2024

Editor/Developer: TCFS

Genre: Driving

Platform: Plus/4





Turbo Outrun is a classic from the good old days of arcades.
Everyone wanted to play it at home on the 8- and 16-bit systems of the time. They wanted it as they did in the arcade, racing down long tracks at full speed.
In 1989, the same year as the arcade release, it was converted on various machines, published by US Gold in Europe and developed by Probe Software for the C64 and Ice Software for the Amiga, ST, CPC and ZX.
The C64 version received a very good reception with positive reviews in all the magazines of the time.
Well made, fast and with a good resemblance (in terms of gameplay) to the original game.
Needless to say, Ice Software's version for Amiga/ST and CPC/ZX was not as successful.
From 1989, we move to the beginning of 2024 and do so on a platform that was little understood when it was launched but is experiencing a 'new youth': the Plus/4.
Developer TCFS, starting with the excellent C64 version, has pulled a beautiful version for 'the other' 8bit from Commodore out of its hat.
And what a version!
Beautiful to look at and to play.
Fast enough and in every way on a par with the version released under the US GOLD label for the C64.
The game is still the same: you race with a perspective view behind the car, as in the classic Outrun, with the option of selecting between automatic or manual gearbox.
Instead of the Ferrari Testarossa we have a faster turbocharged F40, variable tracks, police... in short, it's all Turbo Outrun.
The graphics work well and the TED doesn't make you regret the SID.
What was the story about the Plus/4 machine being underpowered or underused?
Let's enjoy this title.

by **Giampaolo Moraschi**

OUR FINAL SCORE

» Gameplay 90%
Very good conversion of a driving game!

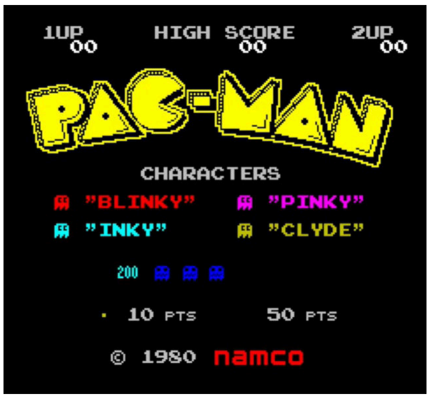
» Longevity 90%
This is a title you will reload every time you get the chance, trust me.

NEW GAME

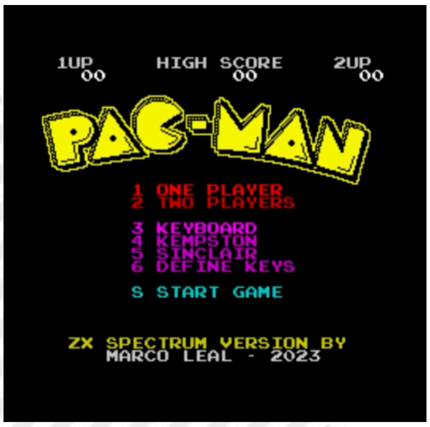
PAC-MAN ARCADE

Year: 2024
Editor/Developer: Marco Leal ZX
Genre: Maze
Platform: ZX Spectrum 48k
Website: https://marco-leal-zx.itch.io/zx-pacman-arcade

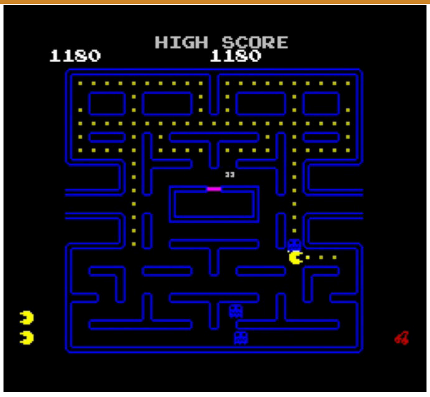
Iwatani in 1980 created an absolute icon, the round Pac-Man.
He was born to diversify from the flood of Space Invaders and Arkanoid clones of the period and was born for fun.



