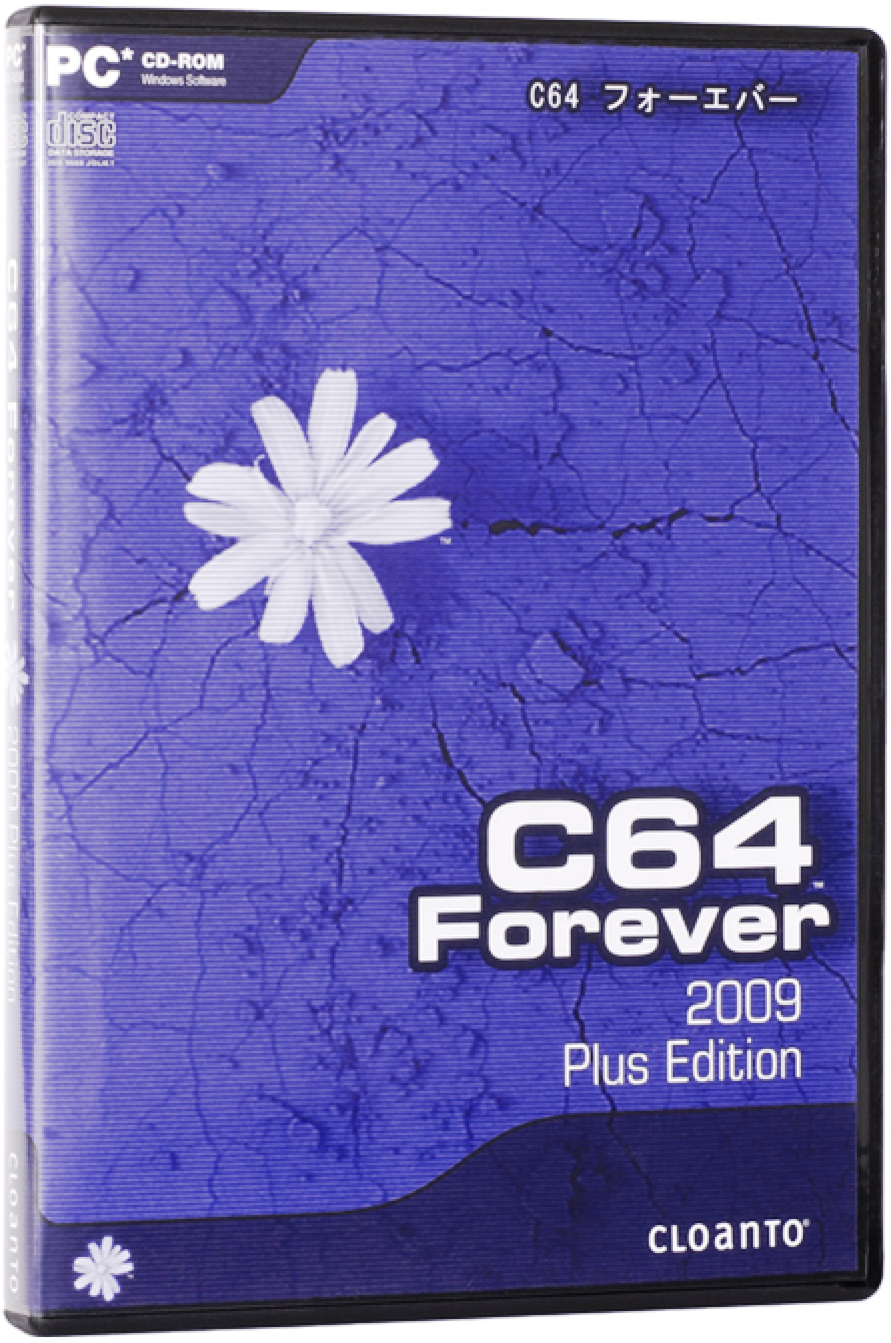
Commodore Free

Issue 31 June 2009

Free to download Commodore magazine
Dedicated to Commodore Computers
Available as PDF Text SEQ HTML and D64 image
www.commodorefree.com



EDITORIAL

Editorial

Some one said better late than never, so here we go then issue 31 of Commodore Free magazine, I would like to thank the people who have emailed in encouragement and suggestions for back problems. I should by now have mailed a response to you, the comments are all welcomed. I would also like to thank

Andrew Fisher	- issue 30 HTML version
DIV8	- Technical support
Peter Badrick	- Spell checking
Al Jackson	- D64 disk editing

And of course
Lord Ronin for his many contributions

Without these people supporting me I think I would have been so overwhelmed I couldn't have continued Commodore Free magazine.

Summer has arrived and I can confirm that by my Computer room being too hot to enter, this is a general test to see if summer has arrived, I haven't managed to enter the room for some 3 months so it was a little moth filled and dusty. We have fresh tarmac outside our house as the road and pavement have now been completely redone thanks to the council I see my voting has made all the difference and the bin collection has returned to Friday that makes it more easy for me to remember to put the thing out for collection. The tarmac in the heat smells fantastic, I wonder if anyone has tried to recreate the smell, an urban myth suggests that using the smell can cure colds and flu, not sure about that but it's a nice small on a sunny day, especially when we have the odd shower that increases the smell of tarmac.

This issue sees an interview with Meeting user through the mail a club designed for Commodore users who can write snail mails to each other and put them in the post box, you know you write on a piece of paper put it in an envelope and put a stamp on the envelope and put it in a post box, someone collect it from the box and sorts al the mail out to its destination, it can take from a few days to several months to arrive.

Also exciting is that Cloanto have finally released the long awaited C64 Forever, some details are in this issue, although as I say it took me so long to finish playing with the release that I didn't get around to creating a review. I have finished a review but will include this in next month's issue, as it looks like I ran out of space.

Great news is that after not checking my inbox for 2 months It seems apart from the usual adverts about chemical companies I have won every major lottery under the planet and just need to click on the links to claim my prize. Of course this is great news but hey just a minute I haven't entered any of these lotteries, ah well it cheered me up a little thinking what I would do with millions of pounds.

That's it from me, I wonder if anyone still reads editorials
Bye

Nigel

Website	www.commodorefree.com
mail address	commodorefree@commodorefree.com

Thanks to

Peter Badrick	Spellchecking
Paul Davis	TXT and Html Creation
Al Jackson	D64 image

CONTENTS

HOW TO HELP COMMODORE FREE

HOW CAN I HELP COMMODORE FREE..
Ok the best way to help would be write something about
Commodore
articles are always welcome,..
WHAT ARTICLES DO YOU NEED..
Well they vary, contact me if you have an idea but I am
looking for..
Tutorials..
(beginners and Expert),..
Experiences..
with Commodore,..
Why I love Commodore machines,.
.**Interviews..**
maybe you have access to a power user.
News
Club meeting
General Commodore news

READERS COMMENTS

Hi Nigel,

I have to thank you for the Forth introduction in Issue 28.Forth is a wonderful programming language and I am deeply surprised that it is available for the old commodore computers. I would like to read more about that in your magazine.

Thx a lot,
Fay.

COMMODORE FREE
Fay thanks yes I enjoyed the forth programming language, it was actually something I could follow quite well and seemed to make sense, I have had 3 other people ask if there was a follow up, so I wonder if this is enough for the writer to put pen back to paper

Dear Commodore Free
Great magazine I have downloaded every issue, Thank you for the wonderful tutorial about Forth do you plan to continue this programming course in future issues?
Keep up the good work
Regard Bill

COMMODORE FREE
Well the credit has to go to Paul Davis he has now started a mini C tutorial for Commodore Free so I guess if he receives enough praise and has the spare time he may look at doing another forth tutorial

Dear commodore magazine
I read the forth tutorial and didn't know that was available for the Commodore 64 do you plan to follow up the tutorial
Regards
Kevin Sprinter

COMMODORE FREE
Kevin thanks for the Comments see above I must thank Paul Davis for his tutorials and general technical assistance with creating Commodore Free. Who knows if they will continue, but the feed back is welcomed

Nigel
Just a note to wish you well and congratulations on the magazine
Best regards
Sue Millington

COMMODORE FREE
Sue Thanks hey I notice no one has complained this month, darn it what am I doing right, ah I will bask in the glory thanks for everyone else who emailed but didn't want to be named take care

NEWS

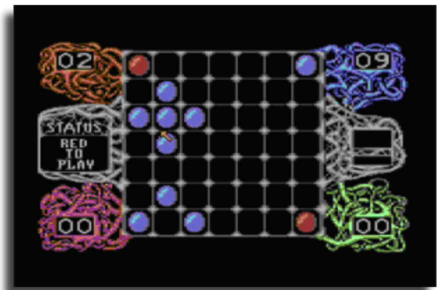
ACID 64 Player Pro v3.00 released

ACID 64 Player Pro v3.00 has been released, containing many updates. The new release permits the user to search through their entire SID collection. The user can now seek through a SID song immediately.

Download the new version at http://www.acid64.com

Games that weren't finds 'INFECTION' - Full game -

Games That Weren't has recovered the full unreleased game called "Infection", the game was originally intended as a 1989 budget release by Virgin Mastertronic. This game eventually turned into "7Up Spot", but the original is arguably a lot better. The additional bonus is the previously unheard David Whittaker tune on the title screen! Enjoy!
And finally, could "Starglider 2" be the next big finding??...



http://www.gtw64.co.uk

The SEUCK Vault Update more new games

Its back to 1990 with FIVE exclusive new games from retro gaming fan Mat Corne, available to download and play for the first time ever from the SEUCK Vault. And we bring you some new SEUCK tips from Richard Bayliss for enhancing your finished game.
http://www.seuckvault.co.uk

From: Andrew Fisher
To: commodore Free
Subject: The SEUCK Vault

For fans of The Shoot 'Em Up Construction Kit, I run a website called the SEUCK Vault. The aim is to gather tips on using the Kit and to publish games created with it. Our latest update includes five previously unseen games from retro gaming fan Mat Corne, and some new tips from Richard Bayliss on enhancing a finished game. We also cover the Amiga version, and hope to include Atari ST content in the near future. The archive will be re-ogranised to make it easier to search for a game. And if you have any experience of using the Kit or games we can add to the site, get in touch with webmaster@seuckvault.co.uk

Contiki for IDE64

A more uploaded more up to date executables from Contiki optimized for IDE64 users has been released. It includes email, ftp, irc, and www clients, and a web server. Also added is the configuration tool by Lodger. The drive numbers are not fixed to 8 anymore, and the file reading has been optimized by using block read, which results in 6 KB/s web serving performance instead of 2.8 KB/s.
http://warez.ide64.org

NEWS

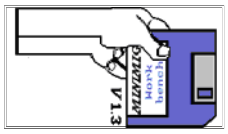
C16-Plus/4 BASIC programming

On the Commodore16.com web page you can now download 4 pdf's about programming BASIC on the C16 and Plus/4. The pdf's contain the following articles:

-Slow and Fast,
-Restore key,
-Window Command,
-Screen Codes / TED Text modes,
-Default and your own Character Set,
-Escape codes,
-ROM, Kernal jump table,
-3-plus-1 and TedMon.

Click here
http://ww.commodore16.com/index.php/component/content/article/51-hardware/251-commodore-plus4-and-c16-hardware-and-advanced-basic-programming-version-101.html

MINIMIG Firmware Update



Hi,
I have just uploaded new firmware for the Minimig. It's still work in progress and not all intended modifications have been already implemented. Nevertheless it's a small step forward. Please report any incompatibility which you encounter.

Main features:
- a lot of changes to improve compatibility (included text file contains more detailed description)
- support for 4 (with ARM mini-board) or 2 floppy drives
- selectable floppy drive speed (double or normal)
- support for hard disk emulation (hardfile support) with soon to be released ARM mini-board
- 28 MHz CPU turbo mode (works also with 16 MHz 68SEC000 parts)
- fast blitter mode
- selectable PAL and NTSC mode from the OSD menu
- OSD menu control with Enter, Esc and arrow keys
- ADF files are not alphabetically sorted but can be selected by pressing a key with the first letter of their name
- support for 2 MB of Chip and 1.5 MB of Slow RAM (hardware modification required)
- scan-line emulation effect

Binary files can be downloaded from here http://minimig.googlecode.com/files/minimig_build_YQ090421.zip (please read the included manual). Sources available from here. http://minimig.googlecode.com/files/minimig_source_YQ090421.zip

A description of how to install extra RAM chips can be found here.

Have fun, Jakub

SVS-Calc 1.0

SVS-Calc 1.0 a spreadsheet system for the Plus/4 has been released. Featuring an intuitive interface, powerful calculation engine, graphics generator, download it from here
http://plus4world.powweb.com/software/SVS-Calc_1_0

2007 CBM engineers Video

-----Original Message-----
From: commodor-bounces@vcsweb.com On Behalf Of rbernardo
To: commodor@vcsweb.com
Subject: VCF East 2007 video of Peddle, Herd, Haynie, Russell

Back in 2007 CBM engineers Bil Herd, Dave Haynie, and Bob Russell gathered at the Vintage Computer Festival East 4.0 in order to celebrate the 25th anniversary of the C64. CBM engineer Chuck Peddle could not attend but had a video conference with those engineers. Originally, Dave Haynie posted video of that day and the conference, divided into 4 parts, at www.youtube.com/user/hazydave

For those who don't prefer the Chuck Peddle video divided into 4 parts at YouTube, BIOS has now posted the entire 1 hour, 57 minute video. Look for "Vintage Computer Fest '07 - Chuck Peddle, Bil Herd, Dave Haynie, Bob Russell" at http://blip.tv/file/2120494

Converted from the original .avi sent by Dave Haynie,
Robert Bernardo (Fresno Commodore User Group)
http://videocam.net.au/fcug
July 25-26 Commodore Vegas Expo
http://www.portcommodore.com/commvex

Commodore 64 Twitter Client

Apparently there is a version of the Twitter client available for the Commodore 64
http://hackaday.com/2009/06/15/c64-twitter-client/

Here is a quote from the website
The last of the Commodore 64's shortcomings has been addressed; it finally has a Twitter client. [Johan Van den Brande] wrote BREADBOX64 for use on the C64/128. It's running on top of the open source Contiki operating system. The hardware is an MMC Replay cartridge with an ethernet adapter. If you don't have the hardware available, you can run it inside an emulator like VICE. Embedded below is a C128D running the program.

