The Magazine For ALL Commodore Computer Users

TPUG Magazine

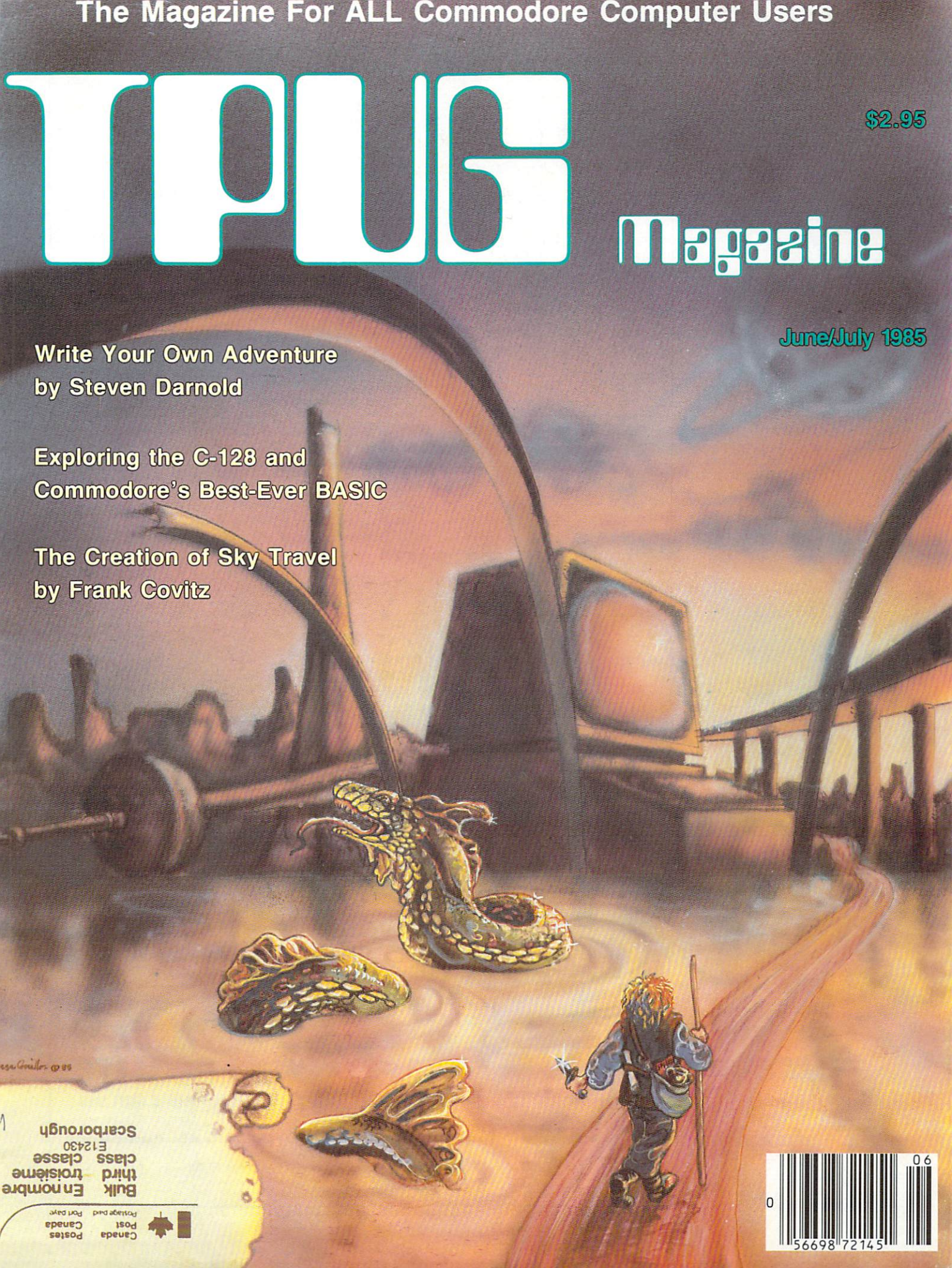
$2.95

June/July 1985

Write Your Own Adventure
by Steven Darnold

Exploring the C-128 and
Commodore's Best-Ever BASIC

The Creation of Sky Travel
by Frank Covitz



Canada Post Postage paid
Postes Canada Port payé
Bulk third class
En nombre troisième classe
E1230
Scarborough

0 56698 72145 06

BETTERWORKING
from SPINNAKER

SPREADSHEET

SPREADSHEET is the ideal program for planning budgets and preparing financial statements, as well as creating text and organizing information.
It is fast, versatile and powerful enough to handle all your spreadsheet needs.

And it's easy enough to understand and use right away.

SPREADSHEET combines a full-feature spreadsheet with basic word-processing and the sorting and searching capabilities of a database.

$59.95

NEW FROM EPYX!