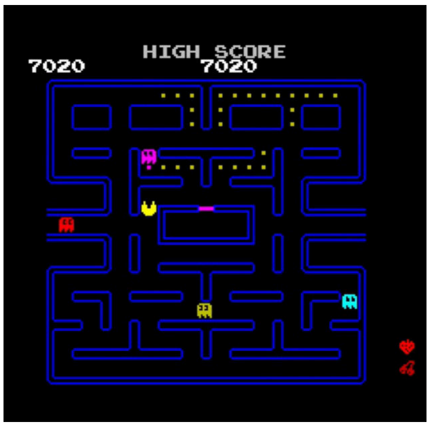
Its countless evolutions were memorable, as were the thousands of more or less direct conversions for all gaming systems from 1980 to the present day.
In 2024 we are here once again to talk about Pac-Man and we do so on the ZX Spectrum through a PERFECT version.



Perfect in form, game logic and visual aspects.
Marco Leal, the developer, has created a superlative product in the very few KB available to him.
All the game frameworks, ghost assault

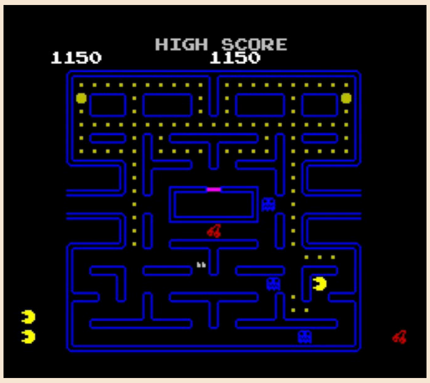


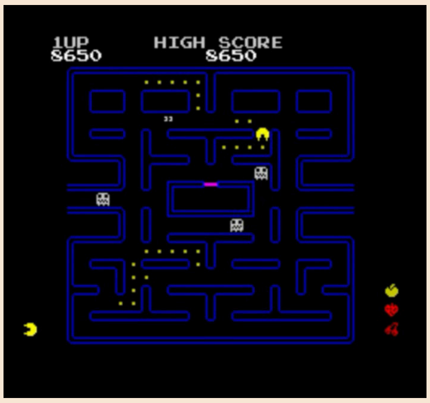
logic, bonuses and pills are there.
It runs smoothly this PAC-MAN Arcade, with its 50 fps on a simple ZX Spectrum 48k. What a wonderful machine isn't it?
Just think, Marco inserted the Game Over screen after level 256 which in arcade caused the system to crash. What a perfectionist!
Two-player support and downloadable from the site in tzx format compatible in emulation and on virtual machines.

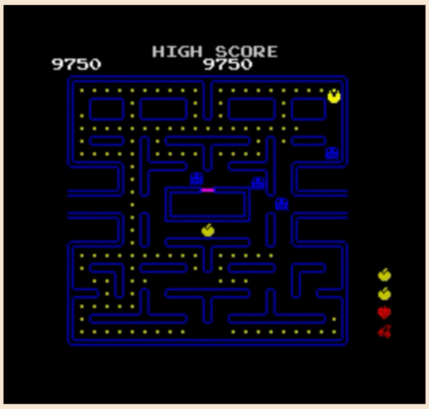


Can you still have fun with Pac Man in 2024? Yes. Definitely yes, it's as much fun as it was in 1980 and on the Spectrum it's even better.