Also there is a youtube video
http://www.youtube.com/watch?v=8m86mm-SMGA
I haven't had the time to verify this but it does seem to look genuine

Contiki for IDE64

A more uploaded more up to date executables from Contiki optimized for IDE64 users has been released. It includes email, ftp, irc, and www clients, and a web server. Also added is the configuration tool by Lodger. The drive numbers are not fixed to 8 anymore, and the file reading has been optimized by using block read, which results in 6 KB/s web serving performance instead of 2.8 KB/s.

NEWS

JiffyDOS Licensing

Jim brain working to license Jiffydos follow the discussion here http://www.jbrain.com/

As announced at the C4 EXPO, I am working with Mark Fellows (Highland IT Solutions) to finalize a licensing agreement for 'JiffyDOS' ROM Overlay manufacture and distribution. In addition to hardware ROM enhancement units, I will also offer image downloads for 1541 Ultimate, C64DTV, and emulator users, as well as an amnesty offering for unlicensed copies.

To minimize manual manufacturing processes inherent in the current EPROM-based JiffyDOS offerings, my goal is to utilize the ROM-el EEPROM/FLASH solution for JiffyDOS hardware offerings

VIC 20 releases

http://sleepingelephant.com/ipw-web/bulletin/bb/viewtopic.php?p=44123#44123
Name: **Omega Fury**
Author: Robert Hurst
Released: June 2, 2009
Requirements: VIC 20 with 16k memory expansion
Description: The improvised sequel to Bally-Midway's Omega Race. An arcade-style space shooter making use of the VIC Software Sprite Stack - 4th edition. Includes source and documentation to the story line and game instructions. The C= Easter egg is also included.
DEMO video: http://www.youtube.com/watch?v=E8bkwvcDXAc
Download the demo http://robert.hurst-ri.us/files/omega-fury.zip

Lower Bucks Computer Users Group for Commodore 64

From B Degnan

I have started archiving and uploading the entire collection of diskettes from the Lower Bucks County C64 User Group, a club based near where MOS and Commodore once operated in Pennsylvania. This archive is a glimpse into a pre-WWW electronic community. When I am done the entire collection of disks, starting from #16 (Jan 1986) through December 1991 will be available. I have uploaded so far the first three years. To use these disks, you will need a way to download and extract D64 (C-64 disk images). If anyone has disks 1-15 I would greatly appreciate D64 copies if they're available.

LUG
M000--M015 (missing)
M016--M029 1986 (there was no M021 or it is missing)
M030--M041 1987
M042--M053 1988
M054--M065 1989
M066--M077 1990
M078--MO88 1991 (with some extra disks)

DiskArt Images - 1987 by Those Designers
DiskArt[x].D64 - file names based on versions printed on disk labels

C00[x] Compute! programs pertaining to C64
C001 - 11/85, 12/85, 1-86
C002 - 2/86, 3/86, 4/86
C003 - 5/86, 6/86, 7/86
C004 - 8/86, 9/86, 10/86
C005 - 3/85

http://vintagecomputer.net/commodore/64/LowerBucksUserGroup/

updated HyperSID

"HyperSID is a subtractive synthesizer with all the C64 SID chip hardware capabilities besides many new software based features So you can take advantage of various Controlling features of a VSTi and also real analog sound with SID character.Real time integration between software and hardware makes HyperSID act like the other VST instruments in your host application"

HyperSynth has announced that it has updated HyperSID, and the associated hardware unit OS, to v1.2.

OS v1.2 has been completely redesigned and some long standing issues have been fixed. In this release all the processing tasks are handled by HW unit. The big flaw in the previous versions was the low update rate of modulation sources but now the HW unit is capable of filling SID registers with update rates up to 28000 Hz.

The VSTi plug-in represents a step forward in controlling the new LFO and envelope modules with great preciseness and resolution up to 14 bit for each parameter plus an ultra fast arpeggiator module to emulate C64 like sound. Due to many requests, HyperSynth has published "HyperSID MIDI CC-NRPN Chart" in the website as a document, and suggests that "it's time for Mac users to start building their own control surface".

Additions & Fixes:
-Added advanced LFO (freq=0.01-30.00Hz with 0.01Hz resolution per step).
-Added new filter envelope (range=0.001-4s with 0.001ms resolution per step).
-Added new envelope2 which can modulate Pulse Width,Pitch and LFO rate.
-Added ultra fast arpeggiator which is capable to produce classic and SIDish arp up to 4000bpm tempo.
-Added save and restore for MIDI-out device.
-Added auto preset update after plug-in startup.
-Fixed a problem with displaying MIDI device names more than 26 characters.
-Fixed OSC3 synchronization bug.
-Fixed a problem that was causing the played note to be sustained or muted when using pitch wheel.
-Fixed unwanted note sustain when switching from mono to poly mode.
-Fixed a bug in processing note off message with zero velocity.
-Fixed FPB button inverse function.
-Fixed transferring FPB(Link to) status to HW unit while sending a preset.
-Fixed a problem in SID oscillators noise waveshape that deselected other waveshapes.
-Fixed conflict in MIDI device which caused freezing in Nuendo and Cubase.
-Migration from non-standard MIDI messages to NRPN for parameters that need 14bit resolution.
-Updated all parameters names to the new recognizable groups.
-Improved preciseness while responding to host automation.
-Improved frequency display for SID filter and PW.
-Huge GUI redesign.
-Cured keyboard problem which did not respond to mouse click.
-Preset manager drop down list is now limited to the plug-in window.
-2x faster startup and much less CPU usage for plug-in.
-Redesigned factory presets.

http://www.hypersynth.com/hypersid.html

C64 emulator for iPhone

A C64 emulator has been developed for the iPhone, but unfortunately the sdoftware has been rejected by Apple because they say it violates the SDK agreement
http://www.pocketgamer.co.uk/r/iPhone/C64/news.asp?c=13974

NEWS

Commodore 64 forever

C64 Forever: Click to Play

When the C64 was launched in 1982 it immediately set the standard for 8-bit home computers. Its low cost, superior graphics, high quality sound and a massive 64 KB of RAM positioned it as the winner in the home computer wars, knocking out competitors from the likes of Atari, Texas Instruments, Sinclair, Apple and IBM.
Selling over 30 million units and introducing a whole generation to computers and programming, the C64 shook up the video games industry and sparked cultural phenomena such as computer music and the demo scene. In recent years the C64 has enjoyed a spectacular revival manifesting itself once again as a retro computing platform.

To allow you to experience and relive the wonders of this unique computer, Cloanto, developers of Commodore/Amiga software since the 1980s, has introduced C64 Forever, a revolutionary preservation, emulation and support package. C64 Forever embodies an intuitive player interface, backed by a built-in database containing more than 5,000 C64 game entries, and advanced support for the new RP2 format, dubbed the "MP3 of retro gaming".

LINKS
C64 Forever Home Page
http://www.c64forever.com

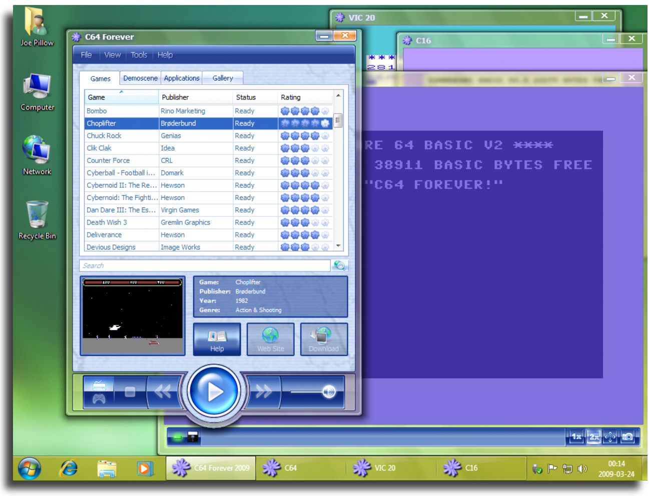
Screenshots
http://www.c64forever.com/screenshots/

Photo of Boxed Version
http://www.c64forever.com/editions/plus-boxed/

HD Video: C64 Forever Live Introduction
http://vimeo.com/4910924

RP2 Format Information
http://www.retroplatform.com/kb/15-122

Commodore Free
More information and a review next month, I spent so long playing I ran out of time to review the item.



Game	Publisher	Status	Rating
Bombo	Rino Marketing	Ready	●●●●○
Choplifter	Brøderbund	Ready	●●●●●
Chuck Rock	Genias	Ready	●●●●○
Clik Clak	Idea	Ready	●●●○○
Counter Force	CRL	Ready	●●●○○
Cyberball - Football i...	Domark	Ready	●●●○○
Cybernoid II: The Re...	Hewson	Ready	●●●●○
Cybernoid: The Fighti...	Hewson	Ready	●●●●○
Dan Dare III: The Es...	Virgin Games	Ready	●●●●○
Death Wish 3	Gremlin Graphics	Ready	●●●○○
Deliverance	Hewson	Ready	●●●●○
Devious Designs	Image Works	Ready	●●●○○

Game: Choplifter
Publisher: Brøderbund
Year: 1982
Genre: Action & Shooting

```
RE 64 BASIC V2 ****
 38911 BASIC BYTES FREE
"C64 FOREVER!"
```

Cloanto Releases Amiga Forever 2009

July 2, 2009 - Cloanto released today Amiga Forever 2009, the latest version of the award-winning Amiga preservation, emulation and support suite for Windows and other platforms. Amiga Forever 2009 (www.amigaforever.com) is the most refined update ever released in the Amiga Forever series. It again sets new references in usability while more than doubling the featured content and providing easier access to a universe of free downloads. Like its sister product C64 Forever, Amiga Forever 2009 introduces a synergistic combination of the RP2 retrogaming file format, RetroPlatform Library to recognize content, and RP2 Manager to export and import to and from other file formats.

The 2009 version includes hundreds of enhancements and was tested to comply with "Compatible with Windows 7" requirements. New features include seamless, one-click integration of the AmiKit and AmigaSYS add-ons and new system ROMs (e.g. the 0.7 Kickstart required to run the software used by Andy Warhol's famous 1985 Launch of Amiga demo). The Plus Edition includes more than 100 games and a selection from the most beautiful demoscene productions ever created. Tens of thousands of games, as well as other software ranging from the oldest to the latest Amiga releases are only a mouse click apart, and can run on powerful PCs and inexpensive netbooks alike.

When both Amiga Forever and C64 Forever are installed, the players share data and software modules with each other, uniformly playing back RP2 games of both 8-bit and 32-bit platforms. The same familiar user interface and settings are also applied to saved states, disk write undo, dual-monitor setups and other advanced functionality. "The way it brings back memories is amazing, and it feels as if Commodore and Amiga were united again," said Michael C. Battilana of Cloanto. Dave Haynie, a long-time user of Amiga Forever and former Commodore International chief engineer on high end and advanced projects, added: "The past IS the future... Amiga Forever is the single best way to run AmigaOS today. Your PC is the fastest Amiga that will ever exist, and Cloanto does a wonderful job of packing the emulation technology, every AmigaOS ROM and Workbench version (most of which I've long since lost to data rot on my floppy collection), games, utilities, Web tools, graphics programs, most anything you would need, all in one place."

Amiga Forever 2009 is available now in three editions:
- Value Edition (downloadable installer for Windows systems)
- Plus Edition (downloadable CD ISO image with additional Windows and platform-neutral content)
- Premium Edition (physical Plus Edition CD and two DVDs, plus instant download of the installer for Windows)

The Plus Edition includes the KX Light boot environment with barebone PC hard disk installation and online updates. The Premium Edition additionally contains more than five hours of videos.

Link to the shop http://www.amigaforever.com/shop/
Amiga Forever website http://www.amigaforever.com/

Commodore are Back AGAIN!

Text from the website
http://www.commodoreworld.com/index.html

We're back! Commodore was a pioneer in the computer industry bringing to market the Commodore C64 which was the most popular computer model of its time. Between 1982 and 1994 17 million units of the Commodore C64 were sold. Commodore is back and better than ever and we've made it our business to provide our customers with cutting edge technology in our new mini Netbooks. Experience the Commodore Evolution! Commodore's new line of netbooks are perfect for kids, home makers, outdoor enthusiasts, business travelers, or any active person.

The netbook's have a lightweight & ultra-portable design that keeps you connected everywhere. The innovative designs are packed with features for functionality, wireless communication, and multimedia entertainment.The Commodore netbook has a sleek & modern look & feel and is available in a wide range of colors.

THE COMPANY
The basis for Commodore International Corporation was laid in 2004, when SATXS Communications BV, a pioneer of digital distribution and live streaming, joined forces with Yeahronimo NV, the first legal download platform in the Netherlands, to form Yeahronimo Media Ventures Inc (YMV). The aim of this merger was to integrate advances in content provision with state-of-the-art digital distribution services. Consumer electronics hardware was added to this mix when YMV acquired the worldwide rights to the Commodore brand name, assets and patents from Tulip Computers early in 2005. This move marked the official start of the Commodore International Corporation.

The essence of the Commodore brand is personalization. Back in the 1970’s and 1980’s Commodore was the first to offer ordinary people computers that they could adapt to their own individual needs and preferences. They could modify graphic interface, create their own music, write their own programs, and play the video games they wanted whenever they chose. Hardware, software, and content could be modified and managed by the user as they wished. This focus on personal freedom went hand in hand with an emphasis on community. Commodore’s revolutionary technology inspired users to share their knowledge and content with others.

Commodore International Corporation is dedicated to building on the strong heritage of the Commodore brand. As a company, our values are those that made Commodore so successful: User-friendliness, an open platform, and unprecedented opportunities for personalization. Check out our products section to see how the Commodore International Corporation is translating the spirit of Commodore for the market today and tomorrow.

Commodore Heritage
The Commodore brand dates back to the 1950’s, when Polish immigrant Jack Tramiel started a small typewriter manufacturing business in Toronto, Canada. As competition in this market increased and consumer needs changed, the company’s focus shifted to adding machines in 1962. Shortly after, they switched to electronic calculators which were considered revolutionary in the late 1960’s.

When the computer age dawned in the 1970’s, Commodore was once again on the cutting edge. Tramiel changed the company name from Commodore Business Machines to Commodore International, Ltd., and moved operations to the high-tech hub the US state Pennsylvania. Here he employed a brilliant computer engineer named Chuck Peddle.

