Commodore Free

A free to download Magazine dedicated to Commodore computers.

Issue 80

CONTENTS

Editor
Nigel Parker

Spell Checking
Peter Badrick / Bert Novilla

TXT, HTML & eBooks
Paul Davis

D64 Disk Image
Al Jackson

PDF Design
Nigel Parker

Contributors
Richard Bayliss
Joseph Rose (a.k.a. Harry Potter)
John Fielden
Lenard R. Roach
Bert Novilla (satpro)

Website
www.commodorefree.com

Email Address
commodorefree@commodorefree.com

Submissions
Articles are always wanted for the magazine. Contact us for details. We can't pay you for your efforts but you are safe in the knowledge that you have passed on details that will interest other Commodore enthusiasts.

Editorial

This issue sees an interview with "pixel" and don't let the cool name put you off. You may remember I reviewed his game "Pulse for the Unexpanded VIC-20" in Issue 79 of Commodore Free. I know (thanks to the people who contacted me) I made some negative comments about small graphical glitches. However, I feel if these *were* cleared up, or if indeed it was possible for them to be cleared up, then the game could be another outstanding release for the VIC (actually it is outstanding as it sits now). I would especially like to see this released as a tape version. I can just see the loading screen and VIC music pumping out as the game loads into memory. The screen clears, then the game is de-crunched, and then... the rest of the game loads and – BOOM! You're right in the Pulse of the action (can you see what I did there with the name?). Pretty cool, ehh? Anyway, I have £10 waiting to buy the tape version (should it ever be released). I have emailed a couple suppliers of Commodore releases, and I suggest you do the same. Demand will drive the creation of the product!

We also have an interview with Nic, who has created, amongst other things, the C64p. This is a portable laptop based on a customised DTV. The price and compatibility issues may be a sour point for some, however, but the beautifully crafted design looks like it rolled out of the factory – not some hobbyist's workshop or bedroom.

Lenard R. Roach shares some of his thoughts about the Commodore 64 and 128, along with some of his programming problems, and the software that almost made him a millionaire (well, a better line than a few quid).

Joseph Rose tells a little about optimising cc65 code, or as he says, "Some tricks and tips on creating some lovely looking C code on the Commodore 64." Then we learn a little about Spaghetti code. (Written By John Fielden)

We then jump back to the VIC again (jumping around here for no reason other than to keep you on your toes). This issue sees an interview with Dane Bills, who, as you may know, created panicman. Can you guess what it's a remake of? Not only does it look and sound good (you may remember Issue 79 had a review of the game), the game play is very close to the real arcade machine. Dane talks about how the game came into being – and the tools and experiences he gained from coding his first 6502 project on the VIC.

Sadly, this month sees the last of the E-tapes, mainly because it's become far too difficult for Richard to find anything to put on the E-tapes. However, if you have a program or application and would like it distributed via Commodore Free on our E-tape download, feel free to email me (along with some instructions, of course!). The game or application must be your own creation and must be released as public domain (or as freely-distributable with Commodore Free).

Finally, we have Issue $03 of our 6502 assembler course. You guys should be really racking up some good coding knowledge by now, so make sure you have enough caffeine and are ready to learn some more new skills (heck, I thought a Stack was like a six-pack – turns out I am totally wrong). I have had quite a few emails about this series saying how they are enjoying following the articles.

Wow, I didn't know there was so much cool stuff in this magazine, so……..

Just in case it hasn't sunk in, can someone please start a campaign for "Pulse" to be released on tape for the unexpanded VIC-20? Thanks for reading, and as usual, please send any comments and suggestions to me.

Thanks to everyone who has helped out with this issue, and as a final reminder – if you want to help with Commodore Free or have an article or idea for the magazine (or even a news item), please get in touch with me!

I am off to read the issue as it sounds great this month. J

Regards,
Nigel (Editor of Commodore Free)
Website www.commodorefree.com
email
commodorefree@commodorefree.com


Your source of Commodore news
Commodore
A place for everything Commodore
is Awesome
Magazines
Amiga Format
Amiga Bladet
Zzap64
Scans
Games
Demos
Applications
AMIGA
Commodore
THE GREAT NEW PACKAGE FOR AMIGA OWNERS
CiA
Christmas is Awesom
CiA
http://awesome.commodore.me

Commodore C64

Back to the Future!

Connect to TV and Play!

- 2 MByte FLASH
- 16 MByte SDRAM
- S-Video out
- Stereo Audio out
- PS-2 keyboard and mouse
- Micro SD card
- 2 Joystick ports

more than 100

C64 games!

C64 Forever

cloanto

Online Order: www.arcaderetrogaming.com

More Information: www.mcc-home.com

ARG

COMMODORE FREE E-COVER TAPE 13

Compiled by Richard Bayliss

Yet again the E-Cover Tape is here, but sadly, this issue is the last one. This issue's E-Cover tape features all of the entries from the 2014 SEUCK Compo, which were released between the start of the competition through 18th April 2014. Also, there's the second installment of the Loader Game Tape Master Kit. You may have also noticed something completely different about this issue's E-Cover tape. The tape loader system has been changed to increase the speed of mastering. The deadline for the final E-cover tape was pretty close. So R-Load by Daniel Kahlin was used. It may be a quiet loader, but you get some nice raster effects amongst the border. :)

... and now for the last time I say, "*It is time to lock in and load.*" :)

SEUCK COMPO 2014

To start off this issue's E-Cover tape, we have some amazing SEUCK Compo goodies for you. Practically all the entries of the 2014 SEUCK competition, which was entered between January 2014 and now. Closing date is of course 30th April. Due to deadline extensions, the following featured games are as follows

Another Day – Another Zombie

(C)2014 Carl Mason
Programming: Carl Mason (Using Sideways SEUCK)
Graphics: Carl Mason
Music: AEG/Smash Designs

This is a score/attack game in which you must survive as long as you can against wave after wave of the advancing un-dead. You (and a pistol) are the only thing between a horde of flesh-eating zombies and your camp of a dozen survivors. If any of the walking corpses make it into the camp, all Hell will break loose – and its "Game Over" (if you're dead that is). You can't afford to let one of those zombies get past you.

You can only fire a couple of rounds before you have to re-load, so use your bullets sparingly, as a wasted shot can mean certain death. The faster you cut through the horde, the more points you will acquire as more dangerous zombies (such as sitters which have a highly toxic ranged attack, or boomers that will explode) appear, splattering corrosive bile around its proximity. Also, a zombie combo bonus is awarded for taking out a huge wave of zombies in succession.

How long can you hold back the creeping doom?

Hero Time 2

(C)2014 IndySoft
Programming: Riszard Nazarewski (IndyJR) (Using SEUCK)
Graphics: Riszard Nazarewski
Music: Richard Bayliss

This is a game for one player only, based in the Medieval era. You are a lonely knight who has discovered that the village is under peril against all evil forces of an evil king. As the lonely knight (a true brave hero from the first *Hero Time)*, you must travel across the land and defeat the dark forces that approach you. They are Skeletons, Bats, and Deadly Spiders. Also strewn across the land lies wooden treasure chests. You must pick those up in a bid to boost your score and pick up extra lives.

There are four different levels which you must traverse across. They are as follows:

Level 1 – The forest
Level 2 – The village
Level 3 – The cemetery
Level 4 – The old church dungeons

Can you fight your way to the end – or will your lonely knight become yet another of the corpses in the graveyard? Best of luck to you.

Double or Nothing

(C)2014 Alf Yngve
Programming: Alf Yngve (Using SEUCK)
Graphics: Alf Yngve
Music: Richard Bayliss

Your world is at war, but as the chaos rages around you, your mind seems to drift – your memories grow contradictory. You fear that you are going mad. Are you living in two places at once? Are you experiencing an alternative life in a parallel reality? Are you one man in one world – or two identical copies sharing one mind and two realities? You must stake everything on finding yourself.
It's Double or Nothing

How to play:
Using any joystick, you control two alternative versions of the protagonist. One version (on the left-hand of the screen) fights an alien invasion, whereas the other version (on the right-hand side of the screen) fights an army of robots. Occasionally, if losing a life, both versions of the protagonist will enter the same reality. Eventually both men will enter a shared space outside normal space-time where they can confront the source of the invasions.

NOXUS

(C)2014 Alf Yngve
Programming: Alf Yngve (Using SEUCK)
Graphics: Alf Yngve
Music: Richard Bayliss

A foreign power has constructed a massive new chemical plant with the codename "NOXUS." It is scheduled to release 50 million tons of SO_2 (Sulphur Dioxide) into the Earth's atmosphere. You fear that you are going mad. The leader of this foreign power insists that the SO_2 infusion will reverse global warming and save the world from rising oceans. Reality? Our scientists conclude that this misguided scheme will trigger a catastrophic cooling of the atmosphere, and may even cause a new Ice Age. You must survive the war to find the source. You (our top agent) must immediately fly into enemy territory, infiltrate NOXUS and sabotage it – before the SO_2 goes into production. You have two hours to complete this mission.

Game instructions:
In the flying level, guide the drone which protects your stealth jet from enemy fire. Failure to do so will cause serious damage and your mission may be aborted. Should you manage to make it to the drop zone (marked with a cross-hair), you must leave your aircraft and infiltrate the chemical plant. Stay out of sight! Some guards will raise the alert if they spot you. Sneak behind the guards and take them out with your short-range Taser. Once your reach the control centre, destroy the control panels with your Taser and make your escape.

Good luck with your mission. You'll need it. This message will explode in 10.. 9 .. 8 .. 7 .. 6 .. 5 .. 4 .. 3 .. 2 .. 1 .. BANG!

Shaken – Tales of the Swordless Ninja

(C)2014 Roberto Dillon
Programming: Roberto Dillon (IndyJR) (Using SEUCK)
Graphics: Roberto Dillon
Music: Richard Bayliss

An evil Shogun has murdered all your clan and stolen your family Katana that was passed along from generation to generation. Now swordless, desperate, and armed with only your lethal skills plus a bunch of ninja stars (Shaken), you have to infiltrate the Shogun HQ. Take back what is rightfully yours and get your revenge.

Instructions:
Use a joystick to move around and advance in the game. Press Fire to throw your Ninja Stars.

The game is divided into three sections:

First is the countryside, which plays like Commando and similar games. After this you will be entering the Shogun's town where a more stealthy approach is recommended to avoid evil ninjas and guards. Finally you will meet the Shogun himself in his garden where you can retrieve your stolen sword. This is *if* you survive first.

Vampire Hunter 2

(C)2014 IndySoft
Programming: Riszard Nazarewski (IndyJR) (Using SEUCK)
Graphics: Riszard Nazarewski
Music: Richard Bayliss

After the defeat of Mozgorioth, the evil Transylvanian vampire in *Vampire Hunter*, Astaroth, brother of the beaten monster, is in New York in an attempt to get his revenge on Adam for killing his brother. He wants to kill the entire population using an ancient poison in the city sewage system. Playing as Adam, you must destroy all the evil forces in New York and kill Astaroth, who will be waiting for you in the theatre. You are the fearless Vampire Hunter.

Loader Game Tape Master Kit 2

(C)2014 The New Dimension / Commodore Free
Programming: Richard Bayliss, Martin Piper
Graphics: Richard Bayliss, Wayne Womersley
Music: Richard Bayliss

To end the final E-Tape, we have a second installment of the Loader Game Tape Master Kit. I'll bet you are surprised. This is a simple tool with which you can master your programs to tape, using a fast-loader system and a loader game to keep you entertained (while loading in the program).

Select the program from the Main Menu and highlight the game which you'd like to master your tape with. Enter the filename properties and jump address for your game. Press Record/Play and you are ready to master your programs to tape with a spiffy loader.

You can choose from the following loader games:

Square Pit

First used on the Psytronik tape version of *Assembloids*.
Guide your square around the screen picking up small squares, but avoid getting crushed by the other squares. You have a limited number of lives. After successful loading, pressing CONTROL will allow you to de-crunch and run your game. Keep playing *Square Pit* until you have had enough of playing the game.

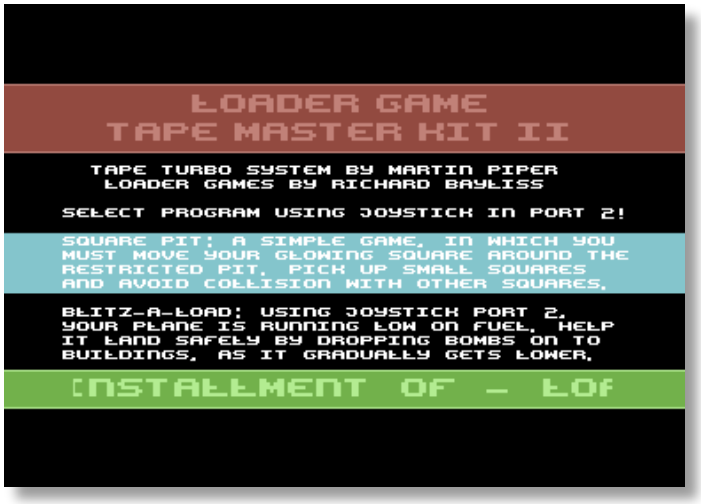
Blitz-A-Load

This is a simple game which features a plane sprite by Wayne Womersley. It was originally for the now-defunct game project called *Up in the Air,* but I got permission to use it anyhow. Press <spacebar> to start the game. The idea is simple: Your plane, running low on fuel, is gradually starting to get lower and lower and needs to land safely. Unfortunately, buildings are in your way. In an attempt to land safely, you must drop bombs onto the buildings using the fire button on joystick Port 2. Each tower has a different height. After one wave is complete, the plane will be able to land safely. Then you will move on to the next wave, where the plane will start at a lower position before-hand. After loading, press CONTROL to run the game you mastered. Otherwise, just keep playing as long as you like.

Back to the master class

So, do you want to master your programs to tape? Is there a restricted file size? Yes, unfortunately there is a restricted file size. The transfer/re-locator routine after the loader game will transfer your

programs from one location (to end-point) to $0801 (BASIC line). Squarepit is $3e00-$cfff, which means 148 blocks max size. Blitz-A-Load is $3a00-$cfff, which means 152 blocks max size, both of which have even BETTER results as compared to the previous tape master kit.



THE END IS HERE

Sadly, that is it – the end of the Commodore Free E-Tape. The next issue of Commodore Free will have no E-Tape; however, if you would like to submit your stuff for future issues of Commodore Free, don't hesitate to send them over to Nigel.

I would like to take this opportunity to say a huge "Thank you" to Commodore Free for supporting my idea. Also of course, a huge "Thank you" to everybody who has been supporting this feature in the past. Cheerio!

SUMMER BLOWOUT!

Buy a copy of Run/Stop-Restore: 10th Anniversary Edition at retail price $17.99 USD and get your choice of either Lenard's latest book, Skits For 2nd Hand Puppets or stay with Commodore and get his software package for only $1.00 (USD) extra!

Software Package Includes:

The Envelope Addressor V4.2

Check it Out

Check Mate

The Ledger

Money Manager 2K (written by Rex Dey)

Hurry! This sale will only be going on from June 1st to September 30th, 2013

FOR ORDERING DETAILS CHECK OUT

www.lenardroach.com

General News

COMMODORE FAN GAZETTE ISSUE 3

Commodore Fan Gazette is a pdf Commodore magazine (Italian language). In this edition: Editorial, Ready...Return!, MorphOS, Aegis Sonix - Amiga OCS, Assembloids, Stunt Car Racer, Super Bread Box, M.A.C.E., Bomberland 64, Amigaro (3), C= 64x, Cartridge games for the C64,. Top 100 and the mail.

http://www.commodorefangazette.com/download.php

Retrogaming Times Monthly Issue 116

Retrogaming Times is all about the retro gaming computers, and features the following articles: Apple II Incider, CoCoLicious, Rejects Gaming Hall Of Fame, More 64! - Avoid The Noid, The Simpsons, Mega Man, Shining Force II, The Pixelated Mage + More!

http://www.retrogamingtimes.com/

Borderline BBS

Borderline BBS is now the first "hybrid" C64 BBS, accepting calls both through dial-up *and* via Telnet! So now you can call at (951)652-1690 or at

telnet://borderlinebbs.dyndns.org:6400

MKD64 RELEASED

Zirias has released *Md65*, a modular tool for creating .D64 images. The main program is for writing tracks and sectors, while loadable modules do everything else, for example writing a directory

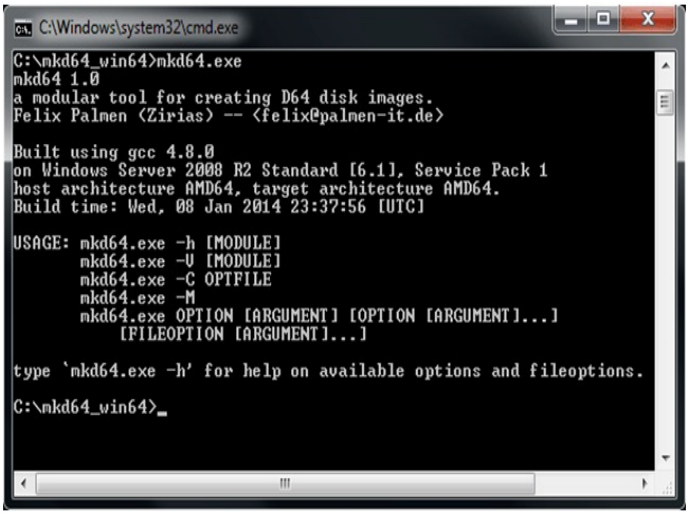
https://github.com/Zirias/c64_tool_mkd64