by Carlo Nithaiah Del Mar Pirazzini







OUR FINAL SCORE

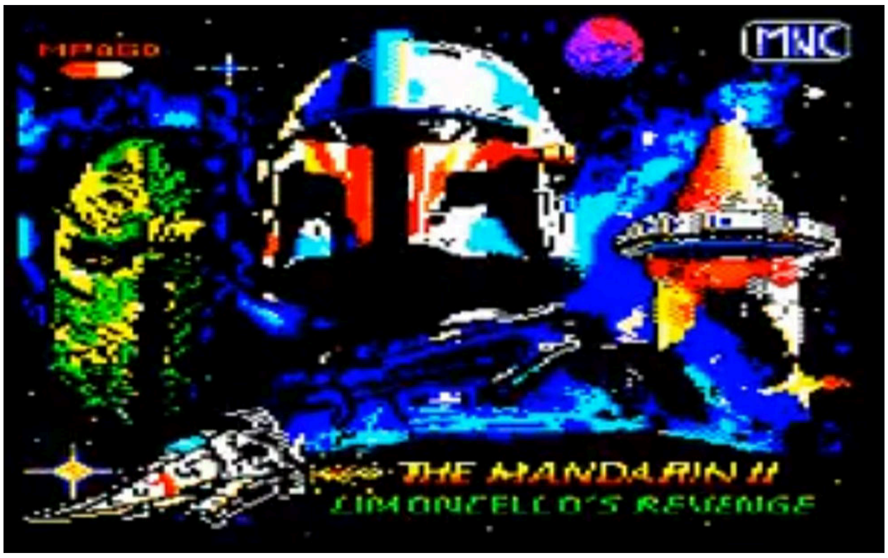
» Gameplay 90%
It's perfect in everything and looks great.

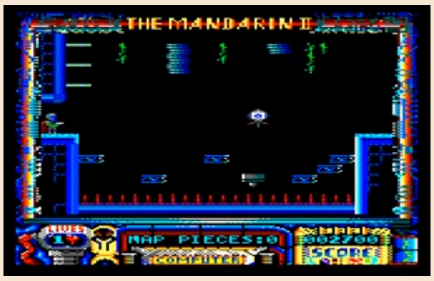
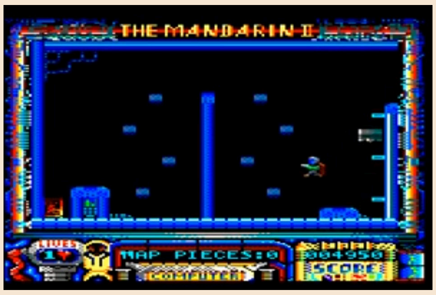
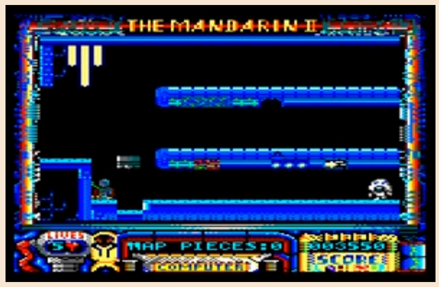
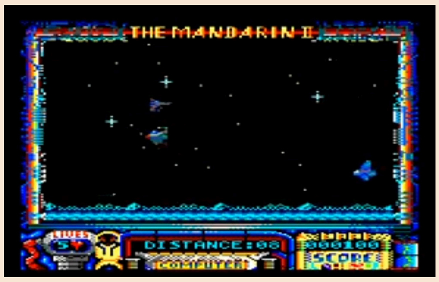
» Longevity 90%
It requires dexterity as in the original one and a lot of practice, but it does not tire.

NEW GAME

THE MANDARIN II

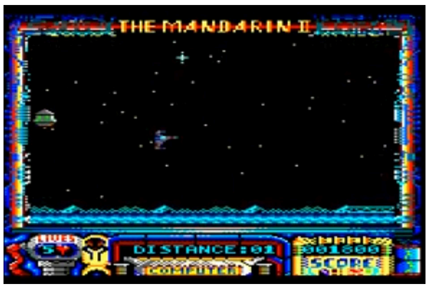
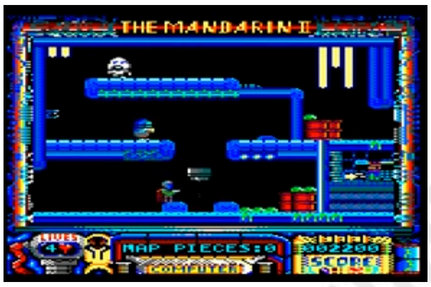
Year: 2024
Editor/Developer: Mananuk
Genre: Action/Platform
Platform: Amstrad CPC/ZX Spectrum
Website: https://mananuk.itch.io/the-mandarin-2-amstrad