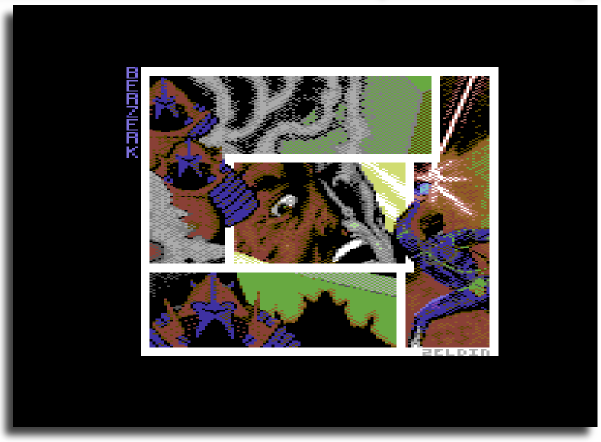
Ignoring the conventional parameters of his time, Peddle put together what is now recognized as one of the first consumer-ready personal computers: the Commodore PET (Personal Electronic Transactor). Its release in 1977 established Commodore as a major name in the home computer market. Further innovations followed. In 1981 Commodore introduced VIC-20, the first color computer at a consumer friendly price. This paved the way for the development of the Commodore 64 personal computer in 1982.

The Commodore 64 (commonly known as the C64) is still recognized today as one of the best-selling computer models. Between 1982 and 1994 at least 17 million units were sold. The unique features of this PC, which could be plugged into a television set for video gaming, revolutionized the concept of entertainment in the 1980’s. Versatile and affordable, with sound and graphics capabilities way ahead of its time, the C64 brought home computing and personalized entertainment within the reach of almost everyone. Before long enthusiast started to develop and share C64 software and games. A community of Commodore fans and a legend was born.

BERZERK REDUX
INTERVIEW WITH MARTIN PIPER

By Andrew Fisher

http://noname.c64.org/csdb/release/?id=80567



2009 has seen two fantastic game releases based on early 1980s games – Fortress of Narzod was converted from the Vectrex, and now the arcade classic Berzerk has made the jump to the C64. Commodore Free talks to programmer Martin Piper.

Q. How did you become interested in converting Berzerk to the C64?

I'm relatively old, so I can remember the arcade machines. So when Trip6 started this thread http://noname.c64.org/csdb/forums/?roomid=11&topicid=58028&showallposts=1 I was interested to see what he had actually got. When it turned out to be C sources it became obvious it wasn't really the arcade version of Berzerk which uses Z80(ish). So I looked for the 6502 Atari version instead and did a really quick hack to get it to partially run on the C64. But the Atari code spent most of its time in a video update loop so it was too slow on the C64. Also the Atari version lacked speech and game play aspects from the arcade version so I shelved it.

Then when nobody else came up with a version I thought to myself that I would quickly test out the basic technical requirements to get a feel for how long it would take to do a new version. Chiefly these were:

1) Getting NMI samples working with the multiplexor.
2) Then seeing how many segments of speech could be squeezed into realistic memory conditions.
3) Maze generation that did not produce a blocked route for the player.
4) Pixel accurate collision detection or background and sprites (just like the arcade) using the VIC registers.

That went surprisingly quickly and the game play then naturally came along.

The polish of the game, adding Zeldin's excellent artwork with sound effects music and hi-score saving, actually took longer to code than the main game. :)

Q. Did you make use of the source code from other machines?

In this version no, it's all coded from scratch or using existing libraries such as the multiplexor, sound etc.

Q. Did you program it on a real machine or emulator?

99% of the work used VICE because it has a couple of my hacks (submitted to the main tree now) to allow it to load labels from ACME (also with some of my hacks) and show extended register information in the CPU execution history. This aids development because it lets me debug easier than a normal C64. The Alt+W warp speed feature in VICE also helps improve development time because I no longer had to wait for decompression or loading when testing each new code iteration. The downside of such development is that now sometimes I find myself pressing Alt+W when I am web browsing or waiting for a large C++ project compilation to finish... Only when I wanted to check everything really works, like the updated VIC collision register code, did I test on a real machine. I also tested on Hoxs64 and CCS64 just to be sure.

Q. What other utilities did you use to create the game?

ACME (with some label producing hacks), my compression utility (on codebase64), CharPad and SpritePad. I gather Zeldin used Koala Paint.

Q. How did you transfer the speech samples and play them in the game?

I found the samples from the old MAME version at 22kHz 8 bit mono, down converted them to 6kHz and then converted them to 4 bit samples using a quick hack custom tool.

Q. Have you played a real Berzerk arcade machine in the past?

Yes, once or twice in the dim past. Mostly I used MAME to check the game play with this conversion.

Q. Will you be trying to create an "enhanced" version as some people have suggested?

Yes. Much better graphics are planned. I also want to try to improve the speech a bit more by getting closer to the raw data using the LPC decompression that MAME now uses. Game play will be changed a bit as well. With better graphics the "bullet proof bow tie" might have to go as there won't be a gap between the head and shoulders for the bullet to pass through. :(I'm half thinking of having some boss robots every few rooms as well, maybe locked doors that only open after several waves of robots are cleared, two player, etc. We will see. :)

Q. Why did you decide to give the bounty to pay for CSDb's hosting costs?

I like to be able to help the community when I can and this version was only really a few days hacking around on something I found interesting. It is a welcome rest from my normal heavy duty C++ programming. :)

Q. Are you working on any other C64 projects at the moment?

Mainly tweaking common libraries here and there which are all available for anyone to use http://codebase64.org/doku.php?id=projects:resurrection
I'm trying to find sources to resurrect a couple of old projects I was working on many years ago. But my work disks from back then are sadly not well documented.

Thanks to Martin for answering my questions, and for all his coding efforts on Berzerk.

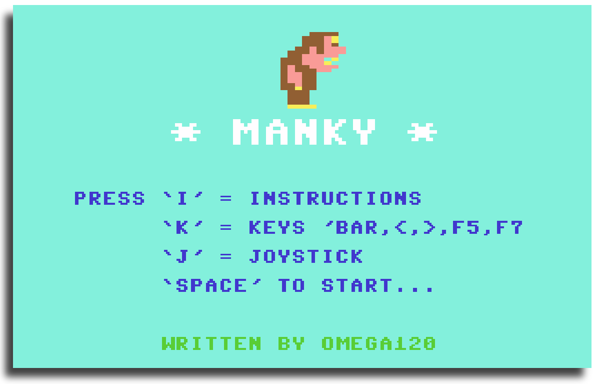
game new review MANKY

By Richard bayliss

http://www.lemon64.com/?mainurl=http%3A//www.lemon64.com/reviews/view.php%3Fid%3D928

downloaded this game to check it out and I decided to write a review for it. First of all the game is frozen with the Action Replay cartridge, as it was programmed with an Action Replay cartridge M/C monitor. It is quite interesting that people are actually still using machine code monitors in Action Replay rather than assemblers or cross-assemblers, but that does not really matter to me. We have a choice. Anyway, on to the game.

First of all you are presented with a cyan screen with large writing that says Manky with a sprite of a gorilla right in the centre above the name. There are also some game options you can choose from. You can select to play from keyboard or joystick then press space bar to play the game, or just press 'I' to read the instructions.



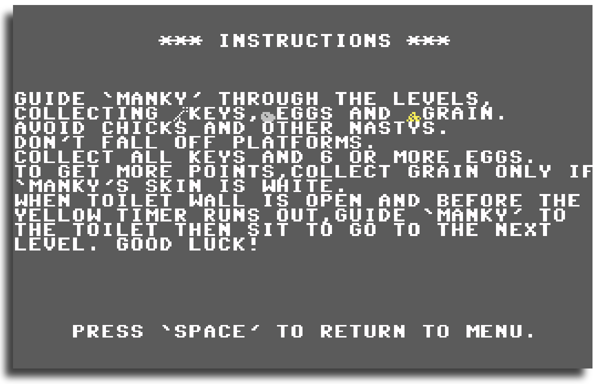
```
* MANKY *

PRESS `I' = INSTRUCTIONS
      `K' = KEYS 'BAR,<,>,F5,F7
      `J' = JOYSTICK
      `SPACE' TO START...

      WRITTEN BY OMEGA120
```

When I read the instructions (After pressing 'I') the screen's colour turns grey and with white writing. The game instructions look extremely untidy. To my opinion there should have been two paragraphs. The first paragraph should read "Guide Manky through each level collecting keys, eggs and grain. Avoid chicks and other nasties and don't fall off the platforms. Collect all keys and six or more eggs. To get more points, collect grain only if Manky's skin is white."

Then a new paragraph which should read:
"When the toilet wall is open, guide Manky to the toilet then sit on it to move on to the next level. You must make it before the time runs out. Good luck!"



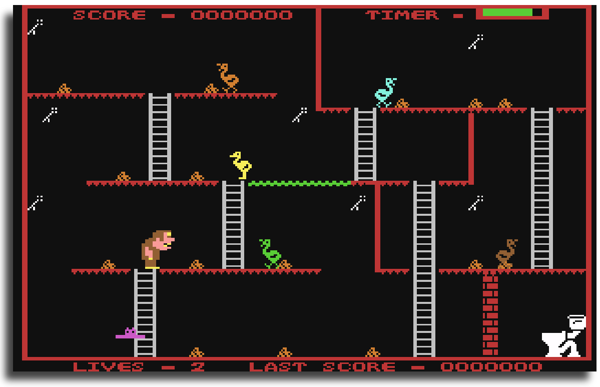
```
*** INSTRUCTIONS ***

GUIDE `MANKY' THROUGH THE LEVELS,
COLLECTING KEYS, EGGS AND GRAIN.
AVOID CHICKS AND OTHER NASTYS.
DON'T FALL OFF PLATFORMS.
COLLECT ALL KEYS AND 6 OR MORE EGGS.
TO GET MORE POINTS,COLLECT GRAIN ONLY IF
`MANKY'S SKIN IS WHITE.
WHEN TOILET WALL IS OPEN AND BEFORE THE
YELLOW TIMER RUNS OUT,GUIDE `MANKY' TO
THE TOILET THEN SIT TO GO TO THE NEXT
LEVEL. GOOD LUCK!

PRESS `SPACE' TO RETURN TO MENU.
```

After pressing the space bar to return to the options screen, you can go on to the game. After pressing the keyboard or joystick option key, which can be annoying. I would prefer the options to be automatically highlighted before playing the game. Now it is on to the game.

The first game screen starts with Manky at the bottom left of the screen, with various ladders which the poor chap has to climb to grab the keys. Also in the game screen, there are various enemies which are trying to stop you. They are the chicks (looks like sprites ripped from Chuckie Egg by A&F Software) and also bats. Looks like a fun game a bit like Chuckie Egg (One classic game that I really enjoyed in my childhood memories). When I control Manky with a joystick in port 2, he moves left, right, up and down okay, but when I try and jump, using the fire button. The jump moves are terrible. The player appears nowhere from the top of the screen and then lands to the ground, which games like Chuckie Egg makes the player jump up and down. This is the major drawback to the game. So if you tried to jump over obstacles, you would find it virtually impossible to do that. I find it impossible to go to the toilet, due to



the poor game play implemented caused by the jumping. Also why does Manky not fall straight to the bottom and die? Animation of the death (Or maybe a white Manky with a halo floating up the screen after falling or collision with bad things would do the trick.) Basically the game itself is just too hard to play, and addictive level would unfortunately be extremely low. Probably one or two plays of this game would put people off. No matter how hard I try to complete a level without cheating, it is literally impossible. Even on a real Commodore 64. Therefore game over is too quick.

On the graphics front, the game looks like a very 1984/1985 platform style game. The chicks look as if they were ripped from Chuckie Egg (EDITOR COMMENT see the amendment at the end of this text)also the design for each level is quite interesting. Presentation is terrible, the front end is extremely basic, but it is good that the player can choose from the game options to have keyboard or joystick control.

Sound? Well, there is not much sound, just an annoying noise for when the player is moving. The sounds are very basic. I would have loved to hear in game music, but not the basic kind.

Overall, this game is really difficult lacking presentation, sounds, music, and game play. The major flaw to this title is of course the jumping controls and the lack of animation, like deaths, and also lack of proper game actions. Manky feels literally unfinished. It probably might have been a nice game if the jumping control was fixed. If this game was going to released commercially way back in 1984/1985, software houses would say No thanks.

It would be very nice if the author of this game, updated this production to allow the player to jump and also fall properly and also add a death sequence (Manky floating upscreen like an angel) and also update the sound effects. If he manages to perform this task, then this game would probably be much better to play.

AMENDMENT
I apologized to him about the Manky review where I said that his game's chick sprites were ripped from Chuckie Egg. After I got a response from him, I realized that I made a big mistake. All work was done by OMEGA120. He originally hand-pixelled the sprites, before he turned it into computer data.

Please feel free to reprint the review if you like, but please add at the end about my mistake (The sprites) and also that he's working on improving the game, including the jump routines

Regards Richard

Interview with MANKY creator Peter Courts (OMEGA120)

http://peterscommodore64.yolasite.com/

CF. Please introduce yourself to our readers

PC. I am Peter Courts known by my handle as omega120

CF. Can you tell our readers about the game MANKY what sort of game is it

PC. The game is for the commodore 64 it's a platform game in the style of "Chucky egg"

CF. what year was MANKY created

PC. I created the game about 6 years ago

CF. MANKY has received some harsh comments, does this upset you

PC. no, the more comments the better, this helps me to improve and add to the game

CF. You have created other games that are on the website to download can you tell our readers a little about each one

PC. Some of the games listed for download are Moggy, Moggy 2, Mr Wong`s Loopy Laundry ,Mr Pilic which is still has to be finished and Manky v2 2009 which I am currently working on

CF. You have received some comments about using Action Replay to code rather than an assembler package can you tell our readers why you use action replay, what are the advantages

PC. This is really just a option that I prefer using, I don't look for advantages, its just what I am used to

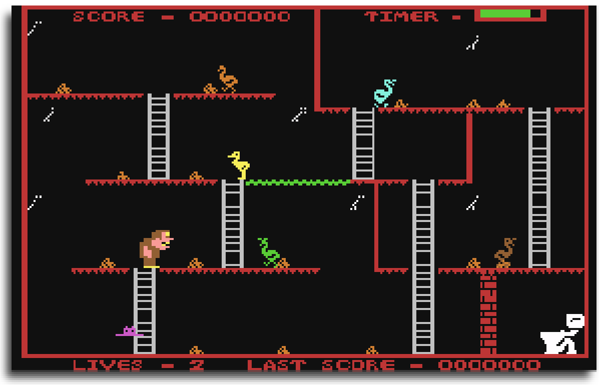
CF. Was the whole of manky coded in AR even the sprites?