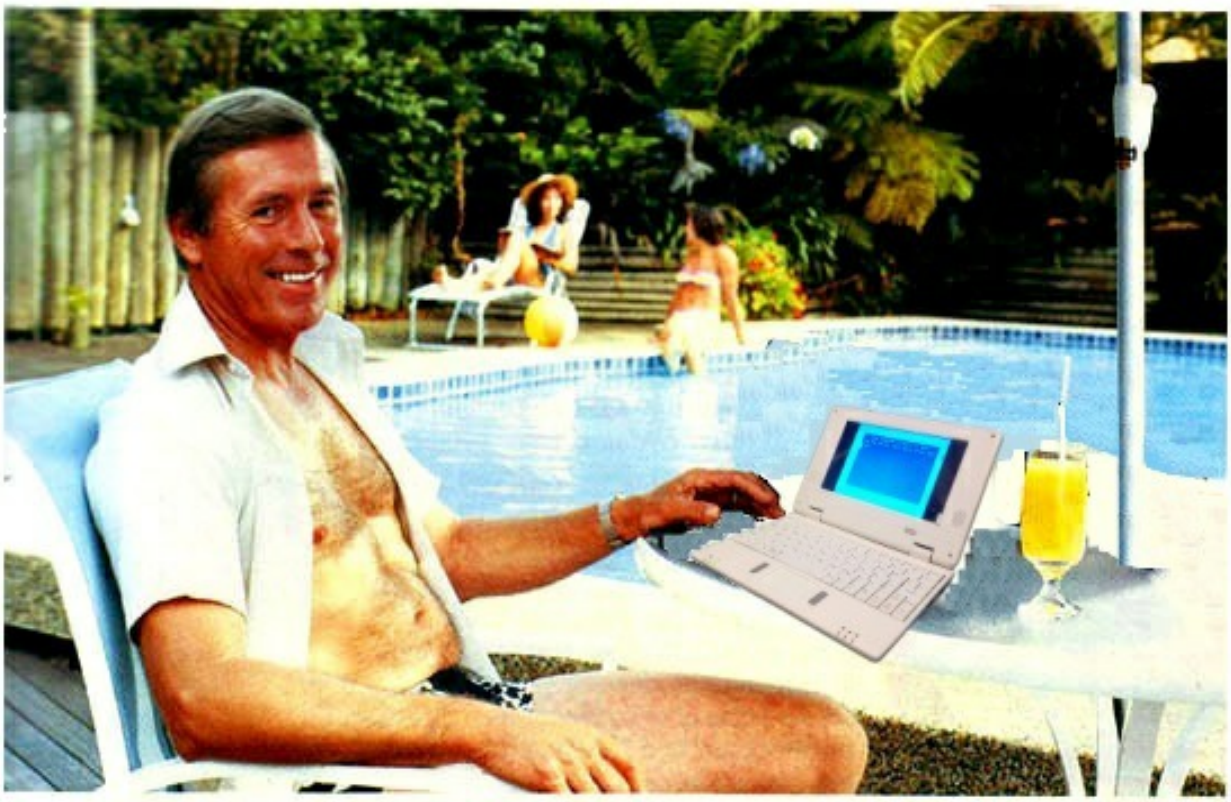
RETRO ASYLUM PODCASTS

Retro Asylum is a English podcast about retro computing. In this, the career of Ben Daglish in Issue 76.

Actually, at the time of writing the podcasts have reached number 79, and this issue contains an interview with graphic artist Stoo Cambridge. For more information head over to

http://retroasylum.com/





Who's keeping up with Commodore?

C64p

SD2IEC

commodore COMPUTER

Keeping up with you.

www.thefuturewas8bit.co.uk

Not just a place to buy SD2IEC's or a C64p

The future really was 8 bit!

Commodore 64 and 128 News

MELON 64

As *Lemon64* seems to be "offline" at this moment, another forum has struck up, quite creatively called *Melon64.* It appears to be like the Lemon forum but doesn't give a bitter aftertaste in your mouth, just the sweet juice of fructose. It's early days but seems to be attracting a number of subscribers, who are friendly and helpful. See what you think.

Here is the response from Melon64's admin:

Hi,
Melon64 was set up initially to provide a temporary home for Lemon users who where left without much information when Lemon's latest downtime spanned 3 weeks. This was the second time in the space of six months that Lemon had disappeared.

I made the decision to continue Melon64 as there is always space for another Commodore 64 forum.

Over the coming months we'll be adding more content and functionality to Melon64. A collection database? File repositories? User galleries? Constructive feedback is always welcome.

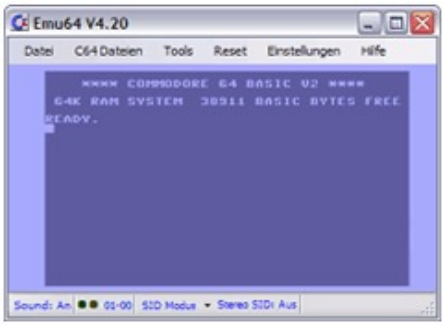
It takes time to grow any community, and it'll take time to grow Melon64.
http://www.melon64.com/forum/index.php

Emu64 V5.0.9 RELEASED

Thorsten Kattanek has released a new (beta) version of his Commodore C64 emulator called *Emu64*. Some of The recent changes are:

-Android version.
-Improvements for the SID, VIC, CIA and the CPU emulation.
-REU and GEORAM emulation is added.

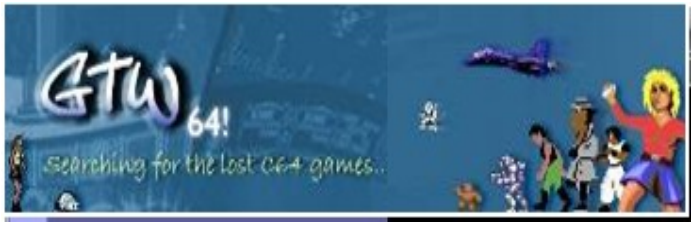
https://bitbucket.org/tkattanek/emu64/wiki/Home



GAMES THAT WEREN'T 64 UPDATED

The *GTW64* web page has been updated. Toki V1 and Yie Ar Kung Fu V1. Updates: Ballfever, Bugs Bunny, Darksyde, Here and there with the Mr Men, Lethal Xcess, Spellcast, Star Tech Games and Your Computer Software Exchange.

http://www.gamesthatwerent.com/gtw64/



CSAM SUPER

CSAM is a Windows application for converting images and video it into a Commodore 64 format. The program analyses the original picture and will make a codebook (2 KByte) and screen-data (1 KByte). With this data you can display your image on the C64. This version uses a new algorithm for a better end result.

http://csdb.dk/release/?id=127248

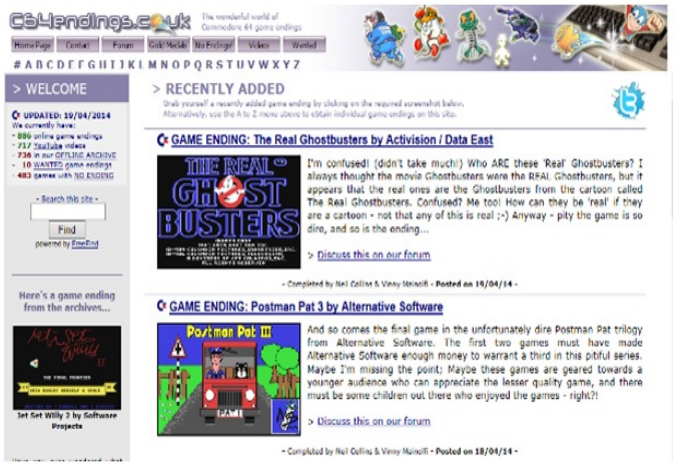
CROWD FUNDED MIDI INTERFACE

I was recently contacted by Frank Buss who wanted to plug his new hardware project. It's a crowd-funded project to create a new MIDI interface Cartridge, but this one will also have an integrated flash memory chip to transfer files to and from a PC or Mac computer. To read more head over to this website, although it appears to be In German language.

http://www.startnext.de/kerberos

C64 ENDINGS UPDATED

The web page c64endings.co.uk has added new endings of Commodore C64 games. The most recent additions are: Harbour Attack (CBM Inc.), Ikari III - The Rescue (SNK), Jackal (Konami), Kinetik (Firebird), Legend of The Amazon Women (Silvertime), Ninja Warriors (Virgin Mastertronic), Oops! (The Big Apple), PSI-DROID (Zeppelin Games), Race Against Time (Codemasters), Shark (Players Premier), Santa Claus' Helper (Santa Claus' Helper), Scarper! (MC Lothlorien), Sabotage (Zeppelin Games), Taskforce (Players Premier), T-Bird (Virgin Mastertronic) and Trans-Atlantic Balloon Challenge (Virgin Games).



http://www.c64endings.co.uk/

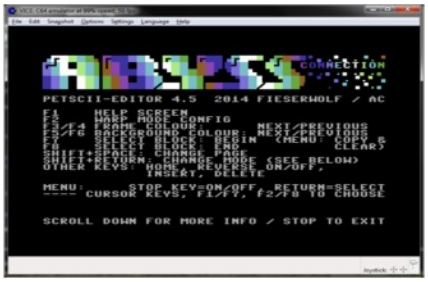
DUREXFORTH VERSION 1.3 RELEASED

This is a modern, lean C64 Forth-inspired by colorForth, JONESFORTH and Blazin' Forth. Direct threaded for simplicity. The project includes a vi clone written in Forth, a high-resolution graphics library, plus MML music support.

http://code.google.com/p/durexfort

Petscii Editor v4.5 RELEASED

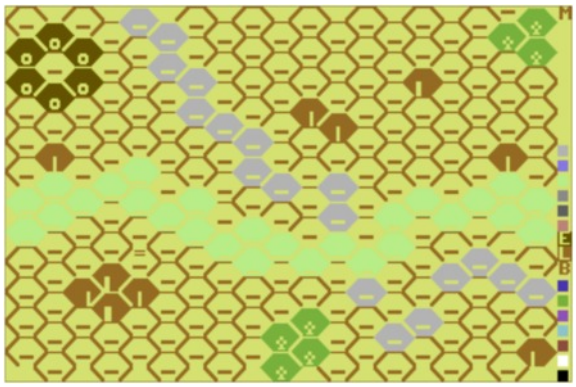
Released by: fieserWolf, of Abyss Connection
A petscii graphics editor package With new features like joystick + 4x4 mode, two pages (buffers), adjustable keypress speeds, loadable music, and more



http://csdb.dk/release/?id=130214

Hexmapper - C128

Raoul has a hex-mapper developed for the Commodore C128. Using this program you can create mosaics, with a hexagon as a unit, make maps for role-playing games, or just cool pictures. With the program, you can adjust each hex with a direction and a colour. You can also move or copy the hexagons. A manual (in English) and some examples are available.



http://raoulm.home.xs4all.nl/products/

Also while you're on the site you may like to look at some of his other software for the VIC-20 like

Denial Scroll for unexpanded VIC-20
Poxeldemo for unexpanded VIC-20
Poxelshow for unexpanded VIC-20

The program names are self-explanatory. I quite like the Denial scroll on the VIC.

He has some other software on the site as well a Jav Mandelbrot generator set and some Android apps

C-ONE CORE RELEASED

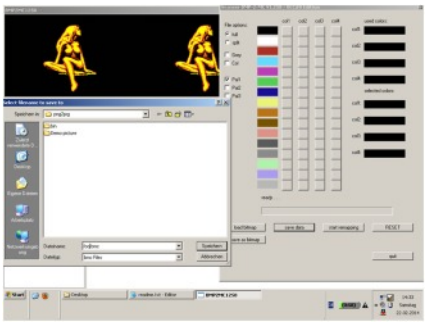
Peter Wendrich, after 4 years of inactivity, released a new "preview" Chameleon core for the C-One.

http://syntiac.com/c_one.html

BMP2MC. UPDATE RELEASED

Seanser has released a new version of *BMP2MC*. *BMP2MC* a bitmap to C64-multicolour conversion program with five colours. The properties are: Gray or Colour (there are three different colour tables available). Full (a file) or Split (bitmap, colour, characters). Tables for remapping colours (PAL1 & PAL2).

http://csdb.dk/getinternalfile.php/128119/BMP2MC1258_OK.zip



THE ACE TEAM NEEDS HELP

The Ace have been working on a new operating system for the Commodore C128. Sadly, the "team" are now down to only one person. So, if you are a C128 enthusiast and want to help with the development, testing or give suggestions, please contact Miro from The Ace team.

http://www.theace.sk/blog/index.php

C64 RELOADED: NEW C64 MAINBOARDS

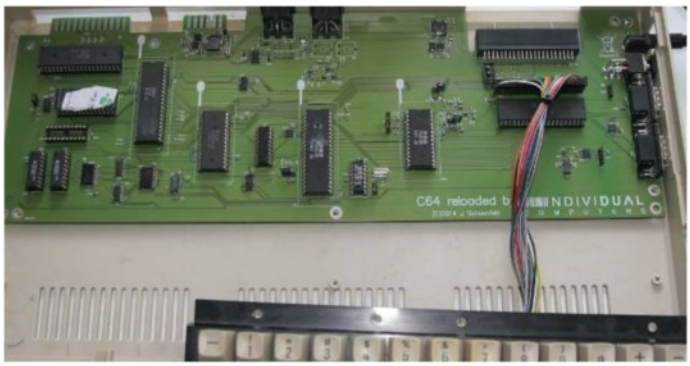
Jens Schönfeld announced a replacement C64 motherboard. Although it was seen as an April fool, it would appear after a fellow reader contacted Jens the project is real!

Jens says that with the age of the Commodore 64 and the high costs to repair a motherboard, he will produce replacement boards as close to the original Commodore schematics as possible.

The project is called ***C64 Reloaded*** and the boards will fit in the original Commodore 64 cases, and feature not only a modulator, but also an S-Video and a 3.5mm audio output. The board can also be jumper-set to output PAL or NTSC, and comes partially assembled. Basically you will need a donor machine to transplant the chips into the ZIF sockets on the *C64 Reloaded* mother board.

C64 Reloaded will be available in limited quantity starting July, 2014 for 149,90 EUR(**). We will start taking pre-orders soon.

http://icomp.de/home/indexe_news.htm



The Impossible Game RELEASED FOR THE C64

a csixx and mayday! co-production
Code: steve ody(csixx)
Music: spider jerusalem
Title gfx: achim volkers
Testing: the ryk

The idea of the game is to navigate the obstacle course, avoiding pits, spikes, and jumping on/over boxes. You do this by pressing the Fire button on the joystick or pushing the joystick upwards. Basically it's a scrolling landscape and you have to jump over things, sort of like a electronic arty-techno version of *Flappy Bird*.

This is the c64 version:
http://csdb.dk/release/?id=129903
http://flukedude.com/theimpossiblegame/

The website says:
A super-addictive and very, very hard platform game, synced to an awesome soundtrack

C64 Power adaptor

MAD Scientist has created a power adaptor for the Commodore C64. He uses a standard power supply from the PC with a Molex connector that has 5 VDC and 12 VDC. His adaptor makes 9 VAC from the 12 VDC and together with the 5 VDC it is connected to the Commodore C64. You can follow the progress on the Forum64.de web page.

http://www.forum64.de/wbb3/board65-neue-hardware/board289-diverses/55283-neues-c64-netzteil-mark-1/

CCS64 v3.9.1 RELEASED

Per Håkan Sundell released a new version of his Commodore C64 emulator. Changes in this version: emulator improvements to support more demos such as EmuSuxx0r from Crest. More PAL artefacts such as blurring, etc., have been implemented. You can download this C64 emulator from Per's web page.

http://www.ccs64.com/

SD-BOX Cartridge v1.09

The *SD-BOX* is a cartridge for the Commodore C64 and features an SD card interface. The cartridge has many features, acting as a disk drive and Datassette. Recent changes: Support for Micrus Copy program, A HEX forecast for the cassette buffer, and a program to remove copy protection. Improvements for loading / verifying of cassettes, and the manual has also been updated.

http://c64.com.pl/index.php/sdbox109.html

Creatures Cartridge - C64

Siem Appelman has a download of the game *Creatures* as a cartridge image. Also on the site is a download of a CRT for the game *Mayhem in Monsterland*.

http://www.siemappelman.nl/download.html

DAVID FOX INTERVIEW

The C64.com web page has an interview with David Fox. David started to work with computers in 1964. When home computers became popular he started making conversions of adventure games. He created the game *Mix and Match Muppets*. You can read the whole interview on the C64.com web page.

http://www.c64.com/

High Voltage Sid Collection (HVSC) updated

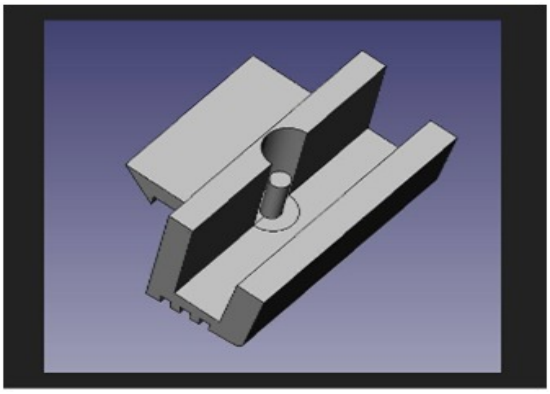
There is a update available of the *High Voltage SID Collection*. There are now more then 44.000 SIDs in the collection. In this update 825 new SIDs, 233 fixed/better rips, 910 SID credit fixes, 120 SID model/clock infos, 8 tunes identified, and 67 tunes moved. You can download the update from the HVSC web page.