SUMMER GAMES II™

- Includes Cycling, Fencing, Kayaking, Triple Jump, Rowing, High Jump, Javelin and Equestrian Events
- Opening, Closing and Awards Ceremony With National Anthems
- Compete Against The Computer or Your Friends
- Individual and Head-To-Head Competition
- One to Eight Players

$39.95

JET COMBAT SIMULATOR™

- Features Realistic Flight and Combat Simulation
- Includes Take-Offs, Landings, and Aerial Dogfights
- Choose Between Practice or Combat Modes
- Select from Four Levels of Difficulty
- One Player, Joystick Optional

$39.95

Butterfield's Machine Language

$17.95

The complete guide to the 1541 operating system.

$25.95

$39.95

PaperClip
(the word processor)

$79.95
with SpellPack $99.95

Consultant
(the database)

$124.95

The ultimate program for learning chess

$69.95

A must book if you own a VIC!

$14.95

Boss Joystick

$23.95

Kraft Joystick

$17.95

COMPUTE!s First Book of Commodore 64 Games

Packed full of games: 'Snake Escape', 'Oil Tycoon', 'Laser Gunner', 'Zuider Zee', and many more. Machine language as well as strategy games. Ideas and techniques for writing games.

$18.95

A User's Guide

$14.95

Grand Master Chess

$24.95

Memorex Disks

(Special) SSDD $19.90/10

BASF

DISKS SSDD $24.90 in hard box

Cassettes

C10 Tapes $1.00 ea.
C20 Tapes $1.25 ea.
Cassette boxes .25 ea.

Ribbons

1525	$12.95
1526	$14.95
4023	$14.95
Spirit 80	$14.95
801	$12.95
802	$14.95
8023	$14.95
4022	$12.95
MX80	9.95
RX80	9.95

CABLES
for disk drive or printer

6ft. replacement cable **$14.95**

6ft. extension cable **$14.95**

Adventure Construction Set

$69.95

ELECTRONICS 2001 LTD.

Name _____
Address _____
_____

☐ Check or money order enclosed ☐ Visa ☐ MasterCharge ☐ American Express
Acct.# _____ Exp. Date _____ Signature _____

ELECTRONICS 2001 LTD.

5529 Yonge Street, Willowdale, Ontario M2N 5S3
Tel: (416) 223-8400

Note: All prices in Canadian Funds. Phone and mail orders welcome.
Ontario residents add 7% sales tax.
Add 5% for shipping (minimum charge $2.00)

TPUG Magazine

Publisher: Louise Redgers
Editor: Nick Sullivan
Assistant Editor: Marya Miller
Director of Advertising Sales: Louise Redgers
Production Assistant: Astrid Kumas
Art Director: Teresa Coviello
Computer Jockey: Malcolm O'Brien
Cover Photo: Roberto Portolese Studio
Typesetting: Noesis, Toronto, Ontario
Printing: Delta Web Graphics, Scarborough, Ontario

TPUG Magazine is published 10 times a year by Toronto Pet Users' Group (TPUG) Inc., the world's largest Commodore users' group. TPUG is a non-profit corporation dedicated to the service and support of owners and users of Commodore computers.

Correspondence: Send change of address and subscription enquiries to: TPUG Inc., Address Changes, 1912A Avenue Road, Suite 1, Toronto ON, Canada M5M 4A1. TPUG Magazine welcomes freelance contributions on all aspects of Commodore computing. Contributions should be sent on disk, though accompanying hardcopy is welcome. Be sure to include return postage if you wish materials returned. Please indicate on the disk label which Commodore disk format and word processing program you have used. Remuneration for articles published is $30.00 per magazine page if the author retains the copyright, and $40.00 per magazine page if the copyright is assigned to TPUG Magazine. All contributions are subject to editing for length and readability. Address editorial contributions to: The Editor, TPUG Magazine, 1912A Avenue Road, Suite 5, Toronto ON, Canada M5M 4A1. Letters to the editor may be sent to the same address. Like other material, letters may be edited for length and readability.

Circulation:
Subscription 16,000 Newsstand 10,000

ISBN #0825-0367

TPUG Magazine is distributed by: Compulit Distributors, PO Box 352, Port Coquitlam BC, Canada V3C 4K6, (604)-464-1221; and Micron Distributing, 409 Queen Street W., Toronto ON, Canada M5V 2A5, (416)-593-9862, Toll Free Order Desk 1-800-268-1238.

Subscriptions to TPUG Magazine may be obtained by joining the Toronto Pet Users' Group (TPUG), Inc. Another benefit of TPUG membership is the right to order inexpensive disks and tapes from the club's extensive software library. Yearly membership fees are:

Regular member (attends meetings)	$35.00 Cdn.
Student (full-time, attends meetings)	$25.00 Cdn.
Associate (Canada)	$25.00 Cdn.
Associate (U.S.)	$25.00 U.S.
	$30.00 Cdn.
Associate (Overseas — sea mail)	$35.00 U.S.
Associate (Overseas — air mail)	$45.00 U.S.