PC. Yes you are correct; all sprite data numbers were inserted manually one by one in the memory bank

CF. What about 2009 MANKY v2 do you have any date for completion or are you just working a little on their in your spare time

PC. Currently I have no set date, it may be released around January 2010 but then again it could end up or April 2020

CF. on your website http://peterscommodore64.yolasite.com/ you teach people to code a using action Replay, I presume other coders have commented about this to you.



PC. Yes I guess I receive mixed comments

CF. Do you know anyone else who uses Action Replay to Code games or demos?

PC. No I don't

CF. Apart from Manky v2 do you have another games planned

PC. no, nothing in the current future is planned

CF. Have you thought about porting MANKY to any other formats of machine

PC. yes, I actually considered porting the game to the Commodore +4

CF. I see you website http://peterscommodore64.yolasite.com/ is fairly new and you have added a forum recently, do you have other plans for the website

PC. I am always adding and amending to the site

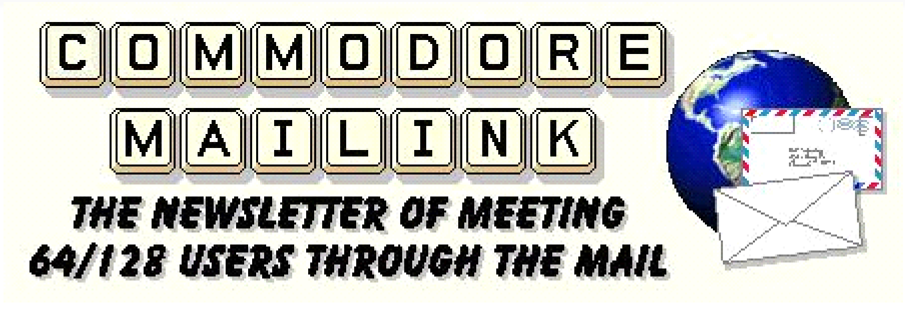
CF. I also see LEMON 64 has added MANKY to there database after user demand does this make things more worthwhile?

PC. I think the game was added in the database just for a topic to joke about

CF. Thanks for your time,

Interview with Rob Snyder

President of Meeting 64/128 Users Through the Mail



CF. Please introduce yourself to our Reader

Rob Snyder Meeting Users Through The Mail (MUTTM)
As for myself, I am 44 years old, married, and father to my five children. I work as a rural mail carrier for the United States Postal Service and live in rural Ohio in the Midwest of the United States of America. I have used a commodore computer my entire computing life only dabbling on other platforms-- my wife has a Mac.

CF.
What was your first encounter with Commodore Machines?

MUTTM
My first encounter with a commodore computer was in 1982 when my brother Dan and I pooled our savings and bought the $595 Commodore 64 and a $60 datasette to enter the home computing world. In 1992, I married, and took a third part time job as a bookkeeper for our parochial school. The school also used commodore 64s and they introduced me to the local commodore club. After a few years, the Commodore Computer Club of Toledo (CCCT) made me their President and I have been ever since.

In the November 1998 issue of the CML, I received a hello and big welcome as one of the seven new members of MUTTM. I have created several issues of our local newsletter when I decided to take a turn as guest editor of CML. I had been editor for three issues over the years when in the summer of 2006, Linda Tanner, then President of MUTTM, emailed me about taking over the role of President. I was excited, intimidated, proud, and honored to guide the correspondence user group of commodore

CF.
Can you tell our readers about Meeting user through the Mail (MUTTM)

MUTTM
Meeting 64/128 Users Through the Mail is a non-profit, group of commodore users that correspond through the mail (and email) an international commodore users group of sorts. To become a member, you need to fill out an application. To get an application, just contact the President through email or regular mail giving your postal address to receive the application.

The website has the information too, but since we are a correspondence club, we continue to have the mail our main submission tool. The application lets you tell a bit about yourself, such as address, commodore equipment, interests and hobbies. You send the application with the membership fee to the club. The information is used in your Bios which is sent to other club members only.

CF.
What benefits does the user gain from joining MUTTM, I know there is a newsletter sent 6 times per year

MUTTM
I feel the main benefit of MUTTM is the BIOS. The Bios is THE LIST of MUTTM members. It includes names, addresses, equipment used, interests and other hobbies of our members. While not compiled for mass mailings and other solicitations, the list is for commodore users to contact other commodore users about computer interests and other mutual interests of the correspondents.

Over the years, there have been many friendships which began from the initial contact derived from the Bios. As for other benefits of the club, the Commodore MaiLink is what many members join to receive, as it was only available to members. This year, we are making the newsletter available on the internet hoping to attract more like people to become members. I also feel with the lack of commodore 64/128 specific publications, C= users are at a loss for articles about their favourite machine. Making CML more accessible, I hope to fill that loss.

CF.
What is your involvement with MUTTM, and what input do you give to the Magazine?

MUTTM
As President, my involvement is mostly making sure the club keeps its focus; and keeps going. Keeping members connected through the Bios and CML articles is what I see as MUTTM`s mission. Jean Nance once said she wanted to give members everything promised instead of promising a lot. We promise 6 well-produced newsletters, and a list of fellow commodore 64/128 users.

I believe Jean also had final say into what was left in or out of the newsletter, as all material used to be sent to her even after she quit editing. I don't have that power as I don't usually see the whole newsletter until I get it in the mail. I believe as the 23 years the club has been around, most members know what can and shouldn't be in the newsletter. I hope nobody abuses that trust. As for my other duties, I write the Meeting News; included in every issue which is just were the President get to sound Presidential.

I also like to edit an issue a year. In addition, I took on the responsibility of Bios editor and Treasurer as Brian Vaughan and Emil Volcheck respectively stepped down due to health issues. I took these posts as the membership is not as large and thus not as cumbersome as in years past. I feel I can effectively do these tasks to keep the club going healthily. When we found our family was due to have another member, I did worry about time for club activities. Even with the busy time of membership renewals and the birth of Luke (along with the holidays) nearly coinciding, all seem to have worked out. Except for finishing this questionnaire ;)

CF.
Where do articles come from for the magazine?

MUTTM
The club newsletter, the Commodore MaiLink, as named by Eloise Carey, mainly has articles from our members and is edited by a guest editor who (for the most part) volunteers. Most issues are produced using only commodore equipment. While some of the guest editors use other equipment to assist in the newsletter, we pride ourselves on the best newsletter made FROM our 64 or 128 computers. As Jean Nance stated in the Guidelines For Commodore MaiLink Editors, Members want to see what can be done with their Commodores, not what can be done with other computers. Today's newsletters happen quite a bit different than several years ago. Then disks and printouts were sent via regular mail. It seems today's articles are sent via the internet as PDF or email text. The text for the MaiLink on Disk is mostly from email text or OCRed into the electronic form. Most articles still come from MUTTM members and luckily we have a few members who continually write articles for the MaiLink.

I continually try to solicit more members to contribute. Richard Savoy is our member who prints and mails CML for the rest of the members; truly a large job. As I understand, he has a couple copy machines, B&W and colour, in which to multiply the pages of CML. Along with stuffing envelopes, weighing, and mailing, he also makes available bonus disks he has received from other clubs to members who ask and pay for the added postage. As for a specific process in CML production, here goes:-

The Guest editor collects and organizes articles, prints out needed pages and sends them to Richard who also collects some regularly occurring articles. Richard copies and mails CML. Richard also OCRs some articles for Ken Barsky who produces the MaiLink on Disk. While having the text of the newsletter, the MaiLink on Disk also contains many programs selected by Ken. He works very hard to also create a first class product.

CF.
You don't have internet access at home; Does this cause you problems with the club?

MUTTM
Not having internet access at home is a hindrance especially since taking over as President. Being a correspondence club, a lot of club business happens via the internet. I did have access about five years ago but gave it up for economy. Now I am not sure I want the access with the children getting of age. Not that the internet is bad, just one more thing I would have to limit; like I do with television and candy. To use the internet, I presently drive to the library.

Some years ago, a MUTTM member wondered about PayPal to pay MUTTM dues and I used my private account to accommodate this member. This past year I set up a new PayPal account just for the club but had no takers. I'm afraid the PayPal account is not advertised yet but I hope all who would like to use PayPal would contact the President to get specifics to use the service. We do use PayPal to pay bills such as our website expenses on Videocam.net.au.

CF.
I noticed in the history of the club that the largest number of members was over 300, do you think there is still that much interest or has the internet and the start of free publications like "commodore Free" limited the membership ?

MUTTM
MUTTM`s history is more detailed on our website, but the club started in 1986 by Kirby Herazy, a high school student. He listed the club in the national Commodore magazines of the time to have other commodore users write to him. He shared his list of about 30 Commodore pen pals with his other C= pen pals. Jean Nance, one of the pen pals, took over the organization after Kirby wanted to spend more time on other projects and school. She made MUTTM what it is today.

She started a newsletter and asked for dues to cover printing and postage. After a letter appeared in RUN magazine about the group in 1988, a flood of inquiries and members raised the membership to a high of around 300 in the early 1990s. The numbers have been declining; presently at 49 for 2009, but still more than when Kirby thought the club was getting too big to handle.

CF.
How long do you think the club and newsletter can survive?

MUTTM
As for how long MUTTM can survive, at least as long as I do. As long as there is one MUTTM member, I will continue to serve that member-- to help find other commodore 64 or 128 users to correspond with

CF.
Have you read Commodore Free magazine, how do you feel about the creation of a free magazine for commodore users?

MUTTM
Having a subscription to the Homestead List on Videocam.net.au, I got to know about Commodore Free and have read many issues. I enjoy the magazine and commend you for your effort. I only wish I had more time to read more issues. I do not see your publication affecting our membership other than in the positive, as I feel most of our members enjoy the ease of getting their issues through the mail. Matter-of-fact, only one member has expressed an interest to receive their issue of CML via PDF email. I see Commodore Free as keeping and building interest in our commodore machines and that can only be beneficial to all commodore enthusiasts

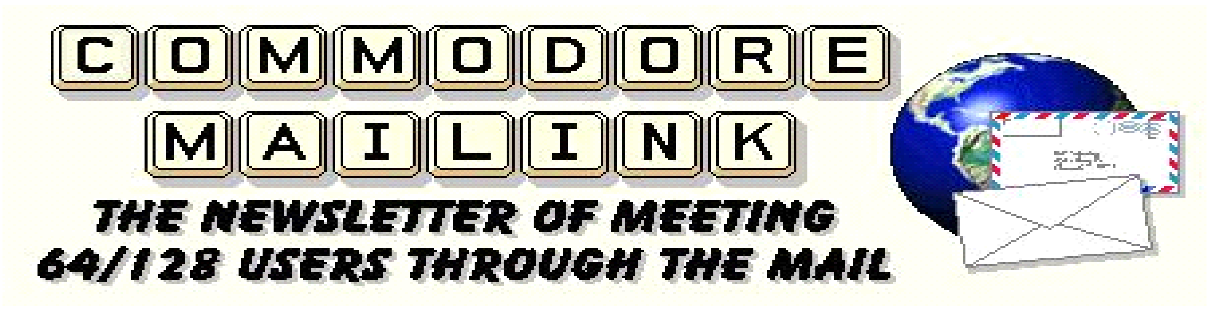
CF. I did think long and hard about clubs like MUTTM and what they try to do I felt that you were unique because of the members BIO and I felt creating Commodore Free would not reduce your members list, also Commodore Free needs to be downloaded and manually printed to be read, whereas MUTTM is delivered pre-printed to the users home address, I welcome comments and feedback from yourself. What is the best piece of hardware you have (peripheral) for your Computer (it doesn't have to be a Commodore machine) and why

MUTTM
My best piece of commodore hardware, apart from the 64 machine itself, is the CMD hard drive. The speed of loading and convenience of storage of so many programs has spoiled me. For two years now I have been without it and I have been relearning the realities of 1541 disk drive space and wear. I still have hope to resurrect the hard drive but it has taught me that someday I may be without some piece of equipment. I need to repair some equipment, especially drives, soon or else there may be nobody who can fix them for me

CF.
Finally do you have any other comments you would like to add?

MUTTM
Thank you for the opportunity to write to you, and your readers about myself and Meeting 64/128 Users Through the Mail—MUTTM

The History of Meeting Users through The Mail



This Is Who We Are:

In the May 1986 issue of Compute!'s Gazette, the User Group listing for the first time included The 64 User Group of America, with Kirby Herazy listed as the founder and president. The members of the group (all 35 of them!) corresponded with each other through the mail, but there were no regular group mailings, no newsletter and no dues.

In 1987, the group's name was changed to Meeting 64/128 Users Through the Mail and Jean Nance became the president. A membership list was started, containing a brief biography of each member, along with their Commodore system components and their computing interests. Later that year, a bi-monthly newsletter was started with the simple, if bland, name of Newsletter, with Jean Nance as the editor.

Most of the articles were (and still are) written by the group members themselves, with occasional items of special interest reprinted from outside sources. There were still no dues required for membership in the group, so donations from the members covered the cost of printing and mailing Newsletter. After three issues were published, a contest was held to give the newsletter a more interesting name, and from the suggestions submitted by the group The Commodore MaiLink was the standout winner.

In 1988 with membership numbers and publishing costs climbing, annual dues of $5.00 were instituted. Since Commodore users being such a, umm, thrifty bunch, the membership numbers dropped precipitously. Because good word-of-mouth and a steady place in the Commodore user group listings in the popular Commodore magazines, the group membership soon began climbimg steadily.

It reached a peak of more than 300 in the late 80s and early 90s. And as the membership grew, so did the membership bio list. Meeting 64/128 Users Through the Mail has always been a non-profit organization and it relies solely on membership dues and the occasional member donation to continue publication.

In 1994, the combined office of Group President/MaiLink Editor was (deservedly) split into two distinct offices. Frank Redmond became the new president of Meeting 64/128 Users Through the Mail, with Jean Nance keeping The Commodore MaiLink editor-in-chief position.