http://www.hvsc.de/

Reproduction of the SX64 parts

Erwin van Betten has successfully reproduced the handle-bar caps and the keyboard clips for the Commodore SX-64. Erwin uses an CAD program to design the parts and a 3D printer to make the parts. The parts are printed with an LulzBot TAZ 3D printer equipped with a Budaschnozzle 2.0 w / 0.35mm nozzle.

http://c64.berrydejager.com/reproduction-of-the-sx64-parts-by-erwin-van-betten

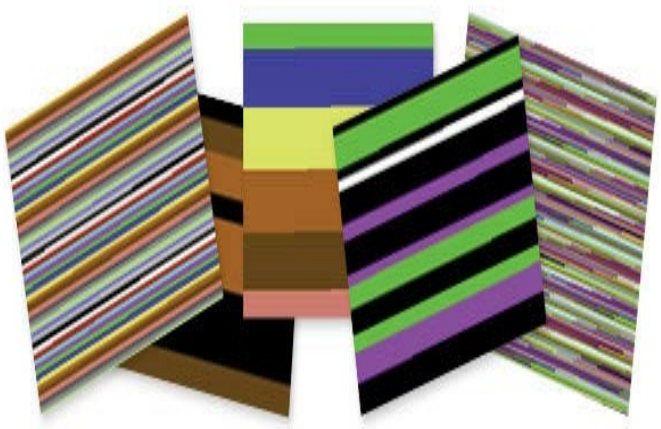


ACID 64 Player Pro v3.5

Wilfred Bos has released a new version of *ACID 64 Player Pro*. Changes in this version: Improvements: Folder management for the HVSC, MUS files, layout, first file played is now always added to history list and other small improvements.

www.acid64.com/

A BOOK THAT CELEBRATES THE BEAUTY OF THE GREATEST HOME COMPUTER EVER MADE; THE COMMODORE 64.
With help via RGCD who have produced an exclusive of micro hexen on Cartridge to sell to help fund the book



The Kickstarter campaign says...
Introduction

Hello! This campaign is to hopefully produce a new exciting book about the Commodore 64. Unlike other books about the C64, it will celebrate the visual side of the computer. Each spread will feature a beautiful image and a few words. This could be a few games facts, a mini review or even a quote from the developer. This will be the first book by new publisher Bitmap Books who specialise in high-end books all about computer games.

https://www.kickstarter.com/projects/2146199819/commodore-64-a-visual-commpendium-by-bitmap-books?ref=live

Contributors

A huge thanks to everyone who has agreed to contribute to this book. Here is a list of who is involved in various ways!
- Stoo Cambridge (Sensible Software)
- Robin Hogg (Zzap!64)
- James Monkman (RGCD)
- Jason 'Kenz' Mackenzie (Psytronik)
- Paul Koller
- Matt Wilsher
- Andy Roberts (Comodore Format)
- Benjamin Wimmer (c64screenshots.com)
- Jonathan Leung (VGMaps.com)

...and not forgetting Tim Nicholls for giving up his own time offering me some great advice and some invaluable tips.


The Adventure Continues In:
REALMS OF QUEST IV
A Ghislain De Blois Production
Featuring:
•10 Dungeon Levels which have been designed by hand with numerous rooms, secret passages, riddles and traps to traverse and overcome.
•10 races and 12 classes to choose from to create 8 players for your party.
•You will meet other heroes to help and guide you during your quest.
•Over 100 unique graphical portraits that depict the various monsters that you'll encounter as you travel throughout the Dungeon.
•Over 135 types of equipment to represent weapons, armor, gems, jewelry, potions, scrolls and miscellaneous magical items.
•60 magic spells that you can cast.
•3 game fonts to choose from.
•A 16 page printed instruction manual is also included.
THREE full adventures for the VIC-20 along with Ultimate Quest :Catacomb for the Commodore 64!
PSYTRONIK SOFTWARE
VIC-20
www.psytronik.net


ONE COP. ONE DRUG LORD. NO MERCY
THE VICE SQUAD
A C64 Game by Achim Volkers & Trevor Storey
GAME FEATURES:
• Multiple stages to battle through! • Superb varied backgrounds!
• Awesome weaponry! • Mission briefings! • Boss battles!
• Funky Soundtracks! • Addictive arcade style gameplay!
PSYTRONIK SOFTWARE
www.psytronik.net
www.rgcd.co.uk
NOW AVAILABLE ON C64 TAPE, DISK & DIGITAL DOWNLOAD FROM WWW.PSYTRONIK.NET AND ON C64 CARTRIDGE FROM WWW.RGCD.CO.UK


Trevor Storey and Georg Rottensteiner Present:
SOULLESS
An Epic Arcade-Adventure For The Commodore 64!
GAME FEATURES:
Massive Map To Explore . Stunning Animation
Searchable Objects . Atmospheric Soundtracks
Detailed Background Graphics . Magic Effects
Separate Animated Intro + End Sequences
Spawn Points . Spirit Stone Puzzle
A3 Poster/Map . Companion CD
This most ambitious release from Psytronik yet! A superb arcade-adventure featuring sumptuous packaging. An unmissable C64 release!
UNLEASH THE BEAST FROM YOUR C64!
Now Available For C64 On Tape, Disk, Cartridge + Digital Download
PSYTRONIK SOFTWARE
www.psytronik.net
www.rgcd.co.uk


SIX C64 GAMES IN AN EXPLOSIVE NEW COMPILATION!
SHOOT EM UP 3
DESTRUCTION SET
2014 C64 RELEASE!
AN ALF YNGVE PRODUCTION!
Screen pictures shown from Commodore 64 version of games.
MUSIC & ENHANCEMENTS BY RICHARD BAYLISS
LOADING BITMAPS BY CARL MASON • COVER ART BY OLIVER FREY
PSYTRONIK SOFTWARE
www.psytronik.net
NOW AVAILABLE ON C64 TAPE, DISK & DIGITAL DOWNLOAD!

Vic and Commodore Plus 4 News

Hires Color 8 plus 4 images

Erich/Unlimited has released a new picture diskette called *Hires Color 8*. You will find 32 images on the two disk sides, and all images use the hi-res graphics mode. The picture show is made with the *Magica* program.

http://plus4world.powweb.com/software/Hires_Color_8

GET THE CAT FOR THE UNEXPANDED VIC-20

Peter van der Woude has released a game called *Get the Cat* for the Unexpanded VIC-20 (joystick required).The game's description says it's... A simple game: Step on the bricks to save your cat.

You can download the file here:
https://drive.google.com/file/d/0Bz1hF7VZSV-UdGx1dk1maXI4cmc/edit?usp=sharing

In the game you move left and right with the joystick. Press Fire to place a brick under your character, provided you have bricks left. Your character can step up one brick only. You can let him fall from any height. The bricks seem to fall randomly, but there is a pattern to make the game easier. If you reach level 5,10,15,... you gain an extra life. Maximum extra lives is only 1. It gets easier when you know every fifth block falls directly above you. Very useful for building stairs to the cat.

Thread: http://sleepingelephant.com/ipw-web/bulletin/bb/viewtopic.php?f=10&t=6942

NEW VIC-20 MULTICART/DEVELOPMENT CART

On Wed., Dec 09, 2009, Robert wrote:
Kent Rittenhouse has produced a VIC-20 Multicart / Development Cart with 32 games, games like Pac Man, Donkey Kong, Frogger, Dig Dug, etc.. The price is $28 for the complete cart (not including shipping). (Snip)

Kent Rittenhouse has now released game set 2 of the VIC-20 Multicart/Development Cart. The 32 games include such games as Artillery Duel, Cannonball Blitz, Lunar Leeper, Mountain King, Pharaoh's Curse, Satellites and Meteorites, and more.
The cart price is still the same. For more information and to see the complete list of games in game set 2, go to:

http://www.gamingenterprisesinc.com/vic20

Game set 1 is still available, too.

Truly, Robert Bernardo
Fresno Commodore User Group
http://videocam.net.au/fcug

FLINALE SLIDESHOW OF NEW VIC GRAPHICS MODES
tokra has released a piece of software called Flinale
The software's requirements are an **NTSC VIC-20** with 32K RAM expansion and/or **PAL VIC-20** with 24K RAM expansion

The program is a slide show for two newly created VIC-20 graphic modes:

For NTSC-VIC 20: IFLI88 (88 x 400 interlace with 8 x 1 colour res)
For PAL-VIC 20: FLI104 (104 x 256 with 8 x 1 colour res)

Download: **http://www.tokra.de/vic/flinale/flinale.zip**

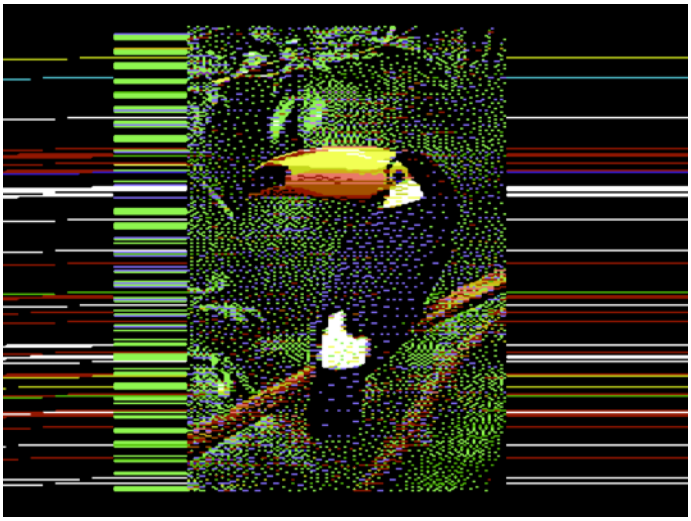
NTSC YouTube
http://www.youtube.com/watch?v=LLI058aHmEg

PAL YouTube
http://www.youtube.com/watch?v=3Hch8cFJqDs

pouet.net-Entry
https://www.pouet.net/prod.php?which=62921

NTSC-Discussion HERE
http://sleepingelephant.com/ipw-web/bulletin/bb/viewtopic.php?t=6951

PAL-Discussion HERE
http://sleepingelephant.com/ipw-web/bulletin/bb/viewtopic.php?t=6569



TWEETING ON A VIC 20
http://www.torontosun.com/2014/02/20/tweeting-on-a-commodore-vic-20

Unbelievable but true, the Toronto Sun newspaper looks back at the first tweets from the VIC-20 computer. Check out the link to read more on this story.

This is Richard Beales tweeting from the VIC-20 at the Personal Computer Museum in Brantford, Ontario, Canada.
http://www.pcmuseum.ca

Amiga News

ARMIGA PROTOTYPE

I was asked to plug this again so ……..
Here is a new IndieGoGo campaign to revive the old Amiga 500 feeling http://igg.me/at/armigaproject/x/6542614

The base *Armiga* aims to emulate the original Amiga 500 with 1MB of RAM as close as possible to the original one. **And for you not to get your hands dirty, a fully legal copy of Kickstart 1.3 is provided with every Armiga!** :O

Specs:

- Powerful **Dual Core ARM CPU**.
- **2 USB host:** Joystick, mouse and keyboard support, as well as pendrives/hdds.
- **SD card slot:** Save your ADFs or bring new ones!
- **Ethernet connection:** Connect to your network for easy ADF management.
- **HDMI:** Digital AV quality on the big screen!.

Features:

We really want this to be an awesome product!. But for that we'll need your help. Lots of functionalities are in the back-log, waiting for the needed funds. However, **your Armiga will come with these features built in:**

- **Full Amiga 500 emulation:** The target machine is the iconic Amiga 500 and right now 90% of the disks are running!
- **Boot to Android:** Armiga supports Dual Boot and comes loaded with Android 4.2.2, so when you're not playing you can have all the power of Android!.
- **Automatic disk load:** Like in the original Amiga; insert the disk and off you go!.
- **Disk dump:** Create ADF images of your favorite games and keep them safe on the SD.
- **ADF support:** Bring your own ADF images on a pendrive or SD card and enjoy!
- **Disk swap:** Just insert the disk and it will be dumped. When time comes to change disks, just select the right ADF. Ain't it easy?.
- **FTP server:** No need to take the SD off to manage the ADFs; just do it from your computer!.
- **Graphical menu:** Simple and elegant, with usability as main focus.
- **Game save:** Save your game and resume later.
- **Autosave:** Forgot to save?. We do it for you!
- **Screenshot:** Wanna share your joy?. Give our screenshot feature a try

BOINGSWORLD PODCAST #50 RELEASED

The 50th Edition of Boing World has been released
Sadly, even Google Translate couldn't help me with the contents. Anyway, it has been released and you now know about it.

http://boingsworld.de/

Interview with Armin Sander (Oktalyzer)

The magazine Obligement has published an intervi with Armin Sander, the German coder behind the famous 8-tracks tracker Oktalyzer on Amiga
Interview in English :
http://obligement.free.fr/articles_t...wsander_en.php
Interview in French :
http://obligement.free.fr/articles/itwsander.php

DIGIBOOSTER PLUGIN FOR HOLLYWOOD

Airsoft Softwair announced a DigiBooster plugin for Hollywood. After installing the plugin Hollywood will automagically be able to play DigiBooster modules in 44.1khz 16-bit stereo.
You can download the plugin from the official Hollywood portal.
http://www.hollywood-mal.com/

Thanks to Hollywood 5's cross-platform plug-in system versions for AmigaOS3 (Classic), AmigaOS3 (FPU), AmigaOS4, MorphOS, WarpOS, AROS (Intel), Linux (PowerPC), Linux (Intel), Mac OS (PowerPC), Mac OS (Intel), Windows and Google's Android platform are also provided.
For more information on DigiBooster and how to order the software, visit the official DigiBooster site.

http://www.digibooster.de/en/index.php

Vampire 600 ACCELERATOR

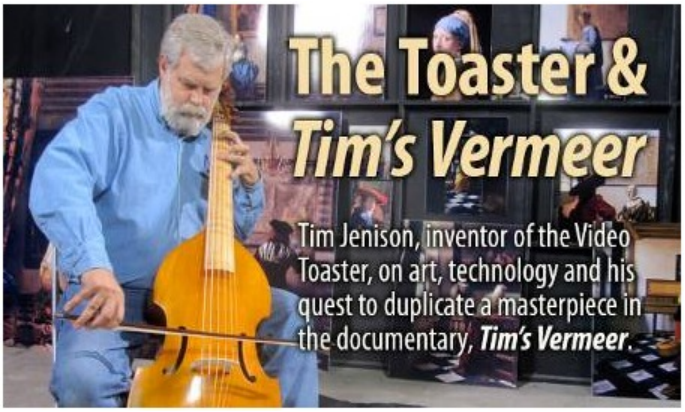
The *Vampire 600 FPGA Accelerator* for the Amiga 600. The *Vampire 600* has been developed by Majsta and has recently become available. On mfilos blog you can read about installing and configuring the new accelerator.

http://www.mfilos.com/2014/01/a600-vampire-600-new-toy-in-town.html

THE TOASTER AND TIM'S VERMEER

The inventor of the *NewTek Video Toaster* was certain that he figured out the secret behind the uncanny realism of one of the world's greatest painters, and certain he could use the same methods to duplicate it. Read a conversation with NewTek founder Tim Jenison on the unexpected intersection of art, technology, obsession, and the Video Toaster in the wonderfully provocative documentary called Tim's Vermeer.

Picture taken from the website creativecow.net
http://library.creativecow.net/wilson_tim/Tims-Vermeer_documentary/1



ANTIRYAD GX 3.3 NOW FOR MORPHOS AND AROS ARM

Antiryad Gx v3.3 was released. This version drop the professional license price.

Here is the list of new features:
- Optimized thread management.
- Added ETC1 texture (de)compressor
- Support of Amstrad CPC SCR file format (without Amsdos headers).
- Support of OCS and AGA shipsets in Amiga 68k driver using new chunky to planar system.
- Support of AHI sound system in Amiga 68k driver.
- Enhanced gx_baseed, gx_dsp, gx_filevirtual, gx_interface, gx_keyboard, gx_obj3d, gx_render, gx_scratchbuffer, gx_soundmixer, gx_winbox objects.
- Optimized OpenGL 4 dynamic rendering.
- Optimized OpenGL 2d flush.
- Support of NACL platform.
- Support of MorphOS PowerPC platform.
- Support of Linux ARM platform (Raspberry PI).
- Support of AROS ARM platform (Raspberry PI).
- Added new keyboard virtual keys.
- Added input tester tool.
- Enhanced video codec.
- New antisector system.
- Optimized gx_math object.
- Added fixed point functions in gx_math object.
- Optimized 2d rendering (gx_screen, gx_bitmap objects).
- Support of Amiga SVX sound format.
- Enhanced bitmap MTR reader and writer, now support bi-planes modes.
- Added high quality IFF bitmap writer and enhanced readed, supporting PC chunky, Amiga planar, Halfbrite, HAM6 and HAM8 modes.
- Added support of Atari ST Degas (PI1, PI2, PI3) and Neochrome (NEO) bitmap formats.
- Enhanced benchmark.
- Fullscreen mode switch with F11 key instead of ALT+ENTER.
- Enhanced Winbox main menu.
- Removed music Gel module, a new music system is now embedded in Antiryad Gx.
- Added http downloader.
- Added new Winbox themes.

http://www.arkham-development.com/

AROS Vision 2.4 uploaded

http://www.amiga.org/forums/showthread.php?t=66988&goto=newpost

Improvements:
-Reworked/Optimized Icons
-Different modes for different directories (name or icon-mode) to improve handling
-Freeware Raytracers added
-New filetypes added (YAFA and many different module types)
-existing Filetypes improved
-special version of AppStore added (indieGO Marketplace text client)
-big number of small improvements, f.e. a number of GUI Toolkits added)

Planned for next future version:
improve integrated developer environments
adaptions to real hardware (like changing icon set)
make use of AREXX ports of the different applications
adding own small components

Redit – WORD PROCESSOR

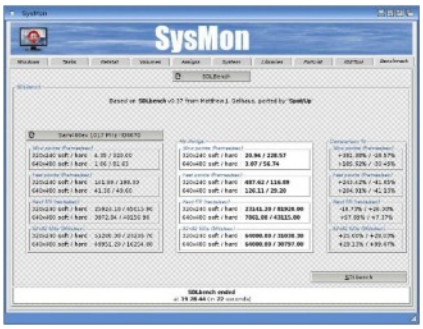
Redit is a word processor for the Amiga computer. It can be run from Kickstart 1.2 and 0,5 Mbyte upwards.