For further membership information please contact:

TPUG Inc., Membership Information
1912A Avenue Road
Suite 1
Toronto ON
Canada M5M 4A1

1552 Hertel Ave.
Suite 144
Buffalo, NY
USA 14216-2882

TPUG Telephone Numbers:
Business Office (416)-782-8900, (416)-782-9252
Magazine Office (416)-782-1861
Advertising (416)-782-1861

DIRECTORY

Changes, Changes, Changes

Dear TPUG members:

In this world of ours, nothing is static. Nothing remains the same for long enough to focus on it, let alone take the picture or get it developed. Especially in the world of microcomputers, where today's success is tomorrow's ho-hum. For those of you who enjoy the ever-changing scenery and look forward to tomorrow's sky-line, you're in the right business. As someone said about Toronto's weather — if you don't like it — just wait!

And so it is with TPUG. The faces change as we try to keep on top of the ever-changing world. It is with great regret that we say farewell to Chris Bennett, who is a founding director and who managed the club through its meteoric growth to 15,000 members, and to Doris Bradley, who helped with this process and whom so many of you know on the other end of the telephone. Both have departed TPUG, and we wish them both well in their future ventures (Chris remains on the Board of Directors and is looking forward to the beginning of the PC 10 Interest Group in the fall).

We, the directors of TPUG, owe a great deal of gratitude to both these people, who put so much into the club during its formative years. They were hectic times, and both Chris and Doris were instrumental in holding on as the scope of the club expanded to include *TPUG Magazine* as you know it, our library of over 5,000 programs, the many monthly meetings, and the conference.

Thank you, Chris. Thank you, Doris.

Michael Bonnycastle
TPUG President

Message From The Publisher

Summer is here. I hope that yours is relaxing and lazy, and filled with all of the things that you like to do best. This year we will not be publishing separate July and August issues. The current issue is June/July; the next one will be August/September. This will allow our exhausted magazine staff to visit their families and also enjoy a bit of the summer. I wish to thank them and also our front office staff for their dedication and hard work over the last few months, as we have been making the changes that provide you with this new and improved magazine. We are still under construction, with more good things to come.

We apologize for the very late May issue. From the start we were plagued by problems with that issue — Murphy's Law has never been more convincingly proved than by us during the month of April. We will endeavour to prevent this happening again. Thank you for your patience.

This month's spectacular 'adventure' cover was painted by our art director, Teresa Coviello. We like it so much that we have had it printed up in full-size poster format as well, on good quality art paper. Watch for an announcement in the August/September issue on how you can order copies by mail.

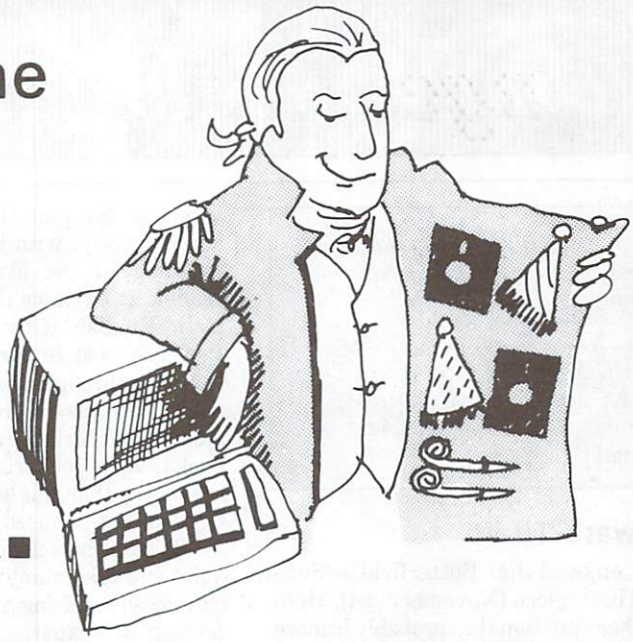
As usual, there will be no meetings during July and August, with the exception of the SuperPET chapter. SuperPET meetings will probably be held as usual, on the third Wednesday of each month. Call the TPUG meetings line for up-to-date information.

Finally, TPUG would like to hear from those of you who have placed orders with B & R Enterprises of Pefferlaw, Ontario, and have not received satisfaction. Please write to me at TPUG. If you were planning on placing an order, please hold off at this time. There have been several complaints and we are trying to find out why. We will keep you posted.

Once again, enjoy your summer. We will see you in September with all the new machines (Commodore Canada promises two more after the C-128 and PC 10 before the end of the year), new groups and new interests.

Louise Redgers
Publisher

You're invited to the biggest party at Valley Forge since George brought the boys!



M.A.R.C.A.

The biggest Commodore User Fair in the US.

July 26, 27, 28

Valley Forge Convention Center, Valley Forge PA

- Speakers! • Seminars! • Hanging out!
- Fun! • Vendors! • Great Buys!
- Social Events! • Fun!

Areas Tours available.

Meet the names you've only read about. Jim Butterfield. Dick Immers. Len Lindsey. Many, many more! Ask the questions you need answers to. Have 2½ days of non-stop Commodore fun! Bring the whole family. Lots to do. See. And buy. Bargains galore!