In 1996, Frank stepped down and Tom Adams stepped up to become the new group president, with Frank assuming the vice-president's position. Long-time members Brian Vaughan and Rolf Miller maintained the membership records and the group treasury, respectively.

In late 2001, Linda Tanner has stepped into the role of President, with Tom Adams and Frank Redmond as Vice President. Emil Volcheck is treasurer, David Mohr is the Managing Editor, with Brian Vaughn as the Bio-Editor. Our Email address editor is Joseph Fenn and Linda also handles the Member Resources list.

Today, Meeting 64/128 Users Through the Mail has members in the United States, Canada, Greece, France and Australia. Some of our members are or have been contributors to such publications as RUN, Compute's Gazette, Loadstar, Loadstar Letter and Commodore World; others are active contributors to the COMP.SYS.CBM and ALT.C64 newsgroups.

The Commodore MaiLink newsletter is 18-20 pages for each issue and is published regularly six times a year (the odd-numbered months). Each issue is edited by a volunteer from the group; the articles, still with only a few exceptions, are written and submitted for publication by members of the group. The membership list is included twice yearly (in the March and September issues) and includes a brief biography, the address, occupation, hobbies, the Commodore or Commodore-compatible components owned, and the special computing interests of each member.

It also includes a contact list of group members with special expertise in software packages, programming and hardware hacking and repair, along with a Commodore vendor list of those sellers that offer discounts to Meeting 64/128 Users Through the Mail members.

All of our members love Commodore computing. If you use a Commodore computer, and if any of this has piqued your interest, then why not join the club?

Meeting 64/128 Users Through the Mail Membership Application Information

Meeting 64/128 Users Through the Mail is a non-profit Commodore user group. The Commodore MaiLink is not a subscription newsletter--you can't just drop a check in the mail and expect a newsletter in return; you must fill out a membership application first. This tells the other members a little about yourself--your hobbies, your computing interests and the Commodore hardware you have. All of the yearly membership dues and any donations from members go toward publishing and mailing The Commodore MaiLink. The current dues are as follows:

$15.00 per year for U.S. residents
$17.00 (U.S. funds) per year for residents of Canada and Mexico
$25.00 (U.S. funds) per year for elsewhere in the world

Your check or money order must be made payable to Robert Snyder, Treasurer, since he's the club treasurer and he handles all of the the accounting headaches. Checks made out to anyone but Robert will be returned. Send the completed membership application to Robert Snyder:

Robert Snyder, P.O. Box 64, Metamora OH 43540-0064 USA.
Robert sends the Bio information to our Bio Editor, who updates the membership database.

MEETING 64/128 USERS THROUGH THE MAIL
The information will be used to write your "bio" for the membership list. Send it to me at the address below, along with a check or money order for $15.00 in U.S. funds for those in the United States, $17.00US for those in Canada or Mexico, $25.00US for all other countries made out to "Robert Snyder", our treasurer. You will receive all the material for the current year.

If you have any questions, write, phone or Email me (president@mailink.videocam.net.au). I hope you will decide to join us.

Date ________________

Name ____

Address ____

City, State/Province, Zip/Postal code and country ______________________________
____

Telephone number and / or FAX (optional) ______________________________

Sex Male ____ Female ____

E-Mail address ____

Occupation ____

Hobbies or interests in addition to computing ______________________________
____
____

Computers used (Please designate model ie: 64 or 128) ______________________________
____

Drives ____
____

Printers ____
____

Monitors ____

Other ____
____

Non Commodore Equipment ____
____

Special computer interests ____
____

The Bio information may be updated at any time but will not be changed until the next published membership list.

Robert Snyder, P.O. Box 64, Metamora OH 43540-0064 USA

Interview with Pasi Ojala Vic20 Coder and Demo Creator

http://www.cs.tut.fi/~albert/Pu-239/vicual-mmix/

Q. Please introduce yourself to our readers

A. I'm a 39-year-old embedded software engineer that does not shy away from hardware either. I work in VLSI Solution Oy (http://www.vlsi.fi/)developing code among other things for audio decoder chips and various applications using them.

Q. How did you become interested in the VIC-20?

A. A friend gave me a VIC20 around 1989, and I played with it a bit, converting some C64 fonts to VIC's 16-line characters. Nothing really became of it though, because I didn't have time or great motivation to really study the operation of the chips inside the VIC20. My interest in VIC20 sparked in 1995 when Marko Mäkelä and Andreas Boose set out to find out the ins and outs of the VIC-I video chip. I had run out of demo ideas for C64 and was no graphician or musician so it was natural to take a step back to VIC20, the undiscovered demo territory.

Q. Was the Vic your first home computer?

The first computer I played with was a ZX-81 owned by a friend of mine. A few years later another friend had a VIC-20 and we created a few basic programs with it. My first computer was Sinclair Spectrum 48k. I created a few BASIC games with it, using user-definable characters and used a lot of string operations to make the games fast enough. At some point I broke it by fiddling with the expansion port. The machine visited a repair shop but never worked very well after that.

After a few years I bought a C64 and a 1541 drive. I also bought a Handic 300bps / 1200/75bps modem and called the few bulletin board systems available in Finland at that time. The phone bills made a stop to that for a long time until the first BBS's appeared in the Tampere area code.

This encouraged me to start my own BBS Pasbox (imaginative, right?), which ran my own software on a C64 and a 1581 disk drive. Machine language subroutines that I wrote made the BBS faster and better.

One of the first and nicest routines was a TOD clock with calendar that was automatically updated by the TOD alarm interrupt at midnight. Leap years were of course supported. You could get the time and date in human-readable form into a string variable by giving the right SYS command. You got for example "21:31.35 Fri 5.1.90".

Another nice routine was a line wrap and hyphenation routine for finnish.

Fittingly to the theme, another BBS in Tampere was run on a VIC20, also with self-made software. We even created a message transfer protocol that swapped messages between our systems. The system was called Pet-Net and a few other systems also joined that.

On the hardware side I created a 256kB RAMDISK that stored the recently read messages so the 1581 drive would not need to spin up so often. The RAMDISK hooked into the load and save vectors so it could be easily used from BASIC programs.

In the later years the C64 was retired from Pasbox service and A500 took the role using a BBS software named AXsh that looked like a Unix shell, but with protections so that you could not run any non-trusted programs. I even piped usenet newsgroup messages from my university account to the BBS.

Q. Have you programmed for any other Commodore computers (Amiga included)? If so, how does programming the VIC compare? i.e., does a lack of hardware sprites help or hinder

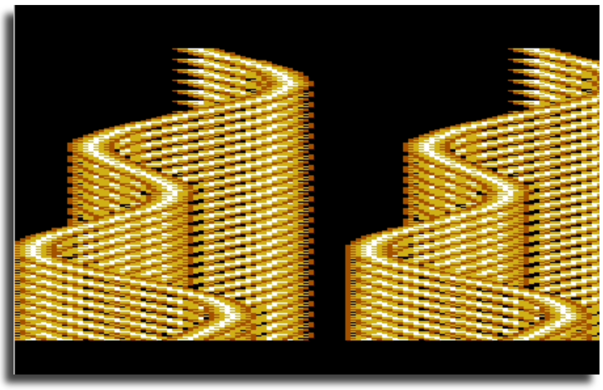
A. I never did any demo coding on the Amiga, just "productivity" like AXsh, stroke commodity, c1581-handler.

VIC20 demo coding compared to coding on C64 is at its core the same thing: to use the capabilities at your disposal in the best possible way. When working with Vic you are just much more limited in every way, which emphasizes good ideas. The lack of sprites forces you to select ideas that the machine's good at instead of trying to make it do something it does poorly.

Q. What is your favourite 8-bit demo (excluding your own works)?

A. I'm an old-school dude and I appreciate technical over pretty. I really liked the Think Twice! and similar demos that showed some VIC-II trick for the first time or made good use of it. There are tons of demos that are spectacular in graphics and sound, and I'm quite amazed by those as well.

Q. Can you tell our readers about VICUAL MMIX



A. VICUAL MMIX is a nine-part demo for the unexpanded VIC20. New parts are loaded from disk as the demo progresses The demo tries to show visually interesting effects that have not been seen on VIC20 before. Some of the parts are further developed versions or different takes on parts seen in my previous VIC20 demos.

Q. Is there anything specific behind "this kind of demo" and "such effects"?

This time I toned down the "use less known VIC-I features" and "this is impossible in 5.5kB" and went more in the direction of keep it simple, use interesting visual effects and unusual colour combinations. I also made the parts roll on much quicker so that you would not get bored waiting for the next part. If you want to watch for longer and see more pretty colours, you can run the demo in manual mode.

Q. This Demo runs on an Unexpanded VIC20! What was the main problem coding the demo apart from lack of memory?

A. Is that not enough?:-) Well, as I said before, I'm no musician, so help from Anders Carlsson was needed to fill the silence with something you would listen to willfully.

Q. Why create such a Demo on the Vic? is it purely because you wanted to show the machine was capable of such techniques?

A. It's a hobby, coding is fun. And demo coding on VIC20 is so much different from what programming I do every day that it is quite relaxing as well.

Q. Can you tell our readers about your programming environment and the tools used to create the demo?

A. When I started developing the demo in 2002-ish, I used A3000 with Cygnus Editor, dasm, SAS/C, DPaint, and my c1581-handler to read/write c1581 disks directly on the Amiga. Some small utilities were written for previous VIC20 demos and this one. In 2003 I started using a FreeBSD PC, so the development environment changed a bit. I still use dasm, but now with emacs, gcc, and gimp. I also made use of Marko Mäkelä's c2nload that connects between the PC's serial port and VIC20's tape port.

Q. And is the source code freely distributable?

A. The demo and source code is on the web freely available and you can give it to other people. It is intended for people to learn from.

Q. Some readers wonder why programmers spend such time working on demos why didn't you create a game or utility for the Vic?

A. I have thought about making a game for VIC20, but it should not be just any game. Some originality should be involved, or at least something not thought possible with a VIC20. There are a few utilities of mine for VIC20. Although not running on a VIC, pucrunch supports VIC20 and is used in my VIC20 demos.

(That and the small memory keep the loading times short.) PuZip has a VIC20 version, so you can make ZIP archives on your lovely machine, although you need a bit of expansion memory. I have also released a graphics converter that creates graphics files run able on a vic.

Q. I personally thought I had "seen it all" I mean the vic is so limited, although some feel the limitations are an aid to creativity, Do you think there is anything left to achieve on the VIC-20 now?

A. VICUAL MMIX did not push the VIC20 much further than before, so certainly this is not the end.

Q. Would you consider doing a multi-part 'one-filer' which requires a RAM expansion? Or any programming that uses extra RAM? or do you prefer to utilise the hardware "as is"

A. My first VIC20 demo "Vici Iterum MM" required 16kB expansion RAM. Some of the parts were original, some just remakes of old but popular C64 effects. The extra RAM was mainly used for loading the next part while the current part was running.

There are three reasons why I switched to using the unexpanded machine.

1) Requiring memory expansion limits the audience. Although VICE currently emulates PAL VIC20 very accurately, and NTSC VIC20 fairly accurately, there are still people that want to run the demos on the actual machine for one reason or another and they don't all have expansion memory cartridges.

2) Expansion memory can not be used for graphics, because VIC-I can only see the internal RAM. This limits the usefulness of the extra RAM.

3) All cool dudes write stuff for the unexpanded machine :-) Making best use of the limited memory resources is part of the fun.

Q. What would you rate as your best achievement in programming terms?

A. In the 8-bit-world gunzip (a GZip and ZIP decompressor) surely loses to puzip.

Q. can you tell our readers about any other projects you are working on, and also maybe some of the past works you have created, also are these available to download?

A. I already mentioned some. Most if not all of my projects are available from my web pages. You can start browsing from http://www.iki.fi/a1bert/Dev/ .

Q. If you could change 3 things on the vic20 what would you change?

A. I'm only considering things that were probably realistic back them. Machine-level: 8kB of internal RAM instead of 5kB. VIC-I video/sound chip:

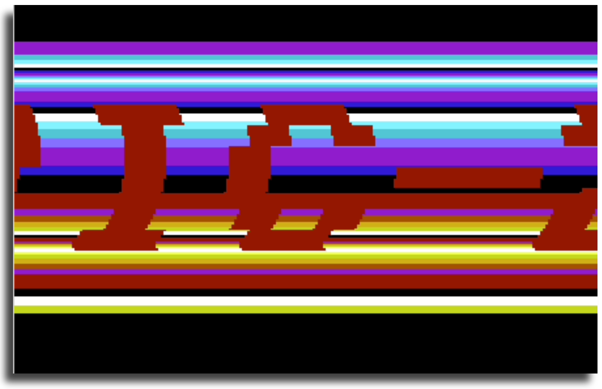
1) Horizontal smooth scroll -- now horizontal positioning resolution is one cycle, which is 4 pixels

2) 8 bits of frequency control for sound -- now 1 bit is used for on/off and 7 bits for frequency

3) global multicolour mode bit, so that in hires-only mode you could use all 16 colours as character colours

Q. Do you program for any other systems, 8 bit or otherwise?

A. I started my 6502 assembly programming on the C64, but it seems VIC20 is enough right now. At work I'm programming our digital signal processor VSDSP in assembly and in C. After 11 years of assembly coding I'm slowly becoming good at it. :-)



Alternative Programming Languages: C Part 2

By Paul Davis

Welcome back to this brief look at cross-development with the C programming language. I hope you are finding it interesting and are inspired to learn more about this useful language. As always, I welcome your feedback and have now set up a blog where you can leave comments on any of my articles. The links for my web site and blog can be found at the end of this article.

Last time, I set a small challenge to write a function that moves a sprite to any position on the screen. How did you do? Here is one possible solution to that problem:

```
void move_sprite(int x, int y)
{
 VIC.spr0_y = y;
 if (x < 256)
 {
  VIC.spr_hi_x = 0;
  VIC.spr0_x = x;
 }
 else
 {
  VIC.spr_hi_x = 1;
  VIC.spr0_x = x - 256;
 }
}
```

Bonus points are awarded if you came up with something along these lines:

```
void move_sprite(int x, int y)
{
 VIC.spr0_y = y;
 VIC.spr0_x = x & 255;
 VIC.spr_hi_x = x / 256;
}
```

Back to this issue and, as promised, this time we are going to be creating our very own sprite handling library. In the process we will learn more about how to build a C program from several smaller components and how these separate parts are compiled and linked together. We will also introduce some other tools to make the development process a little easier.