Changes in this version:
-Create documents with the CLI interface.
-The tab size and colours are configurable.
-Status bar for cursor line and column.
-Switching between documents with a hotkey.

http://www.kaiiv.de/redit/de/

SysMon - Amiga Uodated

Sysmon is a system monitor for AmigaOS 4 created by Guillaume Boesel. Recent changes to the program include :
Tooltype to deactivate the ShortHelp.
Benchmark frames are now resizable.
Improved the Picasso function
updated the Italian translation.



http://www.os4depot.net/?function=showfile&file=utility/workbench/sysmon.lha

---

DAVE HAYNIE TALKS ABOUT DEVELOPING THE COMMODORE AMIGA

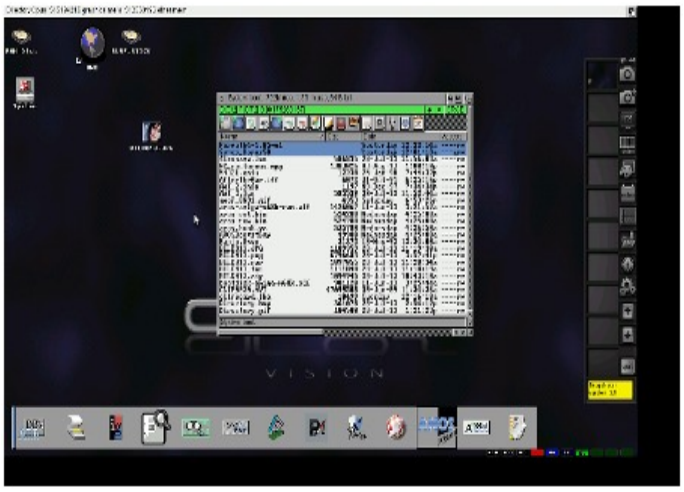
News from Fran Blanche on YouTube:

Dave Haynie talks at VCF East on April 6, 2014 about developing the various Amiga systems, up into the last days of Commodore in April 1994. Introduction by Bill Herd. This was a fascinating hour of must-hear stories for any serious Commodore fan. Dave even wears his Commodore Death-Bed Vigil shirt!

https://www.youtube.com/watch?v=Rcr2CFV0T4I

AROS VISION 2.5

A new version of *Aros Vision* has been released including additional web software, games, tools, and improvements regarding the included development software.



Highlights:
-Poseidon
-ImageFX 1.5
-AmiBlitz 3.6 (new snapshot)
-Additional Amiga-E compilers
-Trog AGA
-Additional SDL-Games
-Ignition fully working
-ViewCSV
-MIDI-File support

Download Page:
http://www.aros-platform.de/html/distribution_download.html#_blank

---

TAWS V0.23 (THE AMIGA WORKBENCH SIMULATION)

TAWS (The Amiga Workbench Simulation) is a JavaScript simulation of the Amiga Workbench 1.x - 3.x for Internet Explorer, Firefox, Opera, and WebKit browsers. With TAWS you can work with the Amiga Workbench inside your favourite web browser. Changes in this version:
-Added OS 1.0 and OS 1.1. OS 3.9: Start-up screen
-AsyncWB and Trashcan. OS 4
-More short keys and menu option
-Early startup control and startup screen

And many improvements for better emulation in IE, Firefox, Opera, Chrome, and OWB.

http://www.taws.ch/WB.html


www.melon64.com
Sweeter than lemons!
www.melon64.com

AMIGA FOREVER AND COMMODORE 64 FOREVER

Amiga Forever

http://www.amigaforever.com

http://www.facebook.com/AmigaForever

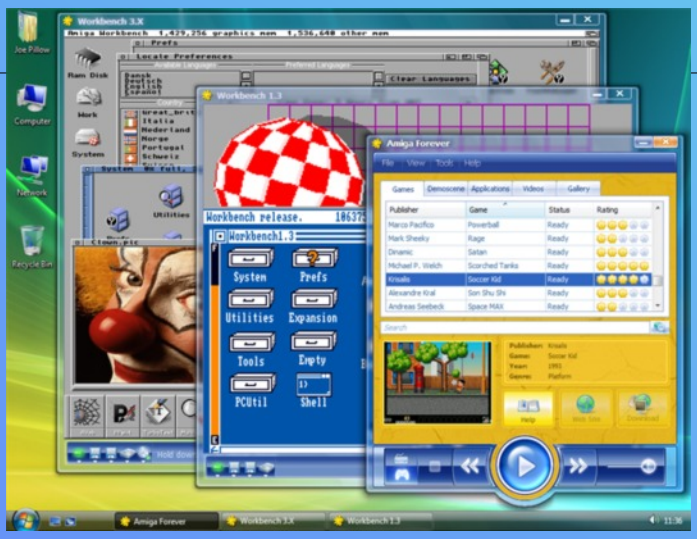
C64 Forever

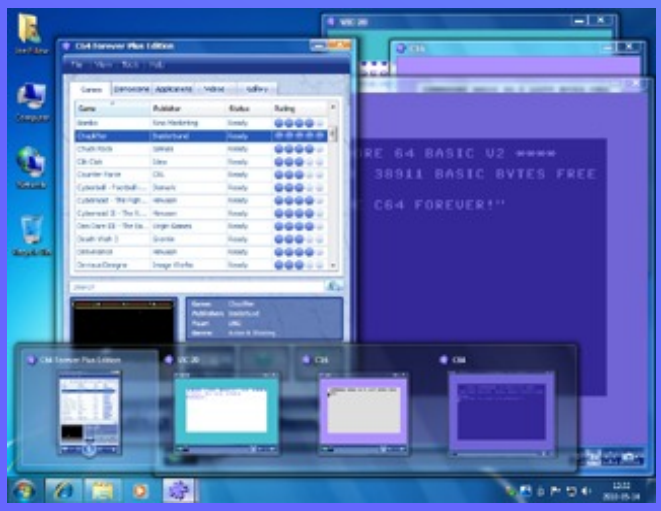
http://www.c64forever.com

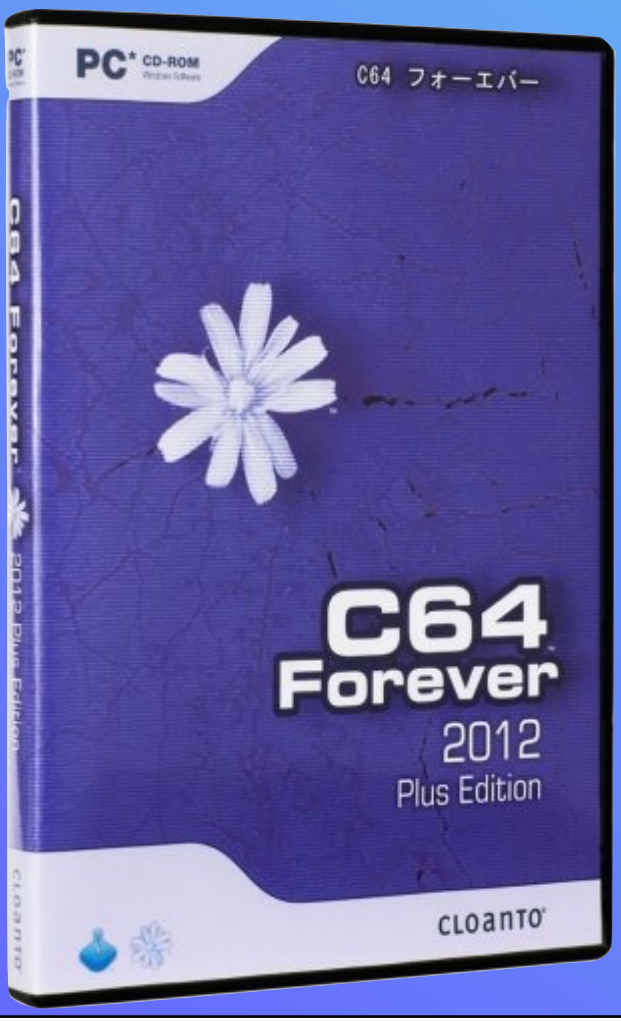
http://www.facebook.com/C64Forever

RetroPlatform Project

http://www.retroplatform.com







COMMODORE FREE INTERVIEW WITH PIXEL CREATOR OF PULSE FOR THE UNEXPANDED VIC 20

Pulse was created by pixel and is a horizontal smooth-scrolling shoot-'em-up, inspired by "Gradius"
The game works on an Unexpanded VIC-20 with a Joystick. The game was reviewed in Commodore Free magazine (Issue 79). If you missed the review I suggest you download and read it. You will find the game and source code available for free download from here:

Program file:
https://github.com/SvenMichaelKlose/pulse/blob/master/pulse.prg?raw=true

Source code:
https://github.com/SvenMichaelKlose/pulse

Q. Hi pixel! Thanks for agreeing to the interview. Please, can you introduce yourself to the Commodore Free readers?

Hi there! My name is Sven Michael Klose. I'm a 39 year-old and have been teaching myself programming since '85. I live in Berlin Friedrichshain (Germany) and I share an apartment with a flatmate and his dog, which keeps driving us nuts. I love running around in embarrassing outfits, playing the saxophone with bands on jam sessions, and of course, hanging out with my beer groups and get hammered. I'm certainly into entertaining people (when I'm away from the computer).

Q. So then, what was the motivation for creating *Pulse*, and especially, why did you created the game for the unexpanded machine?

I always wanted to do something with a 6502 CPU, and I always wanted to write a game. The stock VIC-20 was my first computer. I wondered how much one could squeeze out of the machine with the amount of programming experience people have nowadays. I was depressed out of my mind after my software business didn't take off, so there was a serious need for quality time in front of the computer with the door shut – no socializing. It was a perfect mental holiday.

Q. I criticised the game in Commodore Free (although it did receive a relatively high score) on some of the graphical glitches in the game, flickering and ghosted images, etc. Was this purely down to the hardware and limited resources, and of course, the speed the game runs at, or was it more down to not being able to resolve the problems with your code?

The ghost ship that occurs after the player was hit and the sprites just popping out on the left instead of clipping at the border are bugs; I am afraid I just couldn't find them. Shame on me. I guess I spent far too much time on it. The other glitches cannot be removed from the unexpanded VIC, as far as I can figure it out. The graphics are text characters really, and that always comes with clashes, especially if you intend to move things smoothly. The sprites are double-buffered to make them bearable. When the game runs out of its 48 characters for its 16 sprites, it just picks the last character used, and that's when you get the mirror images. Pulse is dancing the Tango in a phone cell.

Q. When the game is running of course, the small glitches fade into obscurity, especially as the game really rattles along. You mentioned the VIC is absolutely flat-out (processor-wise), so with that in mind I presume the sections and waves had to be very carefully planned out so as to limit the amount of screen graphics. My question is, "Was this planned – or did the game just evolve over time?"

Ha-ha! That's a good one! Me and careful planning... Just kidding. No, that was much easier. Since there's no way to race the beam, Pulse doesn't wait for the retrace. Instead, sprites aren't really removed, but are turned into background star sprites to keep the speed more or less stable. Your VIC doesn't waste a single CPU cycle while running Pulse. There are 16 sprites all the time. Even those behind the background are drawn into ROM. The game speed still varies a lot but no-one seems to notice or at least complain about it. The game grew in small steps, and after each step the next became more or less obvious. I played around with the screws and trusted my intuition, and of course my fond memories of Gradius/Nemesis on the first Nintendo Entertainment System. In the end of the day it was supposed to be an action game – and it's totally O.K. for it to be a little bit stressing. Right before I started programming Pulse I dug through 6502 CPU documentation and wrote a disassembler with the tiniest CPU description possible (in my small universe) in Lisp. That's what probably made me dangerous.

Q. Do you intend any further developments on the game – maybe a loading screen with music and a high score table? Obviously the loading screen would be do-able, but is there any memory left for anything else?

I'd love to see a real tape release. Of course you simply cannot have no loading screen on tape, or can you? Unfortunately no retro software house was interested. There are 50 bytes left now scattered across memory. Maybe it's possible to put the letters "HI" in front of the high score, but that's all I can picture at the moment. Or maybe a little bit more level data. Uh!? Why didn't I think about that earlier?

Q. As the source is available to download, I expect people will want to tweak it, and I presume this is your intention for its release. Would you consider re-working the game if a reader could remove more of the screen glitches?

The source code is simply a gift to the community. The game is very hard to tweak and I'm trying to leave my hands off it myself because you can break things very easily. Everything is kind of interlocking. If somebody comes up with fixes, I'll surely help. Don't forget – the game is public domain. You can do with it what you want. You can make a tape release without asking me. You can re-use any code for your own project without having to slap my name on it - you're probably running out of memory anyway. Readers with questions about the source code shouldn't hesitate to ask me. My address is pixel@hugbox.org

Q. Of course I am mentioning the negatives of the game, and remember the game scored a very respectable 8.5 out of 10, so anyone reading should not be alarmed by the talk of glitches. Were you pleased with the score, and do you have any comments you would like to mention about the values I awarded the game?

I know next to nothing about VIC games and I hardly play any games at all, except the famous Midlife Crisis. I leave it to you as the expert. I'm super happy with your ratings! All technical things aside, it's up to the gamer who is supposed to spend his precious time to rate the game. I couldn't ask for more.

Q. So this is your first VIC game?

Yes, Pulse is my very first game.

Q Do you have any other VIC games planned?

Yes, and it's spoiling my days and nights! I'm making an idiot out of myself trying to create a 3D tank game with filled polygons for the unexpanded VIC. You can observe my progress-in-failing on Github.

Q. Do you prefer working on the standard hardware, or would you consider creating games for expanded machines?

A nice thing about programming for unexpanded machines is the prospect of actually getting something done, and the technical challenges of it suit me well. I'm afraid all I could come up with on an expanded machine is a crap game. I'm just not ready yet – it has to feel right.

Q. Do you work on other Commodore platforms, and will the game be ported to other Commodore hardware?

No, I never worked on other Commodore machines. My parents replaced my VIC by an Amstrad CPC-464 back then, and that one by an early IBM-PC. Grrr! Life can be cruel. Pulse is just right for the VIC. The C64 community would probably have a good laugh about a port, wouldn't they? Maybe the C16 is worth a try. If some retro software house wants a port to another machine I'm all ears. I know there's hardly any money in it.

Q. Some readers have asked me to ask you how the game was created, the tools used, and how the bug testing and coding was worked on. Also, did you have other people testing the game besides yourself?

Linux Mint runs my laptop computer. I used the VICE emulator, a 6502 assembler called 'xa', the VIM text editor, and meditative debugging. I made tiny changes, assembled everything, and checked if it worked. When it did the change went into the Git repository. No debugger involved. A couple of times an endless loop got wedged in to check register contents in the VICE monitor. Never underestimate pen and paper! In the beginning I tried to get my flatmate's attention. "Look! A moving sprite! Look! Some enemies! Look! A scroller." Well, I had to test it myself at first, but as soon as Pulse got sound I turned up the stereo to its max, and everybody and their mothers squatted the joypad for at least an hour each. I just took care that the collision detection was accurate or at least forgiving. It had to be a fair game. My flatmate scored 4271 points, by the way.

Q. Apart from Commodore Free have you had any other comments about the game?

The folks at the VIC-20 Denial forum blew my hair back with their encouraging comments. Love you, too! They made me continue working on it when it had no score counters or sound. I expected this to be another piece of software of mine that'd get dumped into oblivion. But then I got cartloads of positive comments by very excited people and that was totally unexpected and scary. Pulse got a 100% rating on pouet.net. The Micro Mart magazine, which is printed in Britain, told me that they'll publish a news piece about it this week. I didn't read it yet but they already let me know that they find the game most impressive. This thing went totally out of control and beyond wildest dreams on special medication. This possibly cannot happen again. That depression I mentioned is cured for sure, though.

Q. Imagine you could go back in time and were given the option to change one part of the VIC. What would you change?

Full documentation and an assembler shipped with it instead of just a BASIC handbook. It would still be a great educational toy for kids today like that.

Q. Do you have any question you would have liked to have been asked?

Not really. Thank you very much!