Take The Mandalorian straight out of the Star Wars saga... change its name and add a dash of irony to the lyrics and there you have it... THE MANDARIN. A cute action platformer for Amstrad CPC and ZX Spectrum.
The first title followed the events more or less of the first series this second title takes a bit of the second series and mixes it with a search for the plans stolen by the terrible imperial general Limoncello (yes, we have the name of the century...).
The game consists of five different levels/phases. We will have a shoot em up-like phase, a platforming phase and some puzzle phases.
All are well developed in terms of gameplay and level design.
I personally loved the escape phase of the fourth level for its speed and tension.
The Amstrad CPC version is well realised. Beautiful and colourful graphics and a good basic sound made of simple but effective sound effects.
There is good use of colour and everything we see on the screen is understandable and sharp.
The ZX version suffers from some minor graphical glitches but from a playability point of view we are still on the same level as the Amstrad version.
A good, simple and fun title that perhaps ends too soon by the average standard.

by **Giampaolo Moraschi**

OUR FINAL SCORE

» Gameplay 85%
I like multi-event games and the ability to approach them with different logic.
Well-made control system in both versions.

» Longevity 65%
It all ends too soon.
Unfortunately.

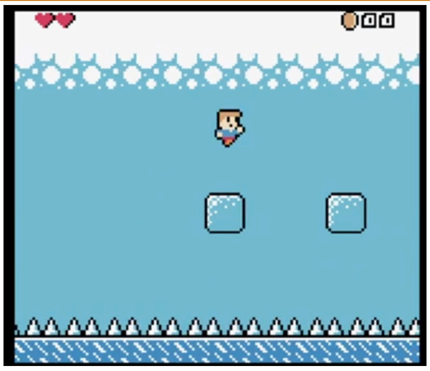
NEW GAME

SERGIO KIDD

Year: 2023
Editor/Developer: PCNONOGames
Genre: Platform
Platform: Game Boy Color
Website: https://pcnonogames.itch.io/sergio-kidd-for-gameboycolor

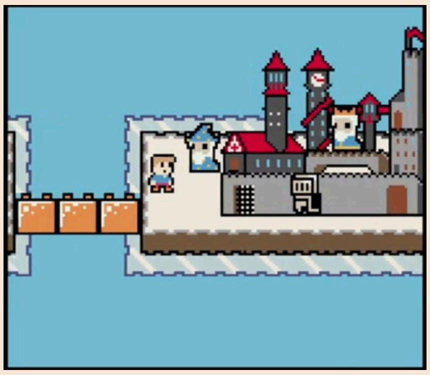
An evil king who imposes new taxes... citizens who rebel... a wizard who turns them into monsters... Plants, poisonous mushrooms, coins to collect... But what is Super Mario?
No it is Sergio Kidd, the new platform game from PCNONOGames with a decidedly odd name but certainly good gameplay.
The title is available in a digital version for EUR 3.95 or in a physical version (can be ordered on the site).

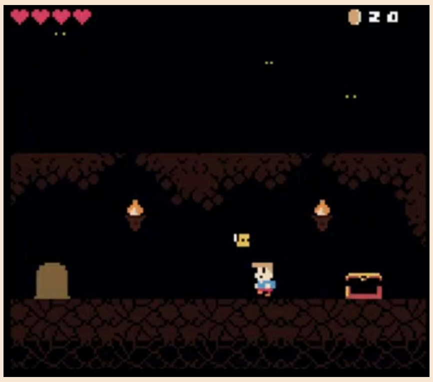
What it's about... or rather what it's about when we enter Sergio Kidd into our GBC.
So it's a platformer of the most classic kind divided into five different worlds with more than 25 levels to explore.
As in all self-respecting platformers of its kind, we will be faced with several rather angry monsters, traps, carnivorous plants, objects such as ladders and coins to collect.
It can be played with pleasure and the worlds are nice and well structured.