Pre-registration by July 1: 2½ days $25 M.A.R.C.A. Members $15 Family Rates available.

For pre-registration information: M.A.R.C.A., P.O. Box 1902, Martinsburg, West VA 25401.

DON'T MISS THE PARTY!

LINE NOISE

TPUG Magazine *invites you to express your views on Commodore computing by writing to:*

Line Noise
TPUG Magazine
1912A Avenue Road, Suite 5
Toronto, Ontario M5M 4A1
Canada

Software Theft

Really enjoyed Jim Butterfield's 'Software Theft' piece (November '84). Here is another situation that probably is more prevalent than most people realize.

My son moved to California and became employed by a software marketing firm. I had written a program to handle my checkbook up to a limit of 99 asset, liability, income and expense accounts. It was simple, fairly fast, and it worked (usually). He suggested that their company market this program.

A royalty agreement was drawn up and signed whereby they did the marketing, using my name as author. I was to receive 20 per cent of the gross sales of the package. There were numerous other p's and q's but basically, I wrote it and they sold it.

The package moved fairly well, and I hinted that some money might be welcome. I did get one check for $2279.23, which bounced (three times). To date, this has been the extent of my payment by them. It is hard to determine how many of the packages were sold, but I do have a letter indicating they owed me $11,734.87.

W.E. (Dinty) More
Le Seur, Minnesota

Butterfield Book A Gem

As a somewhat isolated, relatively aged, neophyte to computers, my primary means of learning is through the printed word. I have, therefore, read many books and magazines in the year and a half since I contracted computermania. While I have obtained some proficiency in BASIC, machine language and hardware were areas of total illiteracy. I would not presume to write a review of a computer book. My knowledge is too limited. But Jim Butterfield's book on machine language is a gem. It is written clearly and concisely, with humour and understanding of the likely pitfalls of the reader. It explains difficult concepts in plain English without use of jargon. Perhaps most important, it guides the beginner through simple short programs which, with the exercises, give the reader hands-on experience and confidence. His book is very well written — considerably less wordy than this letter. It is a pleasure to read and painless to learn from.

While on the subject of books, I do acquire and read many on computers. You may recall receiving a letter from me concerning my experiences with dilithium Press. I have since received a handwritten note to the effect that they are in bankruptcy proceedings. From my side, this seems to match my experience with them.

Thank you, and keep up the good work.

E.M. Hartston
West Galilee, Israel

BASIC Blitz!ed

During further use of the **Blitz! Compiler**, which I reviewed in the March issue, I found that it only compiles BASIC programs into the standard 38K BASIC area of the C-64, starting at decimal 2048.

BASIC cannot be moved when using a **Blitz!**ed program, although the upper end of BASIC can be lowered in the normal way to allow ML programs to be put just below the BASIC ROM.

Nathan Okun
Oxnard, California

Evaluating Evaluation Kits

Has anyone else out there tried submitting programs to Commodore for publication? It seems that Commodore won't look at a piece of software until you do what is required of you from Commodore's **Software Evaluation Kit**. This kit was new to me, so I sent it for a legal 'check-up'. It failed. It had bugs that would lose me my program to Commodore if I did what it asked, which is the following:

- Read the kit, front to back.
- Fill out the questionnaire.
- Send a copy of the program to Commodore on disk with literature, any copyrighted documents, and a money order for fifty dollars.
- Allow thirty to ninety days for evaluation. After ninety days, Commodore notifies you as to whether or not they consider your product marketable.

Now, I did not exactly cherish the thought of sending the fifty dollars and my program to Commodore. It just didn't seem right.

My advice to all programmmers is to stay away from 'Evaluation Kits' from any company and agree to a 'hands-on' demonstration, at their convenience.

Alex Howell
Toronto, Ontario

Commodore is apparently getting out of the software business altogether, so legal questions concerning their evaluation kit have become academic. However, programmers should in general make sure that they have clear proof of authorship of any product they submit to software publishers.

Fast Load Lament

Tired of my slow 1541 drive, I bought the Epyx **Fast Load** cartridge. I use my C-64 mainly for word processing, with **PaperClip**. With **Fast Load**, on numerous occasions, the drive or computer have locked up, resulting in lost files. On two occasions, I lost an entire disk when the directory was damaged. Usually, I can reset everything by turning off the drive, but I have sometimes had to turn off the C-64 as well, losing the current file. **Fast Load** is virtually useless to me. Am I the only one with these problems? Has anyone contacted the manufacturer for 'fixes', or does anyone have suggestions for a 'do-it-yourself' fix?