Optimizing cc65 Code: What the Author Didn't Tell You

By Joseph Rose, a.k.a. Harry Potter

[Intro]

Welcome to my cc65 C optimization documentation! Here, you will find some tricks and techniques to produce the best C code under cc65. Most of the techniques here will work for other 6502 targets, other C compilers, and even other compilers and interpreters. I'm not yet an advanced programmer, but I believe these optimizations to be useful. Some of these techniques may be obvious to some, but you may still find something useful. This document is organized into sections; each describes one technique. The individual techniques follow:

[Middleman]

If your code needs to call the same function or group of functions, in order and with some of the same parameters, you can use a "middleman," where the middleman will accept the call, and call the base function or functions for the callee and supply the constant parameters. This will require some parameters to be passed only once in your code, making for smaller code. An example follows:

Instead of:

```
-----------
extern int i[10];
int func2       (int ramcount,
                 enum machine m,
                 char* language)
{
      ...
}
void func (void) {
{
      i[0]=func2 (64, machC64, "c");
      i[1]=func2 (64, machC64, "BASIC");
      i[2]=func2 (64, machC64, "assembler");
      i[3]=func2 (64, machC64, "FORTH");
      i[4]=func2 (64, machC64, "Pascal");
}
------------
```

Try:

```
------------
extern int i[10];
int func2       (int ramcount,
                 enum machine m,
                 char* language)
{
      ...
}
int func2a (char* language)
{ return func2(64, machC64, language);}
void func (void) {
{
      i[0]=func2 ("c");
      i[1]=func2 ("BASIC");
      i[2]=func2 ("assembler");
      i[3]=func2 ("FORTH");
      i[4]=func2 ("Pascal");
}
----------------------
```

Caching Variables]

If you need a particular element in, for example, a multi-subscript array of structs many times in your code, it is a good idea to read it once, store the value in a local variable and access that instead. If you need to access different members of the struct, assign the struct's address to a local pointer and use that to access the struct. This is made even better if you use a zeropage variable.

[Minimize Function Usage]

Don't use functions you don't need. If you don't need the services of mprintf(), don't use it.

Examples follow:

* My CBMSIMPIO library simplifies displaying text and numbers on the screen. If you don't need the services of the standard screen output library and CBMSIMPIO can do the job, use CBMSIMPIO instead. This can save 2-3k in your program.
* If you need to copy memory from one location to another and the two never overlap, don't use memmove(). memmove() requires more overhead, and memcpy() can do the job.

[System-Specific Functions]

If you're writing code for a specific target, use functions made for that target. This requires less overhead for conversion and otherwise makes for better code in general. An example is if you use file access with a CBM model, using the CBM OS functions to access the OS directly.

[CBM Control Codes]

The good thing about CBM screen output is that it can contain control codes to perform functions such as change color or clear screen. If you need to, for example, clear the screen before writing some text, including a clear screen code in the text can save from an explicit clrscr() call and shave 4 bytes from your code.

[Assembler]

Most programs can be created solely in C. However, some programs may require at least some assembler. When deciding to use assembler in your code and where, keep the following in mind:

* C is a medium-level and is good for calculations and program flow.
* Assembler is a low-level language and is good for data-crunching, hardware-manipulation and OS calls.
* If C can do the job immediately, you should use C.
* If C needs to do a work-around to do the job, you may want to use assembler.
* If you want or need full control over hardware or the computer, you should use assembler.

Don't be afraid to use assembler. It can be beneficial if used properly.

[Assuming Parameters]

This is similar to the Middleman optimization. If a function only needs one value for a particular parameter, remove the parameter and replace it with the value. Then, remove the parameter in the declaration, definition, and calls to the function.

[Tokenizing Calculations]

If you need to use the same calculations over and over, store the calculation(s) in one function each and call the function(s) as needed.

[Optimizing Longs]

On an 8-bit computer, longs are very slow and require a lot of code.
Fortunately, using pointers to longs seems to produce tighter code. I think this is because it allows your program to handle words, while the compiler provides the routines to handle the longs referenced. This, however, should slow down your code even more.

[Calculate Once]

If you need the result of a particular calculation several times, perform it once.

[Use Switches]

Switches are good for many possibilities. Switches load the value once and perform several comparisons on it, saving from the extra loads necessary with ifs. The exception to this rule is a true/false case which works better on ifs.

[Incremental Switch Returns]

When you use a switch to return an incremental value where each condition returns one more (or less) than the previous, reorganize the code by putting the highest (or lowest if less than the previous), using an increment/decrement instead and remove the breaks on all except the last if necessary. An example follows:

Instead of:

```
--------------------------------
char c=0, d;
switch (c) {
case 1: d=1; break;
case 3: d=2; break;
case 2: d=3; break;
case 4: d=4; break;
}
--------------------------------
```

Try:

```
--------------------------------
char c=0, d=0;
switch (c) {
case 4: ++d;
case 2: ++d;
case 3: ++d;
```

```
case 1: ++d; break;
}
-------------------------------
```

[Toggling Bools]
If you know that, for example, b is a bool and either 1 or 0, toggling b using b^1 is shorter and probably also faster than !b.

[Ifs Without Elses]
If you have a series of ifs, all of which are mutually exclusive (i.e. only one will work anyway), exclude the elses. In this way, you avoid the extra jump over the next elses.

[Assigning]
Assigning a value when it's first used can save an explicit load. Ex:

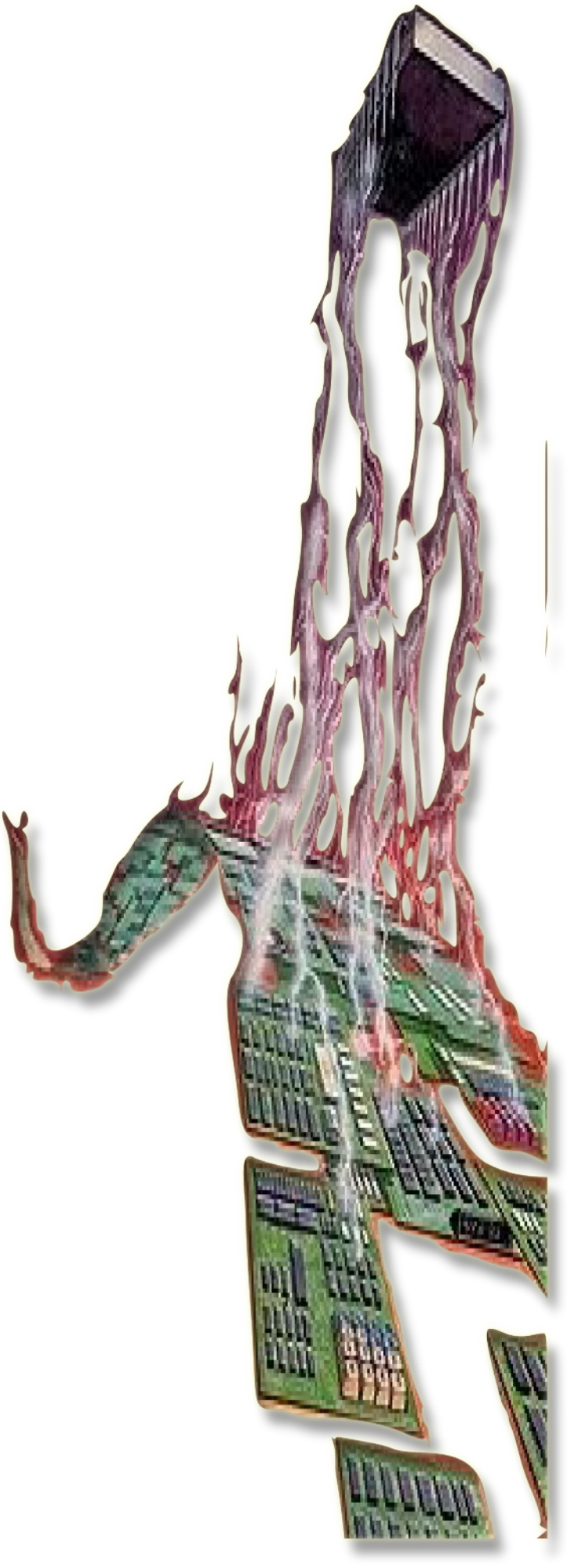
Instead of:
```
------------
c=a*2-1; f00(c);
------------
```
Try:
```
------------
f00 (c=a*2-1);
------------
```

[Ints are shorter and faster than strings]
If you need to compare a string to a list of other strings several times in your code, do the compare once and set an enum to the string's number and substitute the number compare. This also allows you to use switch() to delegate the different tasks instead of a series of !strcmp()'s.

[Avoid Unnecessary Error-Checking]
If a
[Footnote]

I am glad you looked at this document. Please, tell me what you think! If you find this document to be helpful, e-mail me. If you have any suggestions, complaints or comments, e-mail me. If you have any additions, post them on the cc65 contribs site.

My e-mail address is
rose.joseph12@yahoo.com.



INTERVIEW WITH DANE BILLS PANICMAN VIC20 CREATOR

Name: Panicman
Authors: Dane Bills, Jeff Messner
Released: March 2, 2014
Requirements: VIC20 with +3k or +8k, joystick (developed on NTSC)
Description: A maze game clone of a well known 80s game written in assembly

Video of first test:
http://www.youtube.com/watch?v=MKnd8T4pi18

Both versions should load and you can type 'run' from the basic prompt after loading.
e.g.: load "panicman3k.prg",8

There is no difference between the 3k and 8k version other than a splash screen to show the authors on the 8k. I just thought it might be nice to have the 8k executable for someone to run on the real iron if they didn't have a 3k cartridge. The 3K has received the most testing on real hardware.

+3k version:
https://drive.google.com/file/d/0B0VOPYWAvrJHZjJBQW1uVWVYcDg/edit?usp=sharing

+8k version
https://drive.google.com/file/d/0B0VOPYWAvrJHSHhBd0hJbHg5RE0/edit?usp=sharing

discussion thread:
http://sleepingelephant.com/ipw-web/bulletin/bb/viewtopic.php?f=10&t=6870

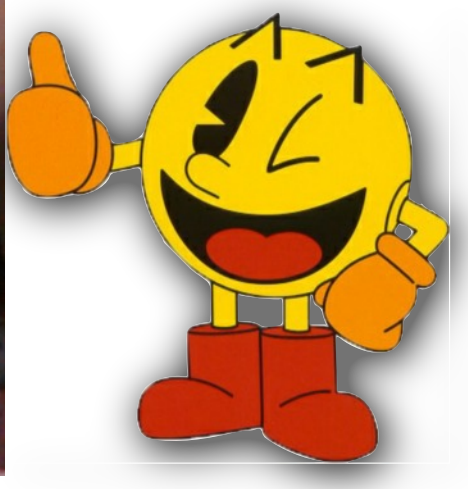
In Commodore Free issue 79 I reviewed a new release for the VIC 20 called *Panicman*, the game received a very high score not just for the game play and sounds but for its accuracy in the conversion to the VIC, I took some time out to chat to the coder, and find out more about the games creation.

Q. Hi. Can you introduce yourself to the Commodore Free readers?

Hello Nigel.

First, let me say I'm honoured to say a few words on Commodore FREE about my tiny contribution to the amazing stuff coming out of the retro-computing community. If it wasn't for all the people creating projects and content this wouldn't be near as much fun as it is.

I'm one of the generations of kids from the 70s and 80s whose lives were really touched by the introduction of the personal computer. A friend from church had shown me an Apple II some time around 1980. I remember looking for a computer of my own. The KIM and AIM65 were the only products even remotely affordable back then. Finally, I saw the "Wonder Computer" at a local Hamfest. It really was a wonder for its time. At under $300, I was able to successfully beg the parents for a VIC that Christmas. I found a picture of my first original VIC. I did a lot of programming in BASIC during those years. I made Gorf and Asteroids, both with custom character sets. They were dreadfully slow of course, being written in BASIC.



Q. Noted in the Credits is Jeff. Are your musical skills lacking? I see Jeff helped with play testing as well as the music. How did you go about testing the game?

I played in elementary school band for what that's worth – not much I reckon. I first started out by dumping the sounds from the arcade into a spectrum analyzer and trying to reverse engineer them. That wasn't a whole lot of fun. Jeff has perfect pitch – he hears notes like I see colour. He took over designing sound effects and music. After his initial distaste at the VIC's out-of-tune scales, he got down to the business of carefully picking notes. He managed to hand compress the Pacman song into an incredible small number of bytes, as I didn't leave him very much room – ever, to put the sound and music in. I was too busy eating up all the memory with beginner 6502 code.

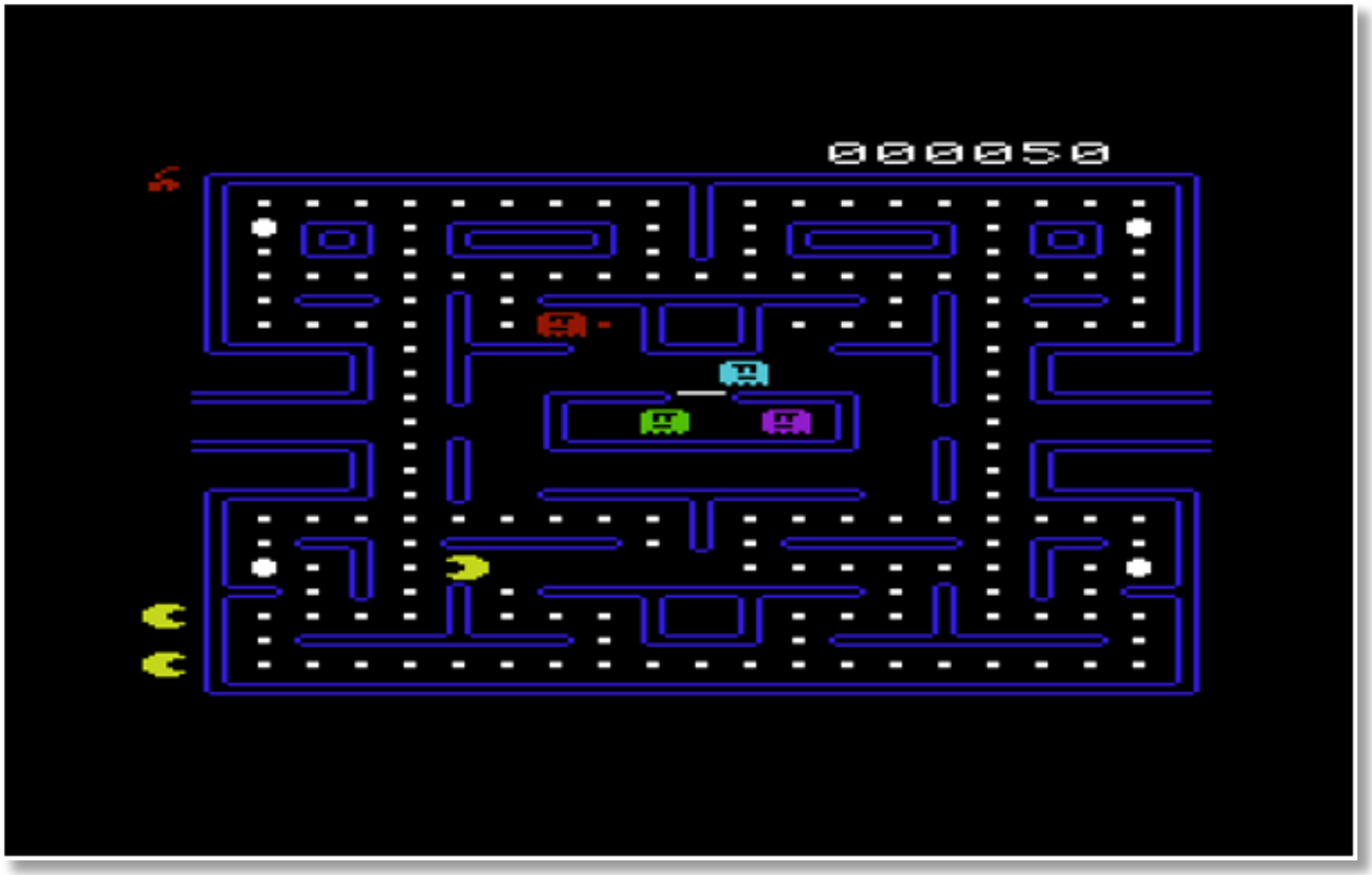
We had some soft "defines" in the code where you could become invincible, or have the level end after X points, and have a second player control one of the ghosts to set up scenarios for the AI. Early on most of the testing was done in WinVICE. Later, with much excitement, I procured some real hardware from eBay to try the game out on. That was its own sort of adventure. We had to find an old Pentium computer to run 64HDD on for file transfer. I had to solder up some transfer cables. That's when I realized this was truly fun, as you could allow the project to take you into any sort of weird area you wanted to wander into. The big test was a self-imposed deadline from Jeff. We had to complete the game in time for a retro gaming party. We loaded the game on a real VIC and let it run for 5 hours alone with strangers and a joystick.

Q. The credits say "developed on NTSC." Now I know its PAL-compatible as I played it on a PAL machine. I presume testing was conducted on PAL machines, and apart from the speed, are there any other differences between playing on a PAL or NTSC machine?

Yes, I guess there shouldn't be much difference, save for the speed. I could tell I was new at this because it never occurred to me to try to run the game in PAL mode on an emulator. I was worried when one of the UK members, Beamrider on Denial, played the game and it behaved strangely. I had not bothered to lock the frame rate down. With the extra time per frame that PAL offers, occasionally the game loop would finish in one frame instead of two. This made it jerky. I made a patch to lock to 1/2 frame rate so the speed would be constant on PAL.

Q. I see in the forums you were trying to make a tape version of the game. Do you intend to distribute this (obviously there will be some copyright issues)?