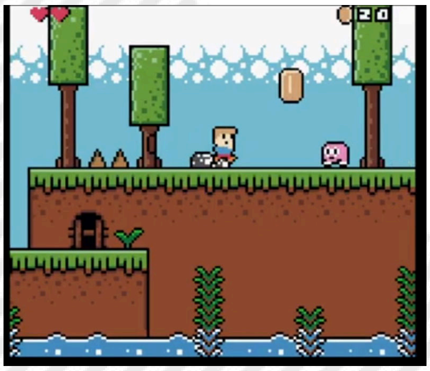


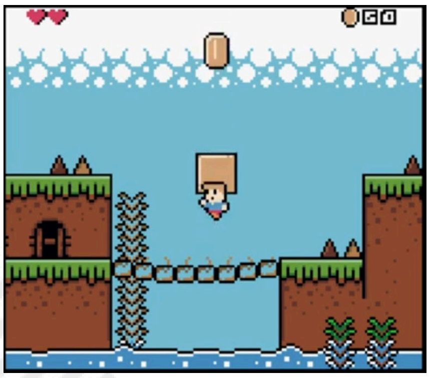
Without a doubt, the influence of the Mario games is evident. Many monsters and a lot of design are reminiscent of some chapters in the saga of the Nintendo plumber, but I don't think this is a bad thing.
There are colourful graphics and a variety of animations, good sound, but maybe little Sergio doesn't have the right charisma for this game... I don't know... I thought it was weak, anonymous.
The controls respond well and the game can be played with gusto.
It's not a very difficult title to get through, especially if you're a Mario Bros and related 'pro', but for 3 euros and a little more it's perfectly fine.
I'm happy to take it to university... on the train, it's the perfect game to pass the travelling time.

by Ingrid Poggiali









OUR FINAL SCORE

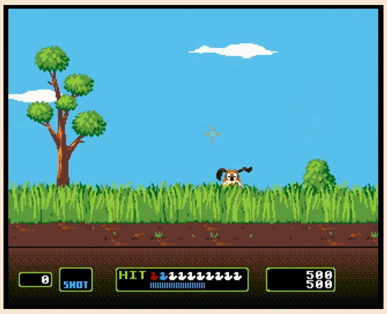
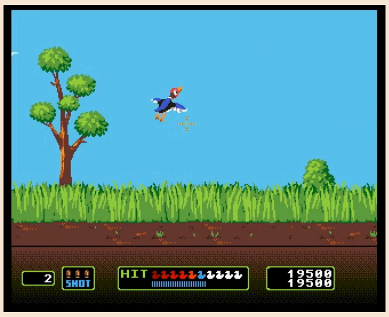
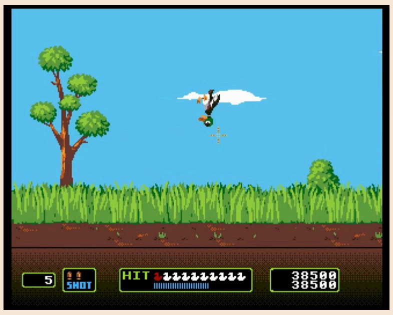
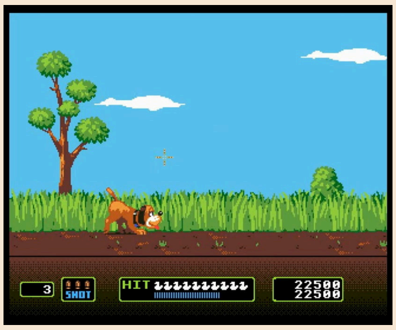
» Gameplay 80%
Good controls and solid platformer structure.

» Longevity 70%
Not too challenging if you have mastered this type of game.