Getting started

We will be using a command line environment to compile our programs, so the first thing to do is start the command prompt/terminal window application for your particular operating system. Next, we need to make sure the environment is properly configured. If you are running Windows enter these commands:

```
 cd c:\cc65\tut
 init
```

The 'init.bat' script is available in the zip file attachment for this article on my web site. If you previously set up the environment permanently in System Properties you do not need to run this script.

Linux and OSX users enter these commands:

```
 cd ~/Documents/cc65
 . init.sh
```

Creating a header file

Before we jump in and create our function library, let's take a step back and see how to create and use a header file. If you remember from last time, header files are used to declare the existence of various functions, data types and constants in the C libraries so we can use them in our programs. There's nothing particularly special about a header file, it's just another plain text file and we can easily create our own.

Many of the programs we have seen so far contain similar code at the start. For example we have used the 'typedef' instruction to create a type such as 'byte' to represent an 'unsigned char'. Rather than keep repeating this code over and over in every program we write, we can put the code into a header file and just include that file in our programs. Let's try it. Edit a new file called 'mytypes.h' (OSX users replace 'notepad' in the command below with 'edit', Linux users replace it with your favourite text editor):

```
 notepad mytypes.h
```

Now enter (or copy and paste) the following code into the file and save it:

```
typedef unsigned char byte;
typedef unsigned int word;
typedef unsigned char bool;

#ifndef TRUE
#define TRUE 1
#define FALSE 0
#endif
```

As is becoming customary, I've sneaked in some new language features here. The first couple of typedef lines should be familiar; they create the new types for unsigned 8 and 16 bit values, byte and word respectively. The third typedef is for a Boolean (true or false) value. The standard C language doesn't have a dedicated Boolean type, it simply treats any zero value as 'false' and any non-zero value as 'true'. It's often helpful to create such a type to make the intent of our programs clearer.

For example, if a function takes a 'bool' as a parameter, it's more obvious that you should be passing in a true or false value. If that same function just declared its parameter as a 'byte' it's not so obvious. As you may have guessed, we will be making use of this 'bool' type in our sprite library.

The next group of lines create constants for our new bool type, TRUE and FALSE. But what's the deal with the #ifndef part? This instruction is one of several that C provides for 'conditional compilation'. The instructions between that line and the #endif line will only be compiled if a certain condition is met. The '#ifndef' instruction stands for 'if not defined'. In other words the whole line means 'compile the following lines only if TRUE has not already been defined'.

Why would it already be defined, you ask? It turns out that making a 'bool' type and the TRUE/FALSE constants is a fairly common practice. It's possible that other header files from other libraries will have such definitions. Since C doesn't allow constants to be re-defined, we use #ifndef as a precaution so that our header file doesn't try to define the values again if they already exist.

To test our header file we need to create a program that uses it. Edit a new file called 'bool.c' and enter this program into it:

```
#include <stdio.h>
#include "mytypes.h"

void main(void)
{
  byte x = 123;
  word y = 12345;
  bool z = TRUE;

  printf("x = %d\n", x);
  printf("y = %d\n", y);
  printf("z = %d\n", z);
  printf("z is %s\n", z ? "true":"false");
}
```

Compile the program using the command:

```
  cl65 bool.c
```

Running the program in the VICE emulator will print out the numeric values of the variables along with the true/false value of the 'z' variable in words.

This program includes our header file to allow it to use the new types. Notice how, in the #include line, the file name is contained in double quotes rather than 'angled brackets'. This is important. The brackets tell C to search for the file in the compiler's 'include' directory. Double quotes tell C to search for the file in the current working directory.

Another new language feature has crept in here. On the final printf line, we use what is called the 'tertiary if operator' represented by the question mark. This is a very useful mechanism in C for performing an 'if' without the full-blown syntax of the 'if' statement. It has this form:

```
  condition ? true_value : false_value
```

The part before the question mark is a condition, something that has a true or false result. In this example, the variable 'z' has such a value already so the variable itself is the condition. After the question mark is the value to use if the condition was true, in this case we use the literal string "true". The value to use when the condition is false is separated from the true value by a colon. Here we just use the literal string "false". The end result of all this is that when the printf function is called, the value of the z variable is tested and if its value is true, the string "true" is passed as a parameter, otherwise the string "false" is passed instead. This is a much neater solution than the alternatives using a standard 'if' statement, such as:

```
  if (z) printf("z is true\n");
  else printf("z is false\n");
```

or:

```
  char *zstr;
  if (z) zstr = "true";
  else zstr = "false";
  printf("z is %s\n", zstr);
```

Learning how to write and use a header file is the first step towards creating our function library. Next, we need a little more understanding of the how the compiler translates our C source code into an executable.

The compilation process

Up until now, we've been using the 'cl65' command to compile our programs. This is actually a convenience command provided by the cc65 authors to make compiling simple programs as easy as possible. But now it's time to learn the individual steps in the compilation process.

To start with, let's create an 'empty' program. Create a new file called 'sprtst.c' and enter the following code into it:

```
void main(void)
{
}
```

Remember from the first article that a C program must have a main function, so this represents the simplest possible 'program' for testing purposes, one that does absolutely nothing.

Now, we're not going to compile this using the familiar cl65 command. Instead we're going to go through the same steps by hand that cl65 would automatically do for us. First, we need to compile the program source code into assembly language using the cc65 command as follows:

```
  cc65 -t c64 sprtst.c
```

One thing to note here is that we now have to specify the target machine using the '-t c64' option. The cc65 compiler works with many 6502 based systems and needs to be told which one we want to use. The cl65 command automatically assumes c64 by default, but the individual compilation commands do not.

Once the program has compiled, Windows users enter this command:

```
  dir sprtst.*
```

OSX and Linux users enter this command:

```
  ls -l sprtst*
```

This command lists all files called 'sprtst' with different extensions. You should see our 'sprtst.c' source file and a new file created by the cc65 command called 'sprtst.s'. This is the result of the C code being translated to assembly language.

The next step is to assemble this file into machine code using the ca65 assembler. Enter the following command:

```
  ca65 -t c64 sprtst.s
```

Enter the dir (or ls) command from before and you will see a new file 'sprtst.o' in the list. Here's a tip for anyone who is not familiar with using a command line environment. You can repeat previously typed commands by pressing the cursor up key until the command you want to run is shown then pressing the return key to run it.

The output from the assembler is not a complete program that you can run on the Commodore. Files with a '.o' extension are called 'object files' and they contain, among other things, machine code fragments. Why create this intermediate file? Well, it allows a C program to be built up from multiple components. You have already seen that many C functions are stored in libraries. These libraries need to be linked together with our own code to create the complete executable program. The tool used to perform this linking is, unsurprisingly, called a linker and is run as follows (enter this all on one line):

```
 ld65 -t c64 -o sprtst c64.o sprtst.o c64.lib
```

This command is more complex than the previous ones and requires some explanation. The '-t' option as usual indicates we are creating a program for the c64. The '-o' option tells the linker the name of the output file, that is the filename of the finished executable program. Here we are calling it 'sprtst'. Following these options are the names of all the files that are linked together to create our program. You will recognise 'sprtst.o' as our program, but what are the other files?

As mentioned previously, every program starts at the 'main' function. This doesn't happen by magic, there is actually a small piece of 'startup' code that is placed at the beginning of every program that performs some initialisation such as clearing memory and then calls the 'main' function. This 'startup code' is contained within the file called 'c64.o' and must be the first file listed in the link process. The last file, c64.lib contains those standard library functions we have been using all this time. The code for any functions called by our program will be copied from this file and linked into our program.

Run the dir (or ls) command again and you should now see 4 files, 'sprtst.c', 'sprtst.s', 'sprtst.o' and finally the executable file 'sprtst'. You could run this program if you like but, as you already know, it doesn't do anything… yet!

Okay, I can imagine you're thinking that's a lot of effort to compile a program that does nothing! Indeed it is, and the cl65 command was provided to remove the need for this effort. But, bear in mind the goal of this article is to create a function library. As you may have already surmised from what we have just done, to create a library we need to be able to create 'object files', and to do that we need to use the separate compilation commands.

No doubt the thought of repeatedly typing all those separate commands isn't appealing. Fear not, we are now going to take a look at a tool that can alleviate some of the pain. It's called 'make'.

Makefiles

Before I can show you how 'make' works it needs to be installed on your computer. OSX and Linux users are lucky enough to already have it but Windows users will need to install a copy. Since the other operating systems use the gnu implementation of 'make' we will, for consistency, also use this on Windows.

The gnu tools are hosted on the sourceforge web site but I'm going to take the opportunity to introduce another useful package called 'unixutils'. This package contains many gnu unix tools that have been compiled natively for Windows, that is they don't require any other DLLs or 'unix emulation libraries' to be installed in order to use them. They are older versions of the tools but work well and are easy to install. If you are more familiar with the gnu tools and unix libraries for Windows, feel free to install whatever version of 'make' you like.

Unixutils can be found at http://sourceforge.net/projects/unxutils. Click on the big green download link, then on the download link of the latest entry in the list of files that appears. Finally, click on UnxUtils.zip to download the file.

Next, unzip this file to a temporary directory and navigate to the usr\local\wbin directory. In here locate the 'make.exe' file and copy it to your c:\cc65\bin directory. This isn't particularly good practice, but it's the simplest way of getting up and running since our environment is already set up to look for commands in the cc65\bin directory. If you are more experienced with such things, you can install the entire utils package to a directory of your choice and add the 'bin' directory to the 'Path' environment variable.

Now, to test that we can run the command, enter this line:

```
 make
```

You should get an error message from 'make' saying it can't find a target or a makefile. If you get a message saying that 'make' is not recognised as a command, ensure that you have placed the make.exe file in the c:\cc65\bin directory and try again.

Right, OSX and Linux users can join us again now. So how can 'make' help us to build our programs? The best way to explain is by example. We need to create a new file called 'makefile' (with no extension). OSX and Linux users can just create the file in the usual way, but if you are using Notepad on Windows there is a problem. If you entered the command 'notepad makefile' it would actually create a new file called 'makefile.txt'. To get around this, enter the command with a trailing 'dot':

```
 notepad makefile.
```

Now enter the following lines into the makefile. Be very careful here, the lines containing the commands (ld65, ca65, cc65) must be indented using the tab key, not spaces. Unfortunately, this means cutting and pasting this text in its entirety will not work. You can copy a line at a time, just remember to press tab before each command.

```
sprtst: sprtst.o
	ld65 -t c64 -o sprtst c64.o sprtst.o c64.lib

sprtst.o: sprtst.s
	ca65 -t c64 sprtst.s

sprtst.s: sprtst.c
	cc65 -t c64 sprtst.c
```

You will recognise these commands as the ones we entered a short while ago. So let's explain how this file works.

An entry in a makefile is formatted like this:

```
  target-file: dependent-files
      commands to build target-file
```

A line that starts with a filename followed by a colon indicates a 'target' file, that is, a file you want to create from some other files. Those other files are listed after the colon and are known as the dependents of the target file. Following this line is one or more command lines, each indented by a tab character. When run, these commands will create the target file from its dependents.

Looking at the makefile we have just created, we can see that the 'sprtst' file is dependent on the 'sprtst.o' object file and uses the ld65 command to build the program from this file. The 'sprtst.o' file is, in turn, dependent on the 'sprtst.s' file and is created from it with the ca65 command. Finally, the 'sprtst.s' file is dependent on the 'sprtst.c' file and is created from it using the cc65 command.

As you can see, we have built a 'chain' of dependencies and corresponding actions. This is extremely useful because it allows 'make' to work out what commands need to be run whenever changes have been made to a particular file. Edit the 'sprtst.c' file and add an include line to the start of it:

```
#include "mytypes.h"

void main(void)
{
}
```

This program still does nothing. I just want to demonstrate what happens when a change is made to a file. Save the file and enter the make command:

```
  make
```

The make command will run through all the dependencies and determine that the '.c' source file has changed so it needs to run the cc65 command to compile it. Then, since the '.s' assembler source has now changed, it needs to run the ca65 command to assemble it. Then, since the '.o' object file has now changed it needs to run the ld65 linker to create the executable. The makefile has helped us to remove some of the complexity of using the separate compiler tools. Now, whenever you make a change to your program, just issue the 'make' command and it is automatically compiled and linked.

Makefiles have many other features that can simplify the compilation process even further. These features are beyond the scope of this article. I will leave it as an exercise for the reader to investigate this in more detail.

Creating the function library

Finally, we're armed with the knowledge and tools required to create our sprite handling function library. So let's get started. First, we should start from the point of having a working program that we will gradually change to use our library functions. Edit the 'sprtst.c' file and change it to the following:

```
#include <string.h>
#include <peekpoke.h>
#include <c64.h>
#include "mytypes.h"

#define SPRITE0_DATA 0x0340
#define SPRITE0_PTR 0x07F8

void main(void)
{
  memset((void*)SPRITE0_DATA, 0xff, 64);
  POKE(SPRITE0_PTR, SPRITE0_DATA / 64);
  VIC.spr0_color = COLOR_WHITE;
  VIC.spr_hi_x = 0;
  VIC.spr0_x = 100;
  VIC.spr0_y = 100;
  VIC.spr_ena = 1;
}
```

Compile it (using the 'make' command) and run the 'sprtst' program in VICE to check that it works.

Now we have a working program, we can write and test functions in our library one at a time by replacing the lines in this program with calls to the new function. This incremental approach: write a bit, test a bit, is a good way to build a library, especially when you're new to this kind of development.

It's always a good idea to start with a little planning. We need to decide what features we want the library to provide for us. It's easy to get carried away and put anything and everything in there. Some thought at the beginning can help to focus our efforts.

For our example library we will keep things simple. We want to be able to show and hide sprites, change their colour and move them anywhere on the screen. We also want to be able to use all 8 sprites, not just one. This last requirement gives us a clue that many of the library functions will need to take a sprite number as a parameter. Okay, let's start with one of the easier functions, the one that enables a particular sprite and shows it on the screen. In our test program above we simply put the number 1 into VIC.spr_ena, but our requirement to use all the sprites means we need to do more than this.