I never did manage to get a tape made. Jeff thought the retro game party might be amazed to see a game load from cassette tape. Many of them had perhaps only been familiar with cartridge based systems such as the 2600. That was everyone except for his uncle, who I understand had given Jeff some computer tapes for his C64 when he was a kid. The tapes were so tiny in size, Jeff thought his uncle had been unduly cheap with him. He later realized they were data cassettes and much more expensive than regular audio cassettes. I can't say I ever had any data cassettes back in the day; it was always a re-purposed audio cassette. The Commodore tape drive never failed me though, unlike the Atari.



Q. Why didn't you just release the 8k version of the game and maybe spice it up with a splash screen?

One of the nostalgia requirements for the project to me was that it had run on the same hardware I had in 1982. I was enamored with the PLOT and DRAW statements of the Apple II back then, so I had scraped enough pennies together to buy a "Super Expander 3K" memory expansion cartridge which had the additional basic commands built-in. This was the target; it had to run within a 3K VIC.

Q. I did comment that the ghosts were a bit dim in my review. You mentioned you tried to copy the AI as close as possible. Was this from the original arcade game?

Yes. I was going to try to verify the correctness of the ghost AI by demonstrating that the same patterns that would work on the arcade machine would work on this version, but that did get out of scope quickly. I would like to point your readers to the brilliant Pacman dossier by Jamey Pittman:

https://home.comcast.net/~jpittman2/pacman/pacmandossier.html .

It contains detailed descriptions of the arcade ghost AI I tried to copy.

When Jeff and I were getting ready to release the game we had a bit of a heated exchange about the game difficulty. It seemed a bit too hard at first and he thought we should turn it down a bit for the party so that people would enjoy it more. After we tweaked the difficulty I always thought it was too easy. Oddly, when I play the original Pacman on MAME I get about the same score across both versions. One of the things that we were doing to make the game harder was decreasing the chase/scatter timer, such that there was a larger period of time when the ghosts were chasing you vs. heading back to their "home" tiles.

I know a few areas where the ghost AI is mismatched to the arcade. I really want to dig back into the game and make sure it's correct. I believe it's possible for the vector math on Inky to overflow and have it mess up his targeting tile. In "frightened mode" the random direction selection isn't working properly. Also, the order in which ghosts evaluate their moves when two moves are tied for the shortest Euclidean distance to their target tile is different than the arcade. I think this is causing there to be more than one hiding spot in the maze. The arcade has one legitimate hiding spot.

One thing that would help make the game harder is the faster version I am working on. When you have less time to plan your moves things get a lot more hectic. I have it running at the full 50 or 60 frames now, but I need to rework the timing constants for the relative speeds of Pacman and ghosts on different skill levels before I can release it.

Q. How accurate do you think the game is to this version?

Hmm... well, it's got to be at least 50% correct. Clyde is supposed to return to his home tile if he gets too close to Pacman. I made him return to the opposite Cartesian coordinate area of the screen. I thought that might make him more interesting. Pinky has a bug in the arcade that I did not reproduce but rather coded up his intended logic. Inky needs some more testing to feel 100%. The game does include the two "speed-up" modes where Blinky will increase his speed during a level. This is accompanied by an increase in the siren pitch.

Q. How limited were you with the VIC's hardware. For example, trying to get the maze to fit the screen and still be faithful to the original. Was this a challenge?

You'll notice the maze has a weird spot in the middle. To match the proportions of the arcade on the VIC the whole maze would need to be narrower. I didn't like the narrower maze as it didn't leave enough room in the ghost box for all ghosts to display. The wider maze made some corridors longer than they should have been. They would not allow you to reach an escape passage quickly enough. This is why I made the middle pylons a little larger – to reduce the longest run that Pacman would have before he could turn.

When playing the game on the emulator the non-square pixels of the VIC really seem to make the apparent speed of Pacman change drastically when changing directions from vertical to horizontal. For some reason, on the real hardware on a CRT it doesn't seem as bad. I was pleased about that the first time I played on a CRT. At one point I wondered if it would be worth altering the horizontal vs. vertical speed to compensate for the VIC's non-square pixels.

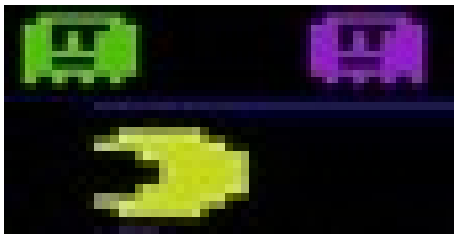
Memory is a brutal taskmaster on the VIC – there is rarely enough, although, as you pointed out in Issue 79 with the Pulse game, miracles are possible, even in the unexpanded 3.5K.

Q. The whole game feels very polished. Was this your first real programming attempt?

The VIC did lead into a programming career. I guess the inventors of those first machines should feel proud of that; they inspired a lot of future programmers. I was doing some C++ for a telecommunication company by 1991. This was the first 6502 assembly project I had ever attempted. I must say – oh my goodness – what a learning experience! I have so much appreciation for how tedious it must have been back in the day. Those guys worked so hard. I think they were some really unsung pioneers of their time.

Q. What do you have planned next? Will it be another port, or do you have an original game idea you plan to unleash?

Everyone loves a cover band – and everyone also loves remakes of classic 80s arcade games. It's a shared cultural experience we all have. I would like to pick one of the "Games We'd Like to See" from the list on the Denial community and take a shot at it. Maybe *Wizard of Wor* or *Elevator Action* would be fun.



Q. You mentioned the Denial community. How important was the VIC community to the project? Would you have just given up without proper support from the community?

When I started the project I tried a promise to not use *any* resources from the internet because I wanted it to be like 1982, where all I had was a paper-printed book to thumb through. I was about 80% successful staying away from the Internet – until I needed some help. What I love about the community is knowing there is a gathering of people who still use the system. That makes all the difference. The VIC20 Denial website was a big inspiration and Robert Hurst's amazing collection of VIC20 games inspired me to try.

Q. So the community is an ideal forum for tweaking and helping to spot bugs. Do you plan any other enhancements to the game?

I'd like to release a second version of *Panicman.* I've been working on one which runs at full speed, and as you mentioned, uses a full 8K to have the arcade title screen and the full intermission and fruit complement. The ghosts should be much closer to arcade AI in it, too.

Q. If you could go back in time to the point where you started coding the game, would you have done anything differently, or dare to say, would you have thought "Nah, I won't bother?"

Oh, dear! Yes, there is so much I'd like to do differently. I always thought that when doing a hobby project you could have everything the way you want it – unlike our day jobs. Well, it's not completely true. Once someone makes a "deadline" for you as I did with the self-enforced date of release, things start to have compromises just like the day job. I would have preferred using an assembler with a linker because that would have made compiling for different VIC memory footprints easier. I wish I could have had time to fix the EMACS mode for DASM so that it would indent properly. It really got frustrating. I would have liked to try some interlacing techniques to do something about the colour clash.

Q. Many readers will ask about the tools, software, etc., you used to create the game. What tools did you use?

One of my early goals was I was not using the built-in debugger in the WinVICE emulator. I made it about 75% of the way through the project before I cried "Uncle!" on that. The tool set was EMACS Editor, DASM Assembler, SVN Source Code Control. I used some Perl scripts to generate the compressed data for the maze and perform the bit rotations for the different frames of Pacman. Jeff would prototype his music directly in Commodore BASIC, sometimes with the assistance of an Excel spreadsheet. I alternated between using Windows and Linux, depending on whether I was at home or ... Ahem, at my "day job."

Q. With so many tools available and cross-assembly being used, do you think it's easier to program the VIC now, or do these tools just make things faster?

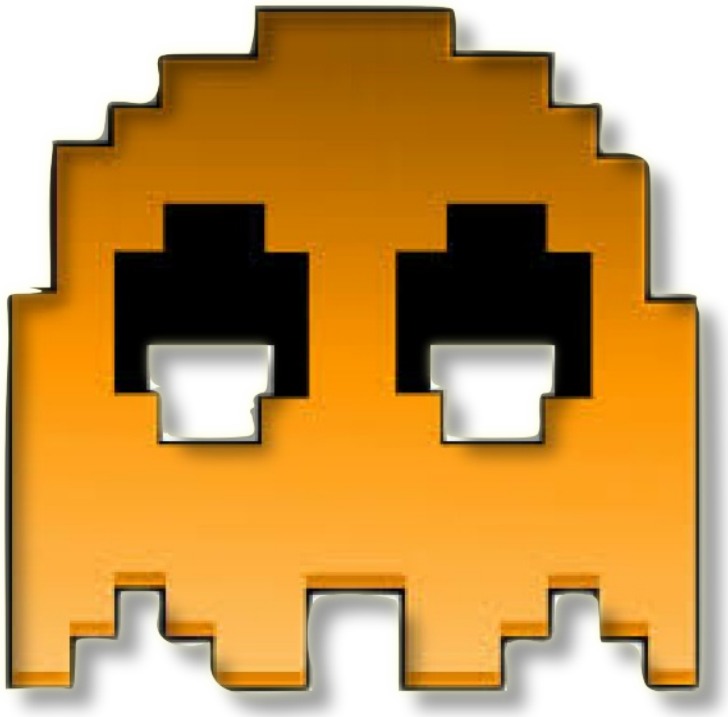
That is a good question. I think it makes things faster mostly. I jokingly refer to the assembler as the "Quantum Assembler." With modern PCs it's done assembling before you even hit the Enter key. Certainly some things I would have hated to do by hand with graph paper – like rotating all the Pacman images. On the other hand, debugging can be a big pain, unless there is some source level debugger that single-steps through your original source code that I'm unaware of. I'm not sure that we really have it that much easier now than then in that respect.

Q. If you could code the VIC in the 80s, what would you have done (maybe a career change)?

Yes, that was the dream – to make a career of programming 8-bits. Honestly though, I wonder, "If I had been the age I am now when the VIC20 first came out, would I have paid any attention to it?" I say that because I pay very little attention to the "popular" technology that is out there right now, like the iPhone, for example. I may have written the VIC20 off as some fad of "the young" while I was busy with my IBM and COBOL programming.

Q. If you could go back in time and change one thing about the VIC, what would you change and why?

Wow, another killer question! Hmm... I remember the day I unpacked my VIC, lamenting how low the screen resolution was compared to the Apple II and Atari. However, there was no way I could have afforded those. That is the obvious complaint of a 12-year-old in 1982, but looking with the eyes of an adult in 1982, what would I have changed? Hmm... I don't think I would have changed a *single* thing. Anything added would have increased the cost – and that was the whole point back then. Maybe they should have included the *Programmers Reference Manual* as part of the base documentation. Commodore had great documentation.



COMMODORE FREE INTERVIEW WITH THE CREATOR OF THE C64p

The Future was 8bit is a website that has grown from a fanatical Commodore owner's own need for toys into a one-stop shop for SD2IEC, C64p, and other peripherals. "My own long-held desire for a C64p is the only reason my line of SD2IECs exist. Something like three or four years ago I built an SD2IEC to try with a C64DTV. The left-over SD2IECs ended up on eBay... and the rest is history."

First, what is the C64p? Well, it is a C64DTV-based laptop measuring 21×14 cm. which Nic has customized. The laptop starts as a new product; then all the insides are removed – even the the TFT screen is replaced. The only remaining parts are the battery and chassis, which are then populated with an SD2IEC, keyboard interface, three custom PCBs, 7″ TFT, 1530 joystick-mode mouse, and an ASUS unit which replaces the charger/PSU. Add a dash of custom firmware flashed onto the DTV chip (which lets you choose kernels). JiffyDOS anyone?

http://www.sd2iec.co.uk/index.html

Specifications of the retail C64p

- C64DTV PAL
- 7" TFT (480×234 Pixel) 4:3/16:9 + infra-red remote
- Mouse pad emulates a 1350 joystick-mode mouse in port 2
- Joystick port x2
- IEC disk/printer port (rear)
- SD2IEC (left)
- Disk swap button (left)
- SD2IEC button root/reset (left)
- Speakers x2
- Volume (rear)
- Power/Charge (rear)
- C64DTV Reset (bottom)
- C64DTV firmware upgradable via Joy2
- LED – Green – Power
- LED – Orange – Charge – bright full charge rate – dim trickle charge
- LED – Blue – SD2IEC
- 1800 mAh Battery (run-time approx. 3.5 hrs depending on system load)
- Audio/Video out (the output isn't switched so the display is dimmed)
- 100-250 VAC Asus Charger
- Custom firmware which supports a number of built-in DTV games, plus JiffyDOS, BASIC, and file browser
- Colour fix has been applied
- Keyboard Twister NG is also fitted

Notes about the Keyboard NG

This means a lot to people who know about the DTV. If you connect a DTV to a real keyboard the biggest issue is the lack of F7, among other oddities (not good if you want to play KikStart!), but the keyboard twister fixes this. NG – Next Generation gives extra functions. There is a hack on the NG ROM (the standard ROM is for German keyboards) so it runs in US mode. The NG ROM, with a simple add-on diode, gives a user the ability to "CTRL-ALT-DEL", which resets the DTV. No need for power-off/power-on.

Q. Will you please introduce yourself to our Commodore Free readers?

Hi Commodore Free readers! I'm Nic from Dorset in the UK. I've had Commodore computers for (I hate to say) 31 years, and still use one every day!

Q. How did you become involved with Commodore and computing in general?

Christmas '82 or '83... I found a VIC20 under my tree. A few years later I managed to get my Dad to buy me a C128… a few years later I bought myself an Amiga 4000. These days, of course, I work in IT – Yay me!

Q. The C64p, although expensive, is quite an elegant-looking device. How long does it take to make the unit?

Well, first – it's expensive because it costs an eye-watering amount to build. I never really thought I'd sell any. After all, I really only built it because I wanted one, but the word got out, so I made a few extras. It's difficult to guess how long one unit takes to build. It took me about 3 years to get them to where they are now, but if I had to guess, I'd say 20-30 hours each.

Q. Are these items made to order – or are they in-stock?

All of the units sold so far were sold from stock. All of the components are in-stock, so I don't have to go fishing for DTVs if I want to make more.

Q. I see you have JiffyDOS. Is this licensed? Some forums have suggested this is not the case. Do you think JiffyDOS was an important addition to the device, and does it actually speed up loading from the SD card? Or is it more for the convenience of extra commands and features?

I don't really follow too many forums, and I really don't get the chance to sit in front of my PC with nothing to do! Anyhow, yes – technically it's not licensed, but it's not the same JiffyDOS you'd find inside a real C64. DTV-JiffyDOS is a hybrid that is easily available for download from the internet. JiffyDOS, just like on a real C64, transforms loading times.

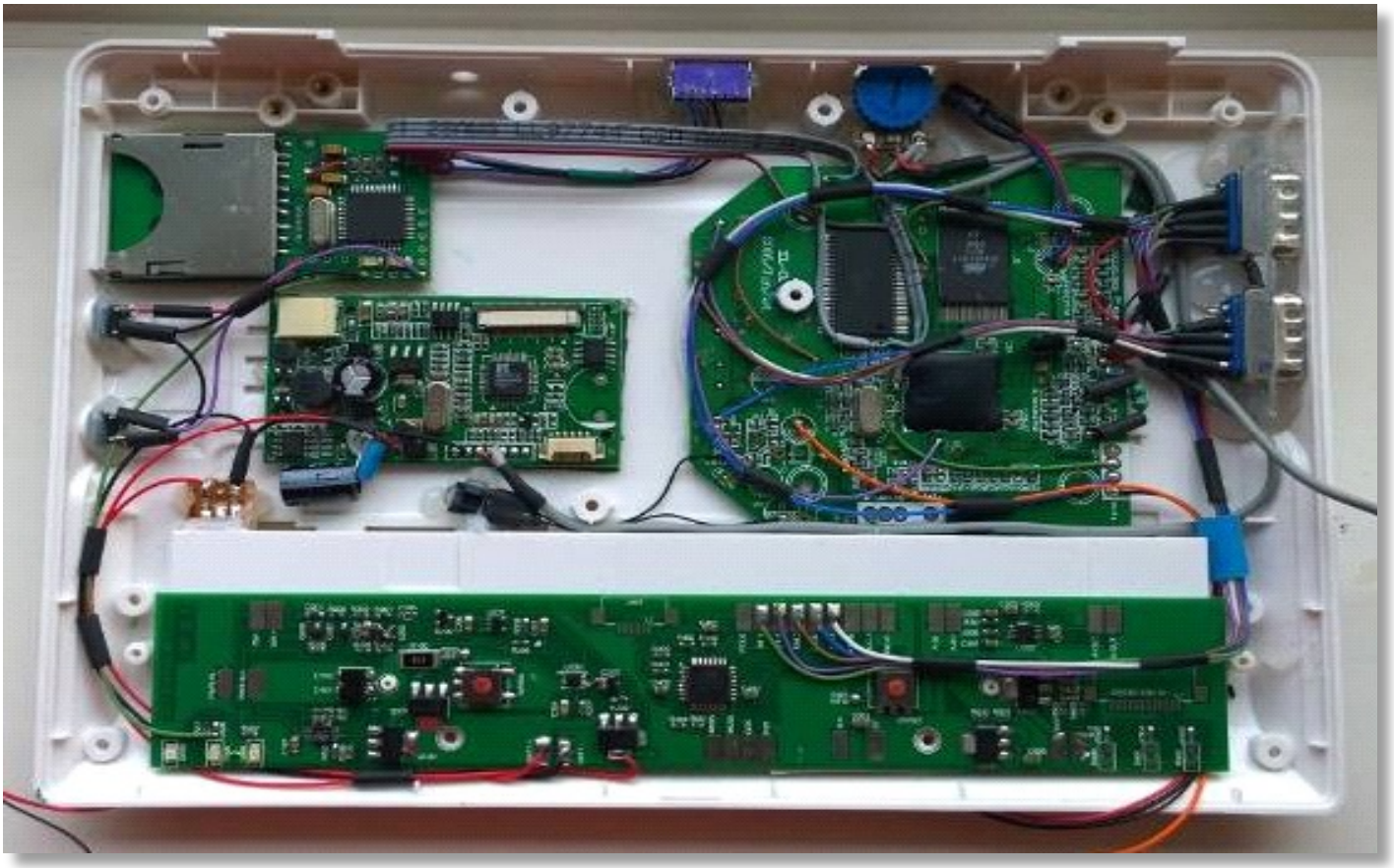
Q. Talking about forums – have you had any negative comments about the unit, and do you have any comments you would like to share back to readers? (no swearing, please)

Urm, I have read interesting comments like, "I'd pay no more than £80 for one" or, "It's a DTV, so it's not 100% compatible." Makes me laugh, as you've only got to use one for 10 minutes to know it's something special. I've loaned a couple of demo units out and the feedback was amazing. It seems most people`s idea of the DTV doesn't do it justice. Yes, some functions are slightly non-standard, but overall, once you turn one on, you don't want to turn it off. The people at CiA

http://awesome.commodore.me/articles/kitty/commodore-c64p-review/

gave me some wonderful feedback. Originally (before they used one) they thought something like this could retail for £100-150, but after they touched one they realised how cool these things are and how much has gone into making them. They agreed that you'd never get anything like this for 100quid.