NEW GAME

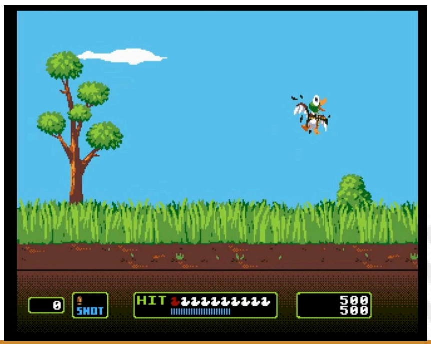
DUCK HUNT

Year: 2024
Editor/Developer: Marggines
Genre: Shoot em up
Platform: Amiga
Website: https://www.ppa.pl/forum/strefa-gier/46460/duck-hunt-kolejna-nowa-gra#m807596

Tell the truth. When you played Duck Hunt on NES, didn't you feel an absolute sense of hatred towards the hunting dog? With its evil giggles when the ducks wouldn't come down? Evil quadruped...
Duck Hunt was one of the must-have titles for Nintendo's 8-bit console along with Super Mario. Capable of selling a ton of Action Sets and Zapper guns.
Simple, fun and well thought out. Utterly irritating when being 'beaten up' by the henchman in our wake.
This unexpected Amiga version comes straight from Poland. Its developer provides us with two files (adf and lha) where we find a colourful transposition of the NES game.
The graphics are not his own work but, he is keen to point out, taken from this site: https://www.spriters-resource.com/custom_edited/duckhuntcustoms/sheet/63915/.
He has not released any technical specifications. No, we tested it on an A600 with 2 mb of Ram and in emulation: everything ran smooth as silk.
The game is all there and is played with the mouse: there are the two modes (single duck and double duck), there's the dog, there's the music.
It tires you in the long run, but it has two great qualities: it doesn't require much mental effort and above all it's free... at least until MAMMA NINTENDO has it deleted from the net. Amusing.

by **Carlo Nithaiah Del Mar Pirazzini**



OUR FINAL SCORE

» Gameplay 90%
Shooting birds is still fun... even on the Amiga!

» Longevity 85%
A nice title to load up every now and then for a quick game.

KEYSTONE KAPERS

Year: 1983

Editor: Activision

Genre: Platform

Platform: Commodore 64/ Atari 2600

Today we will relive our childhood playing 'cops and robbers' even if only virtually. I admit I wouldn't mind replaying it live by adding this ingredient to our nostalgic diversions of retro consoles and home computers.

The game review is about Keystone Kapers, published by Activision in 1983 for the Atari 2006 and the following year for other Atari consoles and beyond. There was a lack of conversion for the 'breadbin' during those years, it's true. But in 2019 here it is! Thanks to a willing and talented programmer, as passionate and nostalgic as we are, we managed to get it over thirty years later! The surprises, as you see, never end.

In the game we impersonate a funny policeman, named Keystone Kelly, inside an 80s-style shopping mall and we are in pursuit of an escaped convict, wearing the classic striped uniform, named Harry Hooligan. Harry roams the mall looking for escape routes and grinding out kilometres on the various floors and escalators. All the cops have to do is catch him and it won't be easy on three floors full of obstacles such as aeroplanes, bouncing balls and shopping trolleys whizzing by at full speed. In addition to these, there is also a time limit (typical of arcade games of the time) to complicate things, as every collision with obstacles (except the aeroplane, which will cost you a life) will subtract a few precious seconds and once it reaches zero the game will end directly.

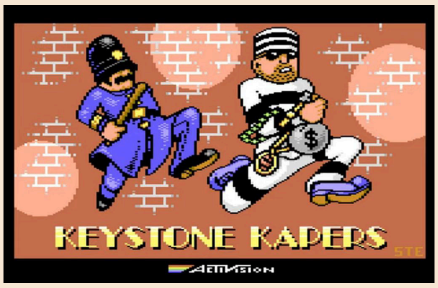
On the floors there are also bonus items to collect, which are essential to reach the 10,000 points to earn an extra life. Luckily on our side there are lifts to move faster on the floors, assuming we don't get stuck like in a 300-family apartment building. Oh I forgot to mention that we will also lose a life if we make the escapee escape through the roof.