Achim K. Krull
Agincourt, Ontario

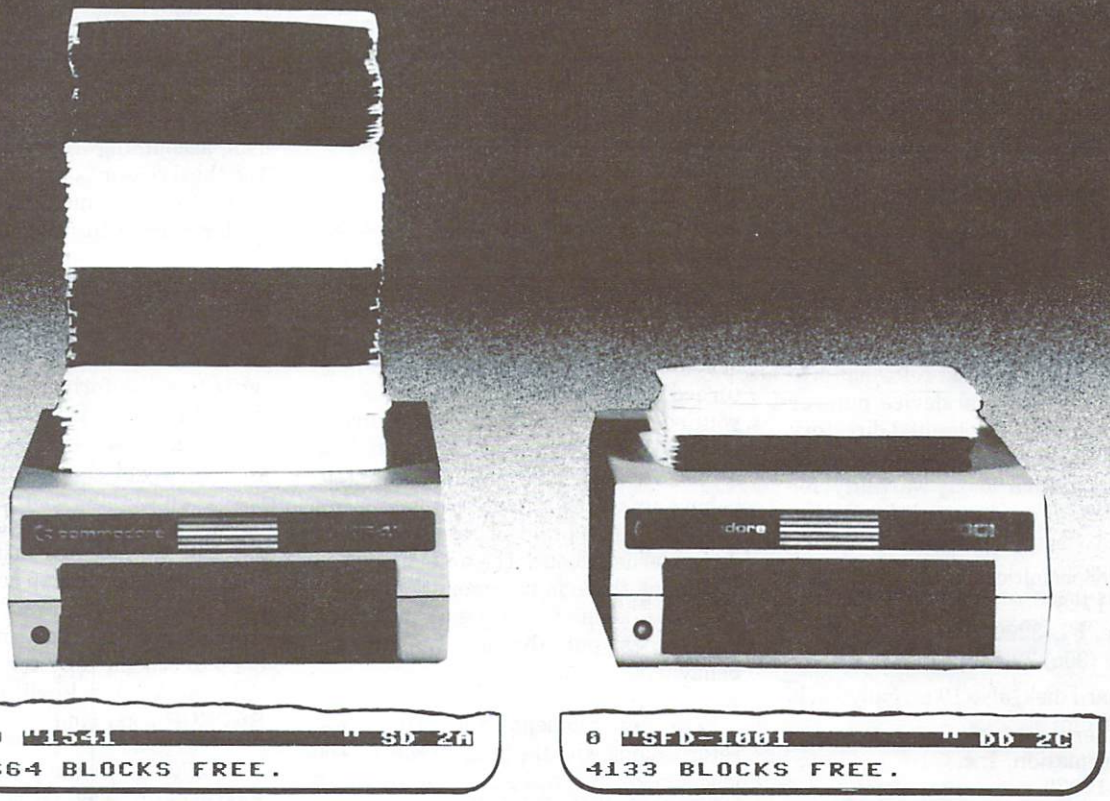
*The first release of **Fast Load** apparently did have bugs that can cause the problems you describe. I have heard that Epyx will replace the defective ones if you contact them. The new versions of **Fast Load** that are on the market are said to work very well and should give you no trouble.*

***Line Noise** presented by Lana Coviello*

NEW

commodore SFD—1001 ONE MEGABYTE DISK DRIVE

NEW



```
0 "1541              " 8D 2A
664 BLOCKS FREE.
```

```
0 "SFD-1001          " DD 2C
4133 BLOCKS FREE.
```

The **SFD-1001** (Super Fast Drive) is now available. With double-sided double-density format, over ONE MEGABYTE can be stored on a single floppy disk. ONE HUNDRED 1541-formatted disks can be reduced to only SIXTEEN **SFD-1001**-formatted disks. By using the intelligent IEEE bus and a bus expansion IEEE interface, the **SFD-1001** loads programs and data over TWICE as fast as the 1541 drive, and ALL THIS inside a case the size of the 1541's!

FULLY COMPATIBLE with any Commodore computer that has an IEEE interface. **FREE** utility disks for both the CBM 8032 and the Commodore 64 are included! Transfer all your files and programs easily from any Commodore disk drive to your **SFD-1001**!

EXPAND your system now with this fast, high-quality, large capacity Commodore disk drive.

The **SFD-1001** is available NOW from Progressive Peripherals & Software, Inc., your quality Commodore software and hardware source. Suggested retail price is only $399.95.
Dealer inquiries are invited...call for more information or for the name of the dealer nearest you.

LOAD TIMES
The 1541 loads 32K bytes of data in approximately 1 minute, 20 seconds.
The **SFD-1001** loads 32K bytes of data in only about 35 seconds (bus expansion interface) or approximately 1 minute, 4 seconds (serial interface).

PROGRESSIVE PERIPHERALS & SOFTWARE
2186 South Holly Suite 200
Denver, Colorado 80222
(303) 759-5713 Telex:888837 TWX:9109971314

The Answer Desk

with Malcolm O'Brien

Hard Disk Update

Shortly after the last *Answer Desk* had begun its long journey to your hands, I discovered that no less than three companies are now offering hard disks for Commodore products. Strangely, only one — Computer Specialties, Inc. — has sent promotional material to the TPUG office complex, so I can only give you details of their product, the CSI ST10C.