They really do look "shop bought" and not knocked together. Everything inside the C64p has been built with



longevity in mind. Calculators were used in the making of these units! To give you an idea – the original donor laptop's charging circuit could only be described as dangerous; I tested seven original chargers – and I kid you not – only two worked after 24 hours, and four of them went "pop" with sparks jumping out of the nasty plastic so-called CE-marked PSUs. My charge circuit was designed by a friend that has designed charge controllers for Nokia/Motorola and Panasonic.

Q. I find it interesting that people spend so much work developing units like this, only to be thrown abuse from the community. Yes, its expensive, but we have already spoken about the amount of work involved. Also, I suspect that in reality there is very little profit to be made from such a unit. Would you like to comment?

That's why I don't bother talking to people in the online communities that do not give anything positive back into them. Some people have a real passion for all things retro; others can only say they could do it better, but of course they don't. The Internet is full of freetards like this; hey – they're probably still living with Mother! If I was trying to make a living wage making these, well, frankly – I couldn't. It is important to make a profit; as with everything I sell, it has a warranty. So you have to cost things with this in mind. If one breaks I have to replace it.

Q. What was the motivation? Was it just a challenge – or did you see a real need for this?

I wanted a C64 I could use anywhere, so I got off my backside and did it!

Q. Of course we need to talk about compatibility and the DTV was never 100% Commodore-compatible. In your experience, how compatible is the device (you mention it runs GEOS)?

No, it's not 100% compatible, but it surprised me on how much stuff does work. From what I'd read about the DTV I wasn't expecting much, but just about everything I've personally used seems to work peachy! Yes, I did have a working GEOS – it was amazing! But some idiot (me) formatted the SD card. It's on my to-do list to recreate the DTV-GEOS disk images.

Q. In GEOS, can some of the DTV's memory be used as a ramdisk?

Yes, this can be done, but the version I ran didn't use this. It's something that I'd love to spend some time on, but I'm kinda hoping that one of the C64p owners might do the leg work for me! I've got other portables in the works, you know, and very little time to play with my own toys.

Q. Maybe you would like to tell our readers about some of the other services you can provide.

I mostly get questions regarding C64 and its repair – that's fine by me. But I can answer most questions on just about any 8/16/32 bit machines, and I'm always happy to help (no Apple please, I'm British).

Q. Why did you select the model of machine you did to start the customisation?

The donor laptop is one of those OEMs you see with different brands printed on them. In fact, I bought so many of these things, I even found OEM clones!

Q. One thing you have created is a very professional looking piece of hardware. It doesn't look to quote yourself like it's a "butchered piece of hardware." Do you have plans for any other hardware device?

Yep, they do look good, even if I do say so myself. Like I've already said, the feedback on the C64p has been amazing. I always worry when you sell things like this, and yes, I do have plans for more hardware. Two more laptops (original hardware) are in the works with a possible third – probably eye-wateringly pricey. They will need custom plastics.

Q. Have you been contacted by Jeri Ellsworth, the creator of the DTV? I wonder what she makes of your customisations. You could say the DTV has been customised to the max with the creation of the C64p.

I did send her a tweet. She probably thought, "Yeah, whatever." I didn't hear anything back.

Q. Of course I have to ask – why call it the C64p?

It was gonna be called the if64 (rude)... "P" for portable (little p, as it's small), and 64p is slightly humorous because I'm British and it doesn't cost 64 pence (we say pee, BTW).

Q. Finally, do you have any comments you would like to make?

Trust me, some of you might do a little wee when you see what I've got in the works

www.thefuturewas8bit.co.uk

Spaghetti Code.
By John Fielden

'GOTO' BASIC (aka 8-bit or "procedural") has often been criticised for causing spaghetti code.

Sorry... No! Programmers (I use the term quite loosely!) are responsible for getting their work in a mess! Look at it this way: When you have to clean your room, you don't say, "*Oh, it's that new wardrobe from MFI. We should have gone to Argos!*" You just don't, do you? You get the missus to do it!

Joking aside, "Good programmers have a pen & paper handy," as the saying goes. However, in my case I seem to use the technique backwards. When something I've started becomes too big to check through on-screen, or there's something I can't figure what I've done wrong simply by looking, that's usually when the pen & paper come out of the drawer. It's a great way to tidy up the mess.

Believing the common trend that blames procedural rather than a person's logic, planning capabilities, etc., the VB.Net company have pretty much destroyed that which was the beauty of BASIC and basic programming. Now, as I look through the walk-through guide of the Visual Studio 2003 Edition, there's very little difference between VB, VC, VC++, VC#, and so on. What is the point when the only real difference is in the syntax (grammar, loosely speaking)? Sometimes it's merely a case of throwing extra symbols in!

I mean, what is the point?

Anyone interested may as well invest in learning its common (intermediary) language. The rest. On the face of it, at least. Seems pretty much surplus to requirements! A waste of the company's time and money investing in the other languages (which more and more seem a mere split off from BASIC anyway - not withstanding OOPs).

OOP, on the other hand, is much more likely to cause SPAGHETTI CODE than any amount of GOTO statements. First, you've got to decide how to band together the groups of events. For instance, do you put the *Click* events together, the items being *clicked* together, or instructions to let this monster which is actually being *clicked*? (ie. left or right mouse button, or whether it is hovering, moving, etc.) When you've figured that little lot out, we're back to the problem of variables and how they just get discarded when you clear the page for a new one (effectively starting a new form). Note: a new form is not the same as a new project (called a Solution in VS.). Glue is a Solution! At least with glue people stick to it!

VS (even BASIC) have become so convoluted, and people aren't bothering to keep up with it. In spite of the claimed statistics, which are actually well-worked propaganda, BASIC is claimed to be the most commonly used language. Maybe so generally, but the DotNet books claim this of VB.Net.

Now, I have issues with this (having looked beyond the wording) and have seen what it is trying to lead us to believe!

1. Is VB.net the most commonly used .Net language?

2. Is it most commonly purchased in recent times, bearing in mind people will have long ago bought other versions, languages, etc.? Most will still have them, and if not, these are usually available free, though being for the purist, are rarely updated to become usable mainstream. Ironically, the only one I know of that *did* try ended up as an OOP version that this writer is complaining as to the complexity of!

3. There are probably several versions vying for attention, and they may concentrate only on specific things. Does adding these (as one) change the stats?

4. How many start with it – only to give up on it? I am quite near doing this, having gotten nowhere beyond pretty front ends in my ten long years – with this and dialysis!

There are probably more reasons in unravelling the propaganda when you look at it. The marketing and advertising capabilities (and prowess) of the richest company the world has ever known – and perhaps its owner, the richest man since Solomon (from the Old Testament, some 4000 years B.C., who graced the people of that ancient time with wisdom – at least he brought that for all his riches!).

I was shocked when I heard Roy, the DER technician who helped me with random numbers. I wrote about it in a prior Nostalgia issue. I learned he had ditched it for DarkBASIC. I'm into apps rather than games, and it sounds like modern-day consumers won't take such things seriously.

So, the last nail is in the coffin; the last straw is drawn. None of the fifty or so books, nor any of the Googled items, tells me in simple, non drawn-out terms how to keep variables in memory across multiple forms (though you shouldn't have to do anything until you wish to free memory by clearing the variables).

The latest is keeping a Boolean, caused via a button click, on a form. I have tried to follow everything I've read though, but can't find anything specific to this. I have even tried creatively with variations and my own ideas – so much so, that the only thing left to do is to wave two fingers in the air – and walk away.

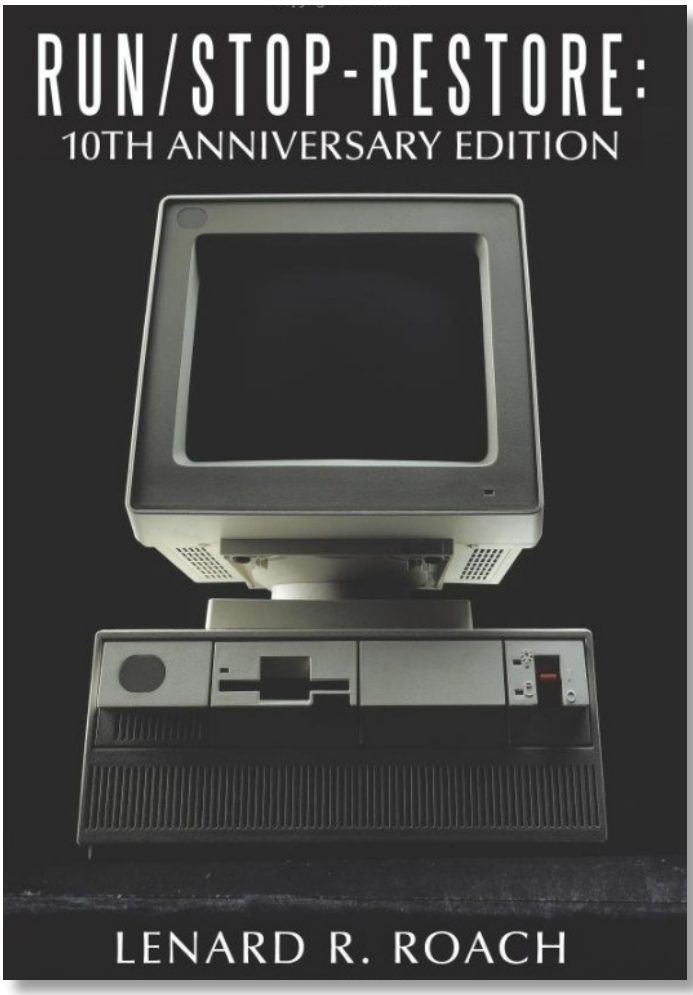
Here's to unhappy prog'ing...
May you all have better luck in your endeavours.

Never On A Commodore
by Lenard R. Roach

I used to do everything from surfing the Net (not the Internet) to budgeting to writing – and Heaven knows what else – on a Commodore. I used to spend several hours a day on the Commodore just doing whatever I wanted. My biggest fun was creating programs in 64 mode on my 128. I enjoyed the fact that, with each subroutine written in BASIC, I added to a larger compilation of separate subroutines that would eventually conglomerate together to become a functioning program by tying each subroutine with either a GOTO or GOSUB command. But, even in programming it was frustrating to look for those little gremlins called *bugs*, which were little BASIC commands worded incorrectly or placed in the wrong subroutine.

I remember working on the program *Check It Out* recently, trying to make the program more compatible with its sister program, *Check Mate*. For those who are not aware, *Check It Out* made it into the very last issue of RUN magazine as one of the featured works of the month. RUN magazine gave me $150 for the work, and like a fool, upon signing the contract, I lost all the rights to make improvements to the program unless I received permission from RUN. Now, here in the 21st century, the programs once owned by RUN have passed from company to company and from hand to hand so that only God knows exactly who owns them now.

If any of you have read my book *Run/Stop-Restore: 10th Anniversary Edition*, I talk about dealing with one of the companies that I found back in the early 2000s that had my program. At first the company wanted me to publish any upgrades in their own magazine which they started to replace RUN, but I couldn't see myself once again signing a work-for-hire contract and losing a second work to the legal nonsense created by my failure to use foresight. Thus, the program upgrades never saw the light of publication.



What did I do with the upgrades? Right now they just sit in my collection of Commodore programs waiting for the day of rebirth. Occasionally, I break these programs out and run them through their paces to see if there are any more changes I can make to them, but so far they seem to fit the bill as they were designed, and the Spirit of Creation hasn't lighted upon me to make any more changes. It is getting time to pull them down again, clear off the dust, and check, just one more time, to see if there is anything I can do with them. Maybe, just maybe, the light will come on and I will begin to work on them again.

Well, like an old man, I get into a cul-de-sac and forget what I was writing about. This article was suppose to be about the wrestling match in getting *Check It Out* and *Check Mate* to talk to each other. Maybe a run-down of what each program is supposed to do may bring some light to what I'm saying.

Check It Out was designed to allow a user to type in all the information needed on a wallet size check. Then the program would print off that inputted information onto a check inserted into a standard Commodore 9-pin printer. In the age where the Internet and hardware has made it easy to pay bills instantly and on-line, this program seems a little moot, but if there are any users out there like me who still like to make out checks, and create for themselves a paper trail to look back upon, then this program is for them. At first, I wrote this program because my hands have developed carpal tunnel and writing checks week after week had become a painful experience. *Check It Out* alleviated some of that pain and made the check writing experience a little less awkward. When I started making out checks ahead of time for gas from my local convenience store, the clerks were impressed by how it was done – and how neat it was to read a check that didn't look like it was written by a three-year-old, since most people in my neighborhood apparently had bad hand writing.

This is when I got the idea to try and go public with the program. First I contacted RUN magazine by mail, thinking they had hundreds of programs to go through, and something as simple as a check-writing program would never fare against some of the works they have published in the past. RUN responded to the positive and sent me a copy of *RUN Script* (their response to COMPUTE'S *Speedscript*) and told me to send the program in with a

corresponding article written in *RUN Script* format. It took me several months to learn how to manipulate *Speedscript* and I wasn't about to take several more months to learn another word processing program just to write one article, so I cheated. I took the article that was written on how to work *RUN Script*, erased the text, and wrote my article in its place, thus having all the margins and spacing's pre-formatted for me – and it worked. Upon having program and article done, I mailed the two disks in, and waited.

It didn't take RUN very long to get back to me with a letter stating they were very interested in publishing my program code and to start talking contract. I called the number they listed in the letter and talked directly with their front desk, who also happened to be working with *Check It Out* on her computer when I called, and she had a few issues with the program. I thought I alpha-tested all the bugs out of the program before I mailed it, but I went ahead and listened to her complaints. It seemed at the time when she went to hit RETURN to print the check, the program would print two question marks at the top of the check before making the check out. At first I thought there was something I was missing in one of the subroutines that I wrote, so while on the phone, I booted my master copy of *Check It Out* (the one I made the copy from that I sent to RUN magazine), and went with her, step by step, through the check set-up process. My version ran just fine with no question marks popping up; when she ran hers the question marks again appeared.

A mystery.

Well, with any mystery, as Sherlock Holmes says, you eliminate the obvious, and whatever remains, no matter how improbable, must be the truth. Software was eliminated, so now it was down to hardware. I asked the front desk person what hardware she was using. She told me she was on a Commodore 128D with special hookups to talk to the remaining computers in the building. Problem? As Charlie Chan would say to Number One Son, "Is possible," but such a hookup shouldn't be interfering with the print status of the program, but I would ask to be sure. She stated that each computer had its own free-standing printer. With that eliminated, there was only one item left, and that was the printer itself. I wrote *Check It Out* to work with the Commodore MPS 802 and MPS 803 printers, with modifications to each subroutine to work with each prescribed printer. I figured this program should work with all Commodore related printers, provided the user knew which line number it was where modifications needed to be made. I asked her which Commodore printer she was using, and she responded that she was not using a Commodore printer, but a specialized electric typewriter that was modified to take Commodore commands.

Aha!