The 'sprite enable' register of the VIC chip is actually a bit-mapping. In other words, each bit of the number stored there corresponds to the setting for one sprite. Since there are 8 sprites and 8 bits in a byte, this fits nicely. The question is, how do we get to a particular bit?

The C language has a convenient way of doing this using the 'shift left' operator. The expression '1 << n' where 'n' is the sprite number will generate the correct value. It's beyond the scope of this article to explain in detail why this is, so for now, just take it on faith that it works.

With this knowledge, we can now make an attempt at a function to enable any sprite. Create a new file called 'sprlib.c' and enter this code into it:

```
#include <c64.h>
```

```
#include "sprlib.h"

void spr_show(byte spr)
{
  VIC.spr_ena |= 1 << spr;
}
```

We have called the function spr_show. It's usually a good idea to prefix the names of functions in a library with the same few characters to identify which library they belong to and avoid name collisions with functions in other libraries. Here we are using 'spr' for our sprite library.

The function takes a sprite number as a parameter and uses the expression '1 << spr' to calculate the bit value for that sprite. It then uses the '|=' operator to combine this bit value with the value that is already in the register. If we just assigned the value with '=' it would turn off all the other sprites except the one we want to show. The '|' symbol is the 'bitwise OR' operator in C.

Next, we need to provide a way for other programs to call our function. We do this by creating a header file for those programs to include. Create a new file called 'sprlib.h' and enter these lines:

```
#include <c64.h>
#include "mytypes.h"

void spr_show(byte spr);
```

The declaration of the spr_show function here is called a 'function prototype'. It just lists the return type, name and parameters of the function. Notice that the prototype doesn't have any code after it in curly braces and that the line ends with a semi-colon. Prototypes are C's way of telling other programs that a function exists and what its return type and parameters are.

Next, in the 'sprtst.c' file, change the #include "mytypes.h" line to read:

```
#include "sprlib.h"
```

and change the line that reads:

```
  VIC.spr_ena = 1;
```

to use our new function instead:

```
  spr_show(0);
```

Now we need to change our 'makefile' to add the instructions to compile the library. Change 'makefile' to look like the text below. Remember to indent the commands with a tab, not spaces (the ld65 command should be entered all on one line, even though it has wrapped around in the magazine listing):

```
sprtst: sprtst.o spr.lib
    ld65 -t c64 -o sprtst c64.o sprtst.o spr.lib c64.lib

sprtst.o: sprtst.s
    ca65 -t c64 sprtst.s

sprtst.s: sprtst.c sprlib.h
    cc65 -t c64 sprtst.c

spr.lib: sprlib.o
    ar65 a spr.lib sprlib.o

sprlib.o: sprlib.s
    ca65 -t c64 sprlib.s

sprlib.s: sprlib.c sprlib.h mytypes.h
    cc65 -t c64 sprlib.c
```

This new makefile contains instructions to compile the sprlib.c file and contains a new command 'ar65' that puts object files into a library file. Our sprite library is called 'spr.lib'. The entry for 'sprtst' has also been changed to add spr.lib as a dependency and as part of the 'ld65' link instruction.

Compile these changes by running the 'make' command. If all goes well, it should compile sprlib, create the library and compile sprtst, linking with the new library. Load and run this program in VICE to check it still works.

Next, we will create the function to change the sprite colour. This will need to take two parameters, the sprite number and the colour value. Add the following lines to the end of 'sprlib.c':

```
void spr_color(byte spr, byte color)
{
  (&VIC.spr0_color)[spr] = color;
}
```

The use of the ampersand in this function needs some explanation. The VIC chip registers for sprite colours are conveniently located together in memory. The colour for sprite 1 immediately follows that for sprite 0. We can take advantage of this arrangement and treat those registers like an array of bytes. The only problem is the variable VIC.spr0_color is of type 'byte' so we can't treat it directly as an array.

In C, an 'array' type is nothing more than a pointer to the memory address of the first element. It follows therefore, that if we can get the memory address of the VIC.spr0_color variable then we can treat it as an array. This is exactly what the ampersand operator is used for here. An ampersand returns the memory address of the variable name that follows it. Once we have that address, we can simply use the familiar square brackets as an index into this 'array'.

As before, we need to add a function prototype to the end of 'sprlib.h':

```
void spr_color(byte spr, byte color);
```

In the test program, 'sprtst.c', edit the line that changes the sprite colour to use the new function:

```
spr_color(0, COLOR_WHITE);
```

Compile all the changes by issuing the 'make' command and run the program in VICE to test it still works.

Let's keep up the momentum and create a function to set a sprite pointer. Enter this code into 'sprlib.c':

```
byte *sprite_pointers = (byte*)0x07F8;

void spr_ptr(byte spr, byte ptr)
{
  sprite_pointers[spr] = ptr;
}
```

For the moment, we have just hard-coded the start address of the sprite pointers in memory. It is possible, and preferable, to use the VIC registers to calculate what this address should be. We will see how to do this later.

Add this prototype to the 'sprlib.h' file:

```
void spr_ptr(byte spr, byte ptr);
```

Finally, change the line in 'sprtst.c' that pokes the sprite pointer to use the new function:

```
spr_ptr(0, SPRITE0_DATA / 64);
```

Compile with 'make' and test in VICE as usual to check it still works. By now you should be starting to appreciate the benefit of the make tool. The effort put in setting it up at the start is now being paid back many times over. Making a change to the library is very simple and we have got into a good rhythm of editing the source files, compiling and testing.

The function to set the sprite pointer is okay, but it still makes the caller calculate the pointer value from the real address. We could change the function, or write a new one, that takes an address rather than a pointer but there are times when pointer numbers are the better fit, animation being the prime example. Our library would be more useful if it provided a way to calculate the pointer number from an address. We could write a function to do this but C provides another way, macros. Enter this line into the 'sprlib.h' header file (all on one line):

```
#define SPR_PTR(addr) (((addr) / 64) & 0xff)
```

In the 'sprtst.c' file change the line that sets the pointer to this:

```
spr_ptr(0, SPR_PTR(SPRITE0_DATA));
```

We have seen the #define instruction used to create constants before. Here it is being used to create a macro. To the program using it, a macro looks just like a function. But unlike a function, it is not called. Instead, the macro is literally replaced by its text value. To demonstrate, let's see what the compiler would do with the spr_ptr call above.

The compiler recognises SPR_PTR(SPRITE0_DATA) as a macro and expands it using the text in the original definition. The resulting instruction would look like this:

```
spr_ptr(0, (((SPRITE0_DATA) / 64) & 0xff));
```

The value SPRITE0_DATA is also a defined constant so this too will be replaced with its literal value. The command now looks like this:

```
spr_ptr(0, (((0x0340) / 64) & 0xff));
```

The compiler is clever enough to realise that because this expression uses literal values it can simplify it by calculating the result. The command now looks like this:

```
spr_ptr(0, 13);
```

This is the major benefit of using macros instead of functions for some tasks. They can help to optimise the code, especially when used with constant expressions and values.

The last function that we need to write to finish our test program moves a sprite to any position on the screen. Enter this function code into 'sprlib.c':

```
void spr_move(byte spr, int x, int y)
{
  byte index = 2 * spr;
  (&VIC.spr0_x)[index] = x & 0xff;
  (&VIC.spr0_y)[index] = y;
  if (x & 0x100) VIC.spr_hi_x |= 1 << spr;
  else VIC.spr_hi_x &= ~(1 << spr);
}
```

This new function is a bit more complicated than those we have previously written, but it uses several of the same ideas. Part of this complexity is caused by the layout of the sprite position registers in memory. Unlike the colour registers which are conveniently grouped together, the position registers are grouped in pairs: sprite0_x, sprite0_y, sprite1_x, sprite1_y and so on. This means that we can't use the sprite number directly as an index to an 'array' of registers. The function needs to multiply the sprite number by 2 to get the array index because there are groups of two registers per sprite.

The spr_move function has also introduced a couple of new operators. The tilde character '~' is the bitwise NOT operator in C and the ampersand is the bitwise AND operator. Together they are used to turn off a particular bit in a value. At some point in the future I'm hoping to write an article that describes in more detail how these 'bit manipulations' work. For the purposes of this article however, suffice it to say that to turn a bit on, use an expression like this:

```
  variable |= 1 << bit
```

And to turn a bit off, use an expression like this:

```
  variable &= ~(1 << bit)
```

We now need to add the prototype for this function into 'sprlib.h':

```
void spr_move(byte spr, int x, int y);
```

And replace these lines in 'sprtst.c':

```
  VIC.spr_hi_x = 0;
  VIC.spr0_x = 100;
  VIC.spr0_y = 100;
```

with a call to the new function:

```
spr_move(0, 100, 100);
```

Compile these changes with 'make' and test the program in VICE as usual.

At this point we should take the opportunity to tidy up the test program. There are a couple of include files and the definition of the sprite pointer address that are no longer needed. Remove these lines from 'sprtst.c':

```
#include <peekpoke.h>
#include <c64.h>

#define SPRITE0_PTR 0x07F8
```

Compile again with 'make' to check it still compiles.

To finish, let's add a couple of new functions. One will calculate the address of the sprite pointers using the VIC registers instead of hard-coding the value into the library. The other will allow a sprite to be expanded. Enter this code into 'sprlib.c':

```
void spr_init(void)
{
  sprite_pointers = (byte*)((((CIA2.pra & 3) ^ 3) *
0x4000) | ((VIC.addr & 0xf0) * 0x40) + 0x03f8);
}

void spr_expand(byte spr, bool expandx, bool expandy)
{
 if (expandx) VIC.spr_exp_x |= 1 << spr;
 else VIC.spr_exp_x &= ~(1 << spr);

 if (expandy) VIC.spr_exp_y |= 1 << spr;
 else VIC.spr_exp_y &= ~(1 << spr);
}
```

The expression in the spr_init function isn't quite as scary as it looks. It's basically split into two parts, the first half works out which VIC bank of memory is being used and multiplies this by 0x4000 to get the base memory address for that bank. The second half of the expression reads the VIC register that controls where in memory the text screen is located and uses that to calculate where the sprites are within the VIC bank. Combining the two halves of the expression gives the actual memory address of the sprite pointers.

Next, enter the function prototypes into 'sprlib.h':

```
void spr_init(void);
void spr_expand(byte spr, bool expandx, bool expandy);
```

Now, let's do something a bit more with our new functions and show three sprites. Change 'sprtst.c' to look like this:

```
#include <string.h>
#include "sprlib.h"

#define SPRITE_DATA 0x0340
#define SPRITE0_DATA (SPRITE_DATA+0x00)
#define SPRITE1_DATA (SPRITE_DATA+0x40)
#define SPRITE2_DATA (SPRITE_DATA+0x80)

void main(void)
{
 memset((void*)SPRITE0_DATA, 0xff, 64);
 memset((void*)SPRITE2_DATA, 0xcc, 64);
 memset((void*)SPRITE1_DATA, 0xaa, 64);

 spr_init();
 spr_ptr(0, SPR_PTR(SPRITE0_DATA));
 spr_ptr(1, SPR_PTR(SPRITE1_DATA));
 spr_ptr(2, SPR_PTR(SPRITE2_DATA));
 spr_color(0, COLOR_YELLOW);
 spr_color(1, COLOR_GREEN);
 spr_color(2, COLOR_BLACK);
 spr_move(0, 290, 100);
 spr_move(1, 280, 110);
 spr_move(2, 270, 120);
 spr_expand(0, TRUE, FALSE);
 spr_expand(1, FALSE, TRUE);
 spr_expand(2, TRUE, TRUE);
 spr_show(0);
 spr_show(1);
 spr_show(2);
}
```

Compile the program with 'make' and run it in VICE to see the results.

Over to you

In the limited space of this article we haven't been able to create a complete library. There are several other functions that would be useful additions. I will leave this as an exercise for the reader to implement some of these missing functions.

To get you started with some ideas, how about writing a function to hide a particular sprite and one to hide all sprites? You could also try writing a function to set the multicolour mode of a sprite and functions to set the corresponding multicolour registers.

An implementation of these functions, plus a few more, can be found in the zip file that accompanies this article on my website, listed below.

Final words

Last time, I mentioned that this article would show how to implement a function in assembly language. Unfortunately, there wasn't enough space to include that content here. If there is enough demand I will write a third part to this series focussing on optimisation techniques and assembly language integration. Please write in to the magazine or post a comment on my blog if you would be interested in reading about this.

All my articles and their associated resources are now available at my web site which can be found at http://sites.google.com/site/develocity/commodore/articles.

Or leave a comment on any of the articles at my blog which is at http://retrodev.blogspot.com/

Confessions of a BBS Sysop Addict

Lord Ronin From Qlink

Lenard Roach of the Kansas City Commodore group wrote an article for my users group A.C.U.G about his BBS, the Pulpit. 1 received and e-mail from him asking if it would be OK to have it printed here in Commodore Free. Well our policy is that anything in our cat box liner unless otherwise stated is open for republication in C= related places.

That though gave my mind a bit of a thought about all that I can achieve these days. So going over the back issue in the stacks, I added more to my thought. What ever happened to the entire C= SysOps of the past? Well maybe with Lenard's article and the following text we may be able to hear from more of them. At least that is the hope.

How did I the world's most fanatical Commodore user set into the BBS scene? When at the start I didn't know what a BBS was or that they even existed, I was also still unaware of the differences in PC platforms between the C= and others. Yeah I thought that the Apple disk would run on the C= when I started out in I993. Actually it was an article in the groups newsletter, more used for a space filler on the single side that we had for a newsletter at that time.

Telling about a BBS called the Open Zipper Long story about getting started made short. I had a 300 baud modem and Star Term and an Amber monitor I gave it a try and was in a load of trouble at the start. Here I am on a heretic run BBS, in PET with no colour. On 40c when the presentation was 80c and in Ansi I wasn't too happy, but impressed with the fact a computer could talk over the phone to another.

Got really freaked out when text started to appear on the screen. What had I done wrong? Well nothing the SysOp broke into chat with me. Now I didn't know what "chat" or a "SysOp" was at that time, Thankfully Ben the Sysop, was once known as IceMan and ran the Ice House BBS in the 1980s in Portland Oregon, on a C=64. So he helped me set things up on my and his end, I was hooked from then on. He ran Spitfire and in my area here in Astoria and Clatsop county in Oregon.