Available for all Commodore products except the VIC 20, the CSI ST10C features 10 megabytes with serial and IEEE interfaces, external reformat protection switch, external device number select switch (8 or 9), unlimited directory space, and backup software. All of this can be yours (with a 90-day warranty) for $1595 (US). For more information contact:

Computer Specialties, Inc.
P.O. Box 1728
Melbourne, FL 32902
Telephone (305)-725-6574

Another hard disk, also 10 megabytes, is available from:

Fiscal Information, Inc.
P.O. Box 10270
Daytona Beach, FL 32020

Rounding out the list is the Genesis drive from:

Micro Mind
104 Hawthorne Ave.
Pittsfield, MA 01201

This one is a 25-megabyte, half-height Winchester drive, to retail for $1200 (US). Micro Mind also has plans to produce a hard drive to be installed in the small storage box in the SX-64. Watch for further details in upcoming issues of *TPUG Magazine*.

How to use *Fastbackup*

Instead of answering your letters, this month I'm going to answer your phone calls. Most days, TPUG receives telephone calls from C-64 users who either want to know how to use **Fastbackup**, or who simply state that it doesn't work. The fact is that it does work — and very well too! If you've been having trouble with the program, read on.

First of all, disconnect your printer. Don't just turn it off — disconnect it. Evidently, **Fastbackup** must have the serial bus all to itself. Now load **Fastbackup** from TPUG disk (C)TA, then take the disk out of the drive before running it. Put the disk that you want to copy (called the 'source' disk) into the drive, and run **Fastbackup**.

In the upper left corner of the screen you will see a quick countdown. When it finishes, the title screen will come up. On the last line of the display you will see the prompt *Source disk*, and you will hear a beeping sound (make sure the volume is turned up on your TV or monitor). Since your source disk is already in the drive, hit **RETURN**. Now the magic begins. The screen vanishes and the drive begins to behave in a very strange fashion — but normal for this program, and nothing to be concerned about. The error light will flash, but there is no error. Sometimes the drive keeps turning even after the light goes out. Ignore the drive's behaviour.

In a few moments the screen will return, along with the beeping sound. The last screen line will show the message *Destination disk*. This is the disk that you are copying to. You may use an unformatted disk, if you want — fresh out of the box; **Fastbackup** formats the destination disk for you. If you use an old disk, of course, all the data will be overwritten.

At this point — with *Destination disk* on the bottom line — carefully remove the source disk and insert the destination disk. Again, don't worry if the drive is still turning! When the destination disk is in place, hit **RETURN**. The audio and video will again disappear. Follow the program instructions as above until the copy is complete (you should have to insert both disks a total of three times each). Now the bottom line on the screen will read *Copy complete*. You'll have to power down the computer to regain control.

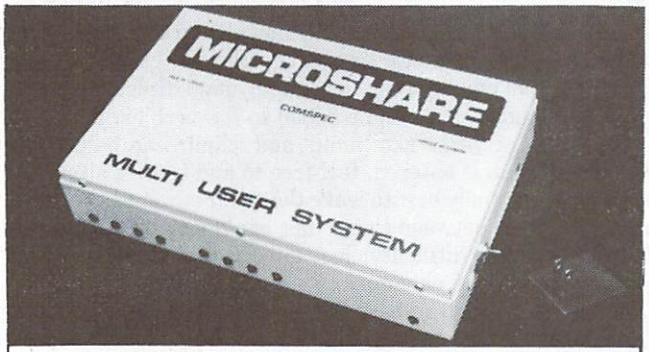
There are three common reasons why you would use **Fastbackup** to copy disks. The first is to make an archival backup of commercial software. This will work with **Paperclip** or **The Consultant**, for example, because these disks are not protected. Instead, they rely on an external dongle or key for the program to operate. Most commercial programs do use disk protection, however. In these instances, 'errors' on the disk will cause the copy to fail. The second reason for backing up is to have a spare copy of your data — **Paperclip** files, spreadsheets, mailing lists, accounting information and so on. The third reason is probably the most important one for me, and will be of particular interest to students and teachers.

Many schools have PETs, SuperPETs, 4032s and 8032s. Most of these are equipped with the CBM 4040 Dual Disk Drives. You should not use a 1541 to write to a disk formatted on one of these drives! The 1541 is read-compatible with them, but *not* write-compatible. If you do, your files will slowly disappear! If you have been working on a disk at school and intend to finish writing your program (or entering your data) at home on your 1541, use **Fastbackup** to copy the 4040 disk to 1541 format. When you've finished updating your files, and take the disk back to school, copy the 1541 disk back onto the 4040 disk with the BASIC 4.0 **BACKUP** command.

Finally, I should remind you that **Fastbackup** was written by Thomas Tempelman as a commercial venture and TPUG is accepting contributions for use of the program on Mr. Tempelman's behalf. Over $1,000 has already been sent. Response has been encouraging, and Mr. Tempelman has kindly allowed TPUG to distribute his second version of **Fastbackup**.

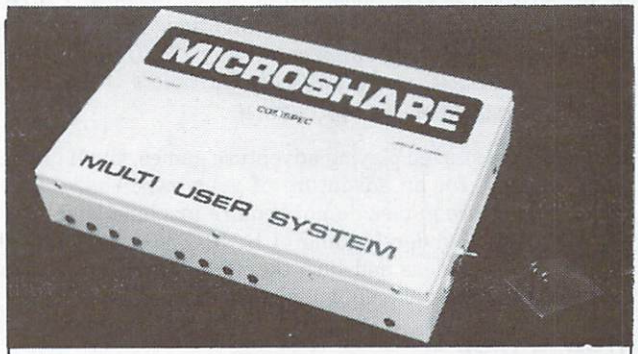
The new version is called **fcopy v2.5.c** and it's on TPUG disk (C)TG. This version is even faster (!), and includes a message from Mr. Tempelman along with his address in Germany. It is slightly different in that the destination disk must be formatted beforehand. Also, the speed-up is due to the fact that only the used portion of the source disk is copied. You are even presented with a graphic display of the used portions of the disk. If you are using either of these programs, please send a contribution if you haven't already. It would be a terrible waste of a good programmer if his financial situation forced him to take a job as a shoe salesman! □

MICROSHARE



PET/CBM MULTI USER DISK SYSTEM

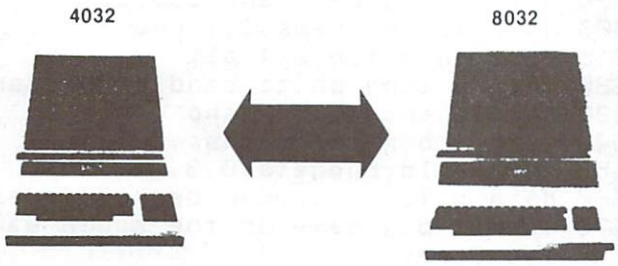
- ALLOWS UP TO SIXTEEN USERS TO SHARE DISK DRIVES AND/OR PRINTERS
- WORKS WITH ALL PET/CBM EQUIPMENT
- 100% HARDWARE INTERFACED
- NO ALTERATIONS TO SOFTWARE OR SPECIAL SOFTWARE REQUIRED
- SOFTWARE TRANSPARENT — WORKS WITH ALL PET/CBM SOFTWARE
- LANGUAGE TRANSPARENT — WORKS IN ANY LANGUAGE
- NO SPECIAL COMMANDS USED
- PROTECTS AGAINST SYSTEM LOCKUP



COMMODORE 64 MULTI USER DISK SYSTEM

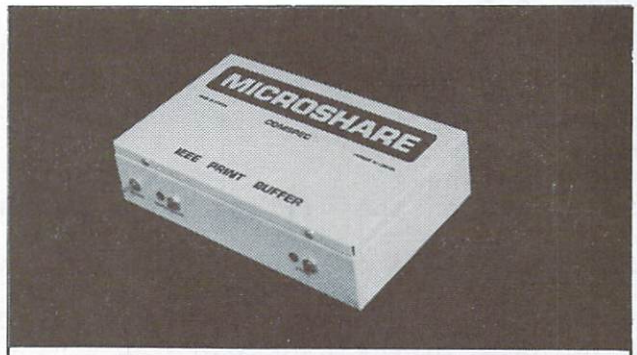
- ALLOWS UP TO EIGHT USERS TO SHARE DISK DRIVES AND/OR PRINTERS
- WORKS WITH ALL 64/VIC EQUIPMENT
- **BUILT IN IEEE AND 64/VIC SERIAL PORTS** (WORKS WITH ALL IEEE DEVICES)
- 100% HARDWARE INTERFACED
- NO ALTERATIONS TO SOFTWARE OR SPECIAL SOFTWARE REQUIRED — SOFTWARE TRANSPARENT
- NO SPECIAL COMMANDS USED
- **BUILT IN 16K PRINT BUFFER**
- DISK DRIVE PRIORITY

GREATER PRODUCTIVITY & LOWER COST FOR BUSINESS AND EDUCATIONAL USE



40 TO 80 COLUMN CONVERSION

- EXTERNALLY SWITCHABLE FROM 40 TO 80 COLUMNS
- EXTERNALLY SWITCHABLE FROM 80 TO 40 COLUMNS
- CONVERSIONS FOR BOTH 4032's AND 8032's
- 100% SOFTWARE COMPATIBILITY IN EITHER MODE
- HARDWARE (NOT SOFTWARE) MODIFICATION
- ALL KEYS FROM 8032 EMULATED ON 4032
- SPECIAL FUNCTION KEYS



MICROSHARE 64K PRINT BUFFER

- LOW COST
- REDUCES LONG WAITS — SAVES TIME
- ALLOWS YOU TO PRINT AND PROCESS SIMULTANEOUSLY
- IEEE INPUT — IEEE OR CENTRONICS PARALLEL OUTPUT
- WORKS WITH ALL PET/CBM SOFTWARE
- NO INSTALLATION REQUIRED
- ELIMINATES THE FRUSTRATION OF WATING FOR YOUR PRINTER