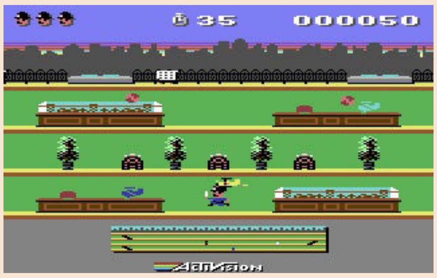
The game does not have an ending and only once we have lost all the lives can we say we have 'finished' it. I find this game to be quite long-lived for the simple fact that the urge to pick it up for a game every now and then is always there, whether it is during a work break or an evening when there is nothing good on TV. The accompanying music is very nice.

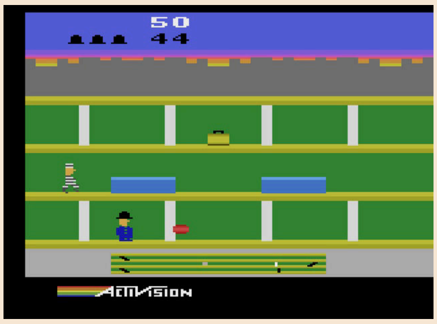
Reviewing the very first version for Atari 2006 I found it very nice and playable and I must say that the conversion on the C64 is very successful, I'm not only talking about graphic elements. A second chapter was also released for the Atari console, not yet present in the home computer version and I hope to see it released soon...

This conversion has spoiled me a bit, like cherries: if you haven't had a chance to play it yet, do so as soon as possible because Activision's signature is more than a guarantee and in those years it made us very entertained and dreamy with its big titles. For a great start to 2024 I'd say it's just the thing, until the warm days arrive and get us out of the house once again. And with the days already starting to get longer we might even play 'cops and robbers' with our children!

by Daniele Brahimi







OUR FINAL SCORE

» Gameplay 80%
Simple, linear control.

» Longevity 50%
Only three levels, but evocative and never repetitive!

Is patience a virtue?

As you have probably noticed, this issue of RMW has been a little longer than expected. Don't worry, we have not run out of content, nor have we run out of ideas. We just wanted to take some time to fulfil some tasks that we have been putting off for too long and to spend some precious time with our families.

In these months of absence, we have caught up with the layout of the English issues. So why haven't you published those issues yet?
Simply to give international readers time to read one magazine at a time and allow technical time for each issue.
If we had published three issues at the same time, we would have risked overcrowding the square and losing appeal. And we do not want that!

Returning to this issue 24-EN, I must admit that I myself was amazed by the quantity and quality of the games featured. One only has to take a look at the ratings box to immediately realise that the quality level of 8- and 16-bit productions of this period is incredibly high. 'If it's a dream, don't wake me up', someone would say, but fortunately it is the pure and simple reality of the facts. Talented programmers, graphic designers and musicians have rediscovered the pleasure of writing code, designing and composing music for machines from 30/40 years ago.
And we are fine with that, since we can often enjoy the fruits of their labours for free or at a negligible cost.

Before closing this issue for good, I want to direct your attention to the last page of the magazine.
I don't want to spoil the surprise for you, but as mentioned in issue 23-EN, there are some fun new features we are preparing for the next issues and... you won't be disappointed, I'm sure.
Of course, we will take our time to do things in the best possible way.

After all, isn't patience a true virtue?

Francesco Fiorentini

RetroMagazine World
Year 4 - ISSUE 24 - AUGUST 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

MR. JACK

Sarà uno dei protagonisti dai prossimi numeri delle recensioni... assieme a Sir Clive, NISHI SAN e tanti altri.

Opera del nostro Fabio De Renzis

RETROMAGAZINE WORLD