That was the most popular BBS OS for the heretic system. In a year I was the Commodore SysOp on his board, and in charge of the PBEM (Play By Electronic Mail) Role Playing Games. Later on I was the same on 6 other boards and on two more I was the Doorgames SysOp, They couldn't believe that with NovaTcrm 9.5 I was able to do Ansi editing. Later with the 128 and Desterm I was able to bypass the fake out 80c screen on the 64. Yeah I was really hooked,

Then I had joined Q-Link about 10 months before the end of that beloved service for us. I Met some of the other diehards there and learned that most user groups had a BBS. I learned as well that they were called night time boards. Operating after 6 or at night.

OK I had one phone line for the house and the shop on the same number. I could do that and decided that the local group needed a BBS. They on the other hand, were not of the same opinion. But did that stop me. not in the least. Of course 1 had no idea of what I was getting into and had no instructions for a C= board. I didn't know that the door games where platform specific either.

What BBS OS to run? At first I tried out an off line one that sounded good; Fantasy Role Playing BBS or FRPBBS, Looked real nice in the opening screen. As I remember it was a colour fantasy sword. Wow I thought here is something that I can use for my C= interest and my RPG interest.

The idea died real soon after that part. Only had 10 message bases for PBEM. That was it in total. Top speed was 1200 and I had just gotten a 240O baud Aprotech modem, and it was limited to PET and 40c only. Hmm that would kill off the Amiga users in the local group, should they ever call. Had to have something that was C= as base line normal and allowed Ansi and ASCII What to try out?

Well there were some attempts at various offerings, but they failed mainly for lack of support for me. The less than lamer at doing all of this stuff. Some had a few documentation notes that talked in what to me was high level ML Programming. Not something for the Beginner at all, I think it was on Q-Link just before the end. Some one told me about "Color64"and where to score it up. OK it was free and that fit my budget real well.

Grabbed it, opened it, printed out the documentation and started to set it up on a 64c, 171, 1581 with a 170 clone REU. Named it the "Two Pound Micro Skirt". Well actually I used the British pound symbol for the name on the screen. Based on what 1 paid for one for my girl friend / fiancée at the time in 1969 when I was travelling through London.

Think I got it on Carnaby Street. Long time ago and a lot of good British Beer at that time. Oh she liked the skirt and dumped me because of the war. Anyway the board ran for about two months. Night time 7 nights a week. I was learning message base set up. Working on the constant problem of space for the U/D area and how to configure it. as well as collecting games for the board. At that time there were two support boards. I logged into them, long distance too I add. Gained some information that helped.

However the President of the A.C.U.G. at that time came by the shop, (where things had been set up.) He wanted my support to make the A.C.U.G. a multi platform group. Opening up to Mac and the heretic as well. He gave me a newsletter from the Multnomah County Computer Club. Or better known at that time as MC-3. They had started as a C= group and then opened up to Amiga and the other platforms. Well they listed their BBS. I decided to give that a ring and see what they had to offer.

Did that several times, in Ansi. As I had expected it to be a heretic run board. One night as I was on the board. The SysOp came into chat with me. I guess it was because of the fact that I had sent feedback to him, and had said things about my little board and being in the users group. Lo and behold, the board was running on a C= 128! He walked me through the steps to change my account online to Commodore, and I adjusted my term programme as well.

Hey it looked great. Better than that DMBBS that RMSoftware had on line. Being the only other one I had contacted. As I am not certain what Approtech ran in Rouge River Oregon. So we talked a good thing that my daily time was suspended for the chat. He gave me some facts about the BBS OS and the author's name. I was taken right off the bat.

So I contacted the author. Dr. Midi a.k.a. Brian Bell in Washington State. He said I had to have a CMD HD to make it work right. Or at the least for a small BBS, a FD-2000.I scored up the HD. took some time to pay back the money. And waited for the disks to arrive.

At that time I converted as best as possible the Colour 64 BBs onto the HD. As I also used it for my Geos work. I had to make a lot of 1581 partitions for my Geos files on the HD. Being the way things worked before Wheels. And did I mention that after spending $80 for the BBS OS I waited? Like three months, then an apology came to me.

Brian had left the envelope on a table at his parents place and spaced it out. Well there where the disks, three 1581s AIR. Which seem to have been stolen from me as of this writing. My registration number #65 was on the labels. I was now the proud owner of Omni-128 BBS. Many in the states and one outside of Munich Germany called Omni Germany. I even called that one once. Tell you right now that the long wait should have been a clue. But hey I was as polite as possibly said a novice.

Well I followed the set up as best as possible, as I paid for a manual, that to this day 15 or so years later 1 have not seen. I had a disk of SEQ files that had been taken from the SysOps message base. No help did I get from Brian. Outside of some connection work on setting up the nodes for the hub. Which at that time was either call his board Omni World and have the entire thing automated. Which is what I did; or send the proper created file as an attachment to him via email and gain the packet in reply.

If as a C= user on GEnie I could do that sort of thing. I never found out how. So 1 just called once a week to the main board. Dropped off my packet and picked up the new one. Oh yeah it took over a year and help from another Omni Sysop Mad Max of the Pink Panther BBS and head of the MH1 crew to get me to figure out how to set up the nodes for the hub files. In fact he came from Idaho twice to help me.

That is several hundred miles to drive. I also was missing files. Game files and some of the operating files. Like the BBS lister. Yeah I was getting the impression that something was wrong. Brian only told me to contact one of the SysOps Mushashi at the Civic BBS for the Civic users group in Calif. on how to edit and to programme online games. There were only 22 of them, and most where not open to non C= users, and many where limited as to a baud rate to play. 98% of them I found out later where hacks from Color64 games.

That was the birth of the, and again because of font differences. The two pound 10 and 6 Vacuum Tube BBS. Named in honour of my English mother and the fact that I studied electronics in the Vacuum Tube/Valve days. That board ran until about April of 20COcc. Sure it was C= graphics, though not exactly I must state. Took ASCII and Ansi. Also took a couple of other formats that I can't remember as no one used them. And something called SupeRcs or something like that. Which I learned much later was a specific to Omni term style. That would have made the colour and characters look better and be more accurately represented.

Oh I contacted The Coffee Shop and Huggy Bear's Den in Washington state. As they also ran Omni 128. My own callers dropped when I didn't have the games they wanted to play. Game play is the most favourite thing in my area. That and they didn't use a C= so on Omni most of the games they couldn't play. Save for the new members that I had recruited for the users group. The others never called either board. In fact they left me holding the bag for the group and all.

Despite the negatives that I presented above, I had a lot of fun with the BBS. Tried to do some things, and fried the programme in many of the games. Having to reinstall from back ups. Only one guy helped me and though he knows his stuff. Even he admits that he couldn't teach a bee to fly. Just doesn't have the ability to present the information. Found that out in the BBS work when he tried to teach me the basics of ML. I had callers from around the states and Canada call me. Some just once and others several times.

I met them in chats and talked about the C=. I had great expectations that I would be able to make the BBS into something bigger and better. If I just knew more. OK I didn't know of things called source codes and all of that ML stuff. Still don't to tell the truth. Yet I was on the board each morning and each evening. Adding files, keeping the message bases going, doing my email and doing the PBEM game turns. 1 averaged about three calls per day on a weekly basis.

1st of January 2000. Something happened to Omni. 1 had tried to call the board on the first Sunday of the month. That was my usual day for packet transfer. No connection, I tried the voice number and got the same result. Tried e-mail and same result. Big Chief in Germany had been trying for a month to contact Brian. Well I needed to send my packet. I also needed help on how to fix the dates on the files and on the message bases. They were all buggered up. To this day, there has been no contact to me or as far as I have learned to any of the Omni BBS SysOps.

We were dropped without a single word. Mad Max sent me a file to install and how to install it, that fixed the file and message base area that he had written. He also sent me a thing he had written that did a special events, and today in history thing, that popped up on login for the user, didn't work on my version of Omni. I needed something else now.

Around the end of 1999 a guy called Dr. Video was on line. He had been contacting SysOps. Telling us about this other BBS programme called Centipede. Did some e-mails with him about what Centipede had to offer. Problem with this is that I had around 5 years invested with Omni. Not that I was all that found of Omni. 1 just didn't know any other form of operation. Like the code for the up loaders comments to files on Omni is the British pound symbol. While it is the "@" symbol in Centipede. Ever try renaming by hand one at a time over 4,000 files? Not a fun task and I never did get it all done.

Still I grabbed Centipede and opened it up. Printing out this strange thing called a manual. Wow man there really was a manual, and it started me on my way. I had everything done to start. Having it on a partition of the HD that ran the Omni BBS, as it doesn’t need to be on Device 8 and partition #1 as docs Omni. Everything except the smegging files.

I just couldn't figure out how to make them show up. Read the manual and tried to do what they said. Just didn't stick in my head the way it was written. Till a SysOp I met on line said the information in a different way and it all made sense. Suddenly the couple thousand files I had put in the proper areas showed up.

I went like a mad man getting it ready to go on line. Over 100 games, that I tested out and classified. Ah I am really a mean bugger at classifying things, I over do it I am told. But I made the menus and set them up. Tested them a bit, as the guy that converted them with the tool.

Well let’s say he didn't do a full test on them. Add here as a jump ahead, that I got another game packet that isn't in the DL version, from the Hub Op at the time Dynamite. That I installed, having about 115 games. 98% of them playable by all platforms.

Well there came some tests. I had the Centipede board up at pre-announced times. Letting the regular callers see the board. I had transferred their account information, password, level and login name to their number and they took a look. All of them said move to Centipede. So in April 2000 came the birth of The Village BBS.

Well I am a big fan of the old T.V. show 'The Prisoner". Needed a smaller name than I had used before. Though there are many differences between Centipede and Omni, and I could spend pages explaining that from memory. Point of it is first that it took me time to convert my mind. Second and more important; Centipede is the BBS that a beginner should use, and not Omni as I did. Because there is a proper manual, and there are other SysOps out there that can and will help.

As I have been doing with one in California; also this is the one that most people I have heard from, use for the telnet C= boards. Not saying that this is an easy one to use, Just that it is better than the Omni one that I started with. Of course this is all based on running the BBS on a 128. So far the best that I have tried, seen and heard about for the 64 is the Color64 BBS system.

Well then for the next 5 years and a few months. The Village ran 24/7. Had a nice Moving star pattern screen saver. I was again on each morning and evening. Before going to the shop and after returning On my days off, something that I don't have any more. I spent the day doing maintenance on the board. Adding files that I had saved along with the notes I had made about the files in the up loader comments. Played some of the games; Empire ones being the most heavily played around here.

Went into the message bases added to comments, started new threads and at time made new sub areas for hot topics. Even with that, and the fact that my Geos and Wheels for both the 128 and 64, along with my Wave files for online Internet stuff. All where on the same HD. Causing me to have the board off during my maintenance and Geos work and online things. The Village averaged in a week time around 5 calls a day.

Not the gigantic numbers of the 80s and early 90s. But there were regulars that called for the PBEM and online games. Others called to see a real C= BBS again. I gained many comments about that fact. Had one guy that called from his laptop, connected to his cell phone when he was doing long haul driving. I remember him stopped at Donner Pass in the snow and calling the board. Had an Amiga user many states away that did the same with his A1200. I would send every new caller a copy of the newsletter Print at that time. Showing them that there was an active though ignorant group of C= and Amiga users, started gaining long distance members that way.

All was good for those years. 1 learned more on how to do some simple base programming in the games. Altering stats and print statements, making it more and more configured to our theme. In July of 2005 I had ordered from Maurice some spare screws and three power supply sockets for my three CMD controllers. I needed to replace the power socket as they were malfunctioning.

Having the HD mechanism slow down and the after stopping start again. If the socket and cord where wiggled just right. They arrived on 27th December 2005. I remember that for no reasons. First it would have been my mothers birthday, and second the HD with all the stuff on it, like over 5,000 files for user downloads and 2GB in compressed files from Pink Panther and the MHI crew that I had copied with permission. Plus my Geos and online stuff, and all the files for the users group. Stopped just a few days earlier.

Oh the parts were needed to make the Zip drive project. I had started on that, and with a borrowed passive cable. Actually made around 20 zip disks ready to be used with Mcopy to transfer all the BBS data and the other things to be preserved on the Zip disks. Still have the disks, but the cable was returned and the project abandoned in despair.

Since obviously the parts took 6 months to be sent and didn't make it in time to backup the data In January 2006, besides the very negative things my adopted son Mark Reed did to the users group and others. I had read on homestead mail list that Maurice had been able to do a data retrieval for one of the list members. I contacted him and he said he would look at the Hard disk mechanism. As instructed I posted it to him. He stated later that one of the heads had become stuck in an unused place.

He suspected the data to still be OK. Ah, I still have that e-mail message. Said the he had ordered and exact duplicate of my mechanism and was going to strip out the heads and then copy the data to another HD and send that to me. Great, I had offered my entire savings of $300 usd for that task.

I write this in the later part of June 2009. Still no mechanism, no answer and in his most recent this year postings on the homestead list, he said he would get to it soon. You can guess that I am not willing to pay that sum or any sum after this length of time. Most of that time with no communication from him to messages.

After waiting some months, I wanted my BBS back on line. Over the past couple of years, I have been helped on this project with tips, ideas and even hardware. Much of it coming from a man online known as Eddie, He gifted me with the power supply. Some mechanisms for the controller. But they had lovely problems like going to sleep. Most recently we are trying his hardware thing. That takes the SCSI commands from the CMD controller to the IDE thing and then to the SD cart. As well as in reverse.

Had some problems with this, and found that the jumpers where wrong. That was corrected. But still no go, not on the 2gb one that I had made, nor on the 4gb one that he had made for me. Guessing that in the installation of the two boards,which was done backwards at first. Something fried on them. As his works well, and in fact he runs Centipede on his SD card on the same boards that he makes and his board is the inner circle as well as being telnet.

Well until I can get something working on this end. The re-birth of The Village BBS is stalled. But this old BBS addict isn't down yet. Once we get this worked out, I'll be putting it back up as a direct dial board run on real C= equipment. But this time it will not share the Geos and online tools, or the SCPU. As it will be dedicated just for the BBS.

Will have a FD-2000 and at least one 5 1/4" disk drive. And the fanatical C= using old RPG playing Sysop. Oh yeah and a new pride of cats getting on it and smegging things up like the last group