DEALER ENQUIRIES INVITED

MANUFACTURED BY:
COMSPEC COMMUNICATIONS INC.
153 BRIDGELAND AVE., UNIT 5,
TORONTO, ONTARIO M6A 2Y6
(416) 787-0617

Do-It-Yourself Adventures

by Steven Darnold

Steven Darnold has written a number of classic adventure games for the TPUG library. Some of his games, including ***African Adventure****,* ***Castlemaze*** *and* ***Tunnelmaze*** *are now being rereleased along with other adventures on TPUG disk (C)GG for Commodore 64. Also included will be Darnold's newest, largest and most complex work,* ***Valleymaze****.*

Once you have started playing adventure games, you'll probably get some ideas for an adventure of your own. Unlike arcade games, adventure games do not require machine language. If you have a good understanding of BASIC, you should be able to develop these ideas and turn them into an effective adventure program. This article will show you how.

Draw a map

The first step in designing an adventure is to draw a map of all the locations, and the connections between them. Number the locations, starting with 1, and think of a brief description for each. You will be able to amend this information later, so don't worry if you're not sure how the final map will look. Just make a start, and everything will eventually fall into place.

Once your map is ready, you can enter it into the computer. It may seem a little strange, entering information before the adventure is completely designed, but I find it much simpler this way. Adventure programs tend to be very long, and it is easy to be daunted by the size of the task. If you do things step by step, the program will be nearly finished before you fully realize how big and complicated it really is.

Data — the rooms

```
9000 r9=6
9010 dim r$(r9),r%(r9,3)
9020 for i=1 to r9
9030 read r$(i)
9040 for j=0 to 3
9050 read r%(i,j)
9060 next:next
9070 data on the village green,3,0,4,
     2
9080 data in a one-room school,0,0,1,
     6
9090 data in a rustic church,0,1,0,0
9100 data in the bank,0,0,0,1
9110 data in the safe,4,0,0,0
9120 data in a dusty storeroom,0,0,2,
     0
```

The first line specifies how many rooms (locations) there are. In this simple example, I have six. In your adventure, set *R9* to the total number of locations on your map. The descriptions begin at line 9070. Following each description are four numbers that show the connections between the rooms. The first number shows the connection north; the second, south; the third, east; and the last, west. For example, if you are in room 5 (the safe), the connection north goes to room 4. A zero indicates that there is no connection in that direction. Thus, the only exit from room 6 is to the east (it goes to the school).

Lines 9010-9060 set up two arrays: one for the room descriptions and one for the connections. You can use these lines exactly as they appear here.

Some adventure routines permit more than four exits from a room by also recognizing 'up' and 'down', and sometimes even northeast, southeast, southwest and northwest. I find it simpler to limit the movement array to the four main directions. If required, 'up' and 'down' can be added to the verb list and dealt with directly; or verbs like 'jump' and 'climb' can be used.

Once your map is entered, feel free to add to it or alter it at any time. It is usually best to leave the order of rooms the same, but a connection between two rooms can normally be changed without too much difficulty: just alter the numbers for both rooms to reflect the change.

Adding rooms is even easier. Increment the value of *R9* in line 9000, and add the new room to the bottom of the list; then alter the data of any of the existing rooms you want connected to the new room.

Now that you have typed in the details of the rooms, it's time to enter the objects you intend to use in the adventure. These include treasures, tools and assorted bits of scenery. Number the objects, starting with 1.

Data — objects and scenery

```
9500 n9=13
9510 dim o$(n9),l%(n9),a%(n9),n$(n9)
9520 for i=1 to n9
9530 read o$(i),l%(i),a%(i),n$(i)
9540 next
9550 data a piece of chalk,2,2,cha
9560 data a huge oak tree,1,0,tre
9570 data a sign,6,2,sig
9580 data a brass key,0,2,key
9590 data a bag of coins,5,3,coi
9600 data wooden pews,3,1,pew
9610 data an altar,3,1,alt
9620 data a long white candle,3,2,can
9630 data a shovel,6,2,sho
9640 data a box of matches,0,2,mat
9650 data gold nuggets,0,3,gol
9660 data a large wooden desk,2,1,des
9670 data a big safe in the south wal
     l,4,0,saf
```

In the first line, set *N9* to the total number of objects you wish to have. In my sample adventure, I use thirteen objects, but you will probably have around one hundred. Each object is followed by two numbers. The first number specifies in which room the object is located at the beginning of the adventure. A zero indicates that the object is not in any room. The gold nuggets, for example, need to be dug up before they appear.

The second number following each object is an attribute value, used to distinguish between the different types of objects. Treasures have a 3; other movable objects have a 2; immovable objects have a 1; and scenery has a 0. The altar, for example, has a 1 to indicate it is an immovable object, while the safe is classed as scenery.

The last item in each data line is a three-letter name for the

object. This is used to identify it, so be careful to give each object a different name. Normally, you should use the first three letters of the main noun in the object's description.

Don't worry about working out all your objects in advance. It's easy to add new objects as you go along. Simply increment *N9* and add the