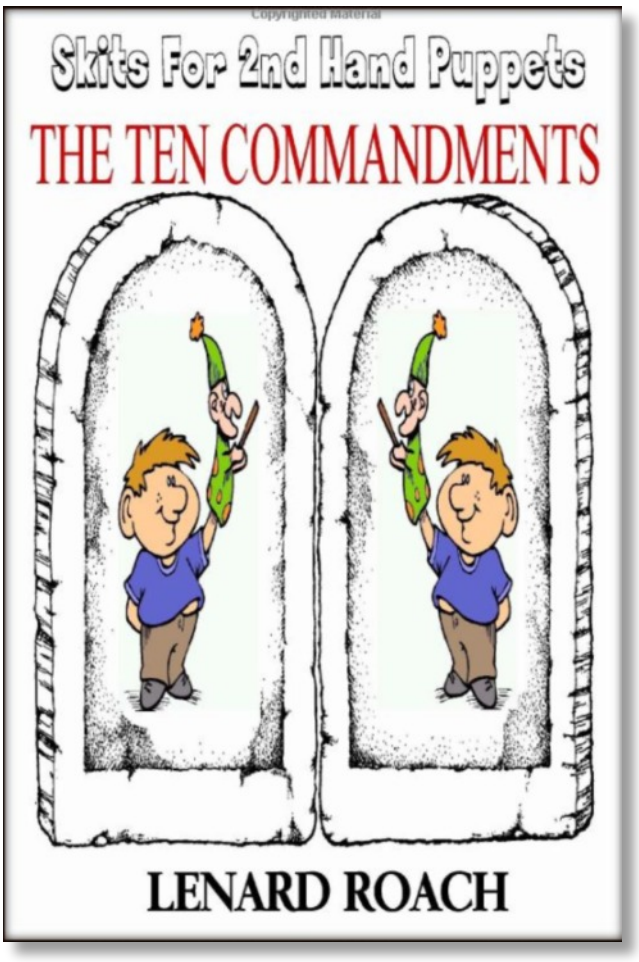
Without knowing exactly what those command channels were, my program would not be able to function properly under those conditions. In order for the program to print on the proper lines of the inserted check, the Commodore had to execute an OPEN 10,4,10 command, which basically told the printer to re-boot and start over from where it left off. Without this command, the printer would print all the information below the line on the check – not a pretty sight. I quickly related this information to the front desk person, who decided to change my article which read, "works with all Commodore and related printers" to "will work with virtually all Commodore and related printers." A smart move on the part of RUN and they helped me save face at the same time.

What the ...?

How did I get here in this cul-de-sac? Apparently, I don't want to write about making *Check It Out* and *Check Mate* work with each other's data. I'm so busy writing nostalgia that I forget what it was that this old man is supposed to be doing. Now this article is almost written and there's very little room left for me to get into the details of all the trials and hassles I had to go through to even get each program to work. Needless to say, when I finally did get each program's data to cooperate with each other, problems remained. Also, let me mention to those who may still be using *Check It Out* in its original published form, they will not be able to use the data formed by *Check Mate* unless serious modifications are made to *Check It Out*. Hence the reason I was looking for the owner of the program in the first place – so I can get permission to release the updated version of *Check It Out* to the public without causing a copyright infringement upset with anyone involved.

Perhaps, with the editor's indulgence, I can compose a second article pertaining to the combat and hardships related to making two seemingly unrelated and uncooperative programs begin to get along. It sounds like a Mideast peace talk conference, and believe me, for the most part, it was. In between articles, I will boot up both *Check It Out* and *Check Mate* and see if they are still getting along, or if more negotiations will be necessary.

In the meantime, the "manhunt" for the new owners of *Check It Out* will continue...



The Assembly Line

"The Place Where Art Meets Science"

$03: The Stack-Part One

By Bert Novilla (satpro)

Hello again! Last time out we refreshed ourselves with a treatment of binary numbers. Today we will discuss a very important component in assembly language programming – the Stack. Just from talking to other programmers through the years (even guys who know what they are doing!) I have come to the opinion that the Stack may be the single most confusing topic in all of assembly language programming. It is without doubt the barrier to entry for many people who explore assembly language. Why is this? Well, I have some ideas, so today we address the facts (and myths) concerning this small 256-byte piece of memory located in your Commodore computer just after Zero Page. Today we will look at the 6502's implementation of the Stack, and next time (in Part Two) we will expand on today *and* tackle how the 65816 implements the Stack. I will show you how to use the Stack effectively (regardless of CPU) in the programs you write. Plus, we're going to get funky and explain some advanced Stack manipulation techniques. If the Stack is confusing or you want to know more about how it works, then please, read on.

What is a Stack?

Many times you need quick, temporary storage for data and the A, X, and Y registers are all busy doing something. Or perhaps you want to pass several parameters to a function somewhere. For times like these there is a mechanism called the Stack. So what is the Stack, where is it – and what does it do?

Well, the Stack is located in an area of RAM immediately following Zero Page in all 6502-based computers. Zero Page, of course, is a 256-byte range of memory, the very first 256 bytes of your computer's address space, or $0000-$00FF. For the 6502 this range of memory holds a special distinction. Many instructions include a special Zero Page addressing mode which executes faster and produces less code. Is the memory itself faster? No. The increased efficiency is due to all addresses in this region of memory having an implied high byte of $00, which means the computer can assume the $00 high byte and do things here using one less byte and one less cycle, and at roughly one million CPU cycles per second, the possibility exists for much more efficient execution because saved (or wasted) cycles can really add up. The Zero Page addressing modes assume a high byte of $00; these addresses are viewed by the 6502 as residing in the range $00-$FF. It is a very heavily used section of memory, and all Commodore operating systems make extensive use of Zero Page memory.

Immediately following Zero Page in memory is the Stack at $0100-$01FF. The Stack is somewhat similar to Zero Page in that the 6502 assumes an implied high byte (of $01), so memory within this 256-byte range can be accessed rather quickly and efficiently, but in a different way – as an offset from $0100. Before we go further, it should be stressed that both Zero Page and the Stack can be utilized using standard 6502 instructions in the same way as any other part of memory, but we don't generally try to program for less efficiency, do we? We usually want one of the special instructions designed specifically for the Stack.

Stack Layout and the Stack Pointer

Stack memory is just like any other memory. Each byte is made up of eight bits like any other byte, but we can use special instructions to read from or write to the Stack. The Stack is also the place the CPU places your return address when you jump to a

subroutine with the **JSR** instruction. The CPU-addressable position within the Stack is automatically maintained by a special register called the **Stack Pointer**. We call the position in memory that the Stack Pointer refers to as the **top** of the Stack, but it's not exactly the top you might envision. It's actually the bottom address-wise, and the position is equal to the value of the Stack Pointer as an offset from address $0100. To put it simply, the Stack Pointer, or S Register, often times abbreviated SP, is an 8-bit register whose sole job is to keep track of the next position within the Stack where data will be written with a special type of instruction known as a **push**. The Stack Pointer works in a way that might seem backwards at first, and that's only because it does work backwards! Stated more correctly, the Stack grows downward in memory. You may remember how the great Commodore pioneer Jim Butterfield described the Stack as a stack of plates in the cafeteria (if you have ever had the opportunity to read any of his excellent books or countless magazine articles). Butterfield taught us that when we placed a plate on the stack it would also be the first plate we pulled from the stack, a system often referred to as LIFO – *last in, first out*. Well, of course he was correct as usual, but...

Huh?

For me this never made sense – only because when I picture the stack of plates I imagine this pile on which we place and remove plates – from the top. The imaginary stack of plates grows upwards, so intuitively we would expect the next push to be to the next higher address. But as many things *computer*, it's exactly the opposite, so I prefer to view it this way: the top of the Stack is the next position at which the data we "push" will go, and each successive push is made to the next lower address within the range $0100-$01FF. The value of the S Register (the Stack Pointer) keeps track of this position. Conversely, each **pull**, which targets data already placed on the Stack, will read from the next position in an upward direction (in memory). We will discuss pulls in a moment because they are handled slightly differently, but for right now let's picture the push.

If you have the chance, take a look at a disassembly of the Commodore 64 ROM, specifically the RESET routine at $FCE2. The very first thing the ROM does (after disabling interrupts) is to set the Stack Pointer to the value $FF, which means the position in memory the next pushed byte will be placed at is $0100+$FF, or $01FF. Recall that the Stack Pointer is actually an offset from $0100. Once we push a byte onto the Stack, the Stack Pointer automatically decrements to $FE, so our next push will be to $0100+$FE, or $01FE. With another push the Stack Pointer decrements once more to $FD, which means the next byte pushed will be placed at $0100+$FD, or $01FD. This process continues indefinitely, right? Well, not exactly.

What happens when we start with a Stack Pointer value of $FF and push 256 bytes on the stack? The Stack Pointer is an 8-bit register with an implied high byte of $01, so it decrements until it reaches $00 and, if once again decremented, it actually wraps back to $FF (keeping in mind an implied high byte of $01). But what about the first push we made to $01FF? The truth is that the data we placed with that first push is still in memory at $01FF, and we can assume the data is important to us, or we wouldn't have put it there in the first place. So what happens if we inadvertently "wrap" the Stack, causing the Stack Pointer to once again have a value of $FF, meaning we placed data to all 256 locations in the range $0100-$01FF? Generally speaking, when this happens we can sum it up with just one word – crash! You can imagine the scenario. But I'll explain it now because it gives us the opportunity to discuss what a Stack pull is.

The Stack Pull

By now we have an idea what happens when we place data on the Stack, and we know the function of the Stack Pointer. Armed with this knowledge it is logical to assume we should match every push with a pull in order to maintain Stack equilibrium. This assumption is correct and the tell-tale sign you have not achieved equilibrium is the crash. Before getting into retrieving Stack data, it should be mentioned again that any data you do place on the Stack is never actually removed. Data in any given location may be over-written, but the Stack is, after all, just regular memory with a very special hardware pointer keeping track of the relative position of our next push. It is the Stack Pointer register value which changes through all of this activity, so as a programmer utilizing the Stack you address data in relative terms as opposed to absolute or indirect addressing as with normal instructions.

To recap, after pushing data on the Stack the Stack Pointer is automatically decremented. Pulling data from the Stack is handled in the reverse order. The Stack Pointer is first incremented,

then the data is retrieved. The data is not physically removed; it is merely "read" from the memory at the address pointed to by the Stack Pointer relative to $0100. It is this concept which can be difficult for the novice to understand. In fact, if you push data to the Stack and do not over-write that data with something else it remains right where it is, unaltered, and could, in theory, be retrieved anytime during the life of a program – and even after a program has finished its execution.

The Subroutine

We have established that the Stack Pointer is incremented by the CPU before a pull occurs, and that for subroutine calls the CPU places the return address onto the Stack (so that the CPU can later find its way back). The CPU places the address of the *last* byte of the JSR instruction onto the Stack, not the actual return address itself, which would be one byte later in the code stream. So, for the instruction **JSR $1234** (stored in memory as $20, $34, $12) the address placed onto the Stack is that which contains the **$12** byte. 65x family processors always store multi-byte data with the low byte in the lower address, the high byte in the next higher address, so the address $C002 would be placed onto the Stack in reverse order as $02, $C0. In plain language this means the CPU pushes the high byte of the return address first, then the low byte. So now the question becomes, "If we come back (return via RTS) to the $12 byte, how do we get to the next instruction that follows?" The answer lies with the RTS instruction, which first increments the multi-byte value it retrieves from the Stack (hopefully it is the return address!). When the return address is pulled (lower addressed byte first) one is added to the value before it is sent to the **Program Counter** (it is pre-incremented). An internal carry flag (not the one we use) is cleared or set based on the result of this first addition. Then the value of the internal carry flag (which is either 0 or 1) is added to the next retrieved byte, or high byte of the return address. So that's it. One is added to the two-byte return address *before* it is sent to the Program Counter in the exact same way we add multi-byte numbers with the **ADC** instruction. Now the Program Counter contains the correct address to continue execution – the address directly after the jump to subroutine instruction.

Stack Instructions

As mentioned, special instructions are used to access the Stack, and the 6502 has a very simple, limited lineup. Pushing data on the Stack is accomplished with the **PHA** instruction, which reads **P**us**H A** Register. This instruction pushes the contents of the A Register, or Accumulator, onto the Stack, and decrements the Stack Pointer. Pulling (retrieving) data requires the **PLA** instruction, or **P**ul**L A** Register. In this case the Stack Pointer is first incremented; then the retrieved data is placed into the Accumulator. As with all data transfers, certain Status Register flags (in this case Z and N) are cleared or set based on the effect the transfer has on the destination register. The same is not true of push instructions as no destination register is involved.

Another type of Stack instruction involves getting or setting the value of the Stack Pointer itself. We will see in **Part Two** just how valuable these next two instructions can be when setting up what is known as a Stack Frame, which is just a fancy name for a place we can store temporary local data. The first instruction is **TSX**, or **T**ransfer **S**tack Pointer to **X** Register, and it does exactly what it sounds like it does. Its complementary sibling, **TXS**, does exactly the opposite. If you managed earlier to pull out your copy of the Kernal RESET routine you saw this one in action. It **T**ransfers the contents of the **X** Register to the **S**tack Pointer.

A final Stack instruction pair is PHP/PLP. These instruction push or pull the Status, or S Register, respectively. **PHP** pushes the values of all status flags, while **PLP** fills the Status Register with the value it pulls from the Stack. Both instructions, while not used frequently, are invaluable because they allow you to preserve the Status Register flags, which is extremely important during interrupt handling.

In Conclusion

Today was all about the 6502 Hardware Stack, but there is so much more to tell, so we'll pick this up again next time when I will show you how to get tricky with some nifty techniques. Did you know advanced programmers sometimes place certain data on the stack to make programs auto-run? Or that the Stack is actually a very good way to pass parameters to subroutines? Or that the 65816 has advanced instructions that permit direct access to the Stack? Or even that Commodore users can very easily create and use local variables with stack frames just like programmers do on the PC? Here's one for you: I have

written programs which did not use a single byte of regular RAM for data! Well, I would like to show you how this is done. I would also like to show how to call a subroutine and then later return to a different, custom address, and how to return from a subroutine that was never called. And as an added bonus I'm going to tell you an undocumented secret about the 6502 that very few people know about. Here's a hint: internally the 6502 processes BRK, IRQ, NMI, and RESET in exactly the same way! I'm going to show you how the 6502 does this – it's information you won't find in any book. We will get to all of this and more next time out with Part Two of this primer about the Stack. So, until our next meeting, take 'er easy, and I'll see you next time right here... at the intersection where Art meets Science.

Tip of the Day

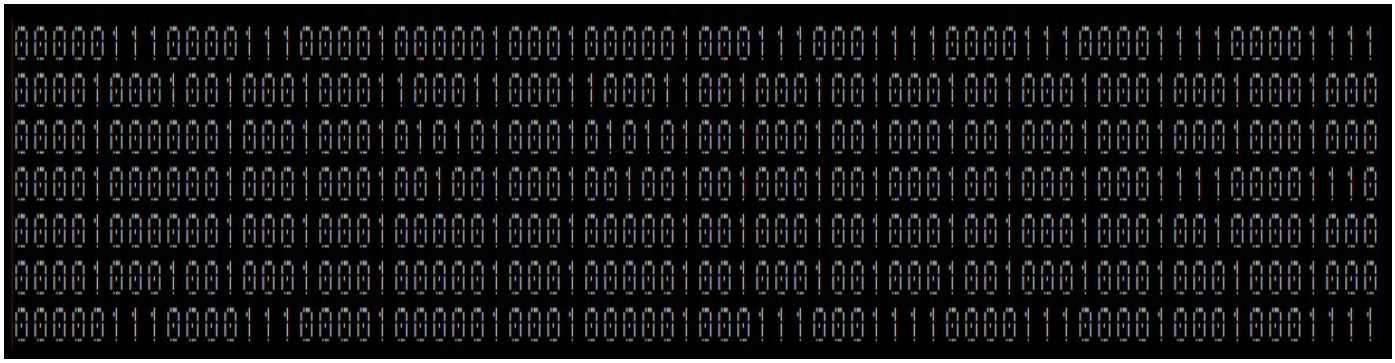
Experienced programmers smartly divide code sections up into separate files that usually concern themslves with accomplishing one specific task. Programs written in assembly language can easily run into thousands of lines of code. A side-effect of all the gained speed and efficiency is that each instruction by itself does very little in the grand scheme of things. Lengthy programs limited to one large file suffer from being hard to read, maintain, and understand. Ever hear the phrase *spaghetti code*? It is probably the worst criticism in all of programming and usually refers to excessive jumps and branches, meandering code paths, incoherence, and of course sloppy programming in general. It is a sign of inexperience, to say the least. Programmers who produce spaghetti code very seldom work well in a team setting, and hardly ever produce optimum code. The tip today is to not fall into the single-file source code trap. It leads directly to spaghetti code. You don't want your friends (or non-friends) leveling this criticism on your code; it is very embarrassing to hear. A corollary to this tip is to split up your code and data sections. It's not mandatory like on the PC, but it is definitely advisable. If you do this you will always know where to find everything and what it all means, even after you forget. And you *will* forget.

Correction: Last month (Assembly Line $02) I made an error that was pointed out to me by my eagle-eyed friend, Arthur Jordison, the author of the excellent IDE named CBM Prg Studio. The last table (right before the Conclusion) was originally written as ...(0*2^6)... and should read:

(0*2^{15})+(0*2^{14})+(0*2^{13})+(0*2^{12})+(1*2^{11})+(1*2^{10})+(1*2^9)+(1*2^8)+(1*2^7)+(1*2^6)+(0*2^5)+(1*2^4)+(1*2^3)+(1*2^2)+(1*2^1)+(0*2^0)

I apologize for the error, and thanks -- that was some good catch, Art. I am truly amazed. Now, don't you have a program to write?

Please send errors, omissions, or suggestions to bert@winc64.com or on Lemon64, username satpro, or at www.melon64.com, username satpro.



Commodore Free Magazine

www.commodorefree.com

Editor
Nigel Parker

Spell Checking
Peter Badrick and Alex Leonardi

Text , HTML & Ebook Conversion
Paul Davis

D64 Disk Image
Al Jackson

ISSUU formatting
Alessandro Di Nepi

PDF Design /Editor /webhost /text collector
Nigel Parker

Website
www.commodorefree.com

Email Address
commodorefree@commodorefree.com

Submissions

Articles are always wanted for the magazine. Contact
us for details .We can't pay you for your efforts but you are safe in the knowledge that you have passed on details that will interest other Commodore enthusiasts.

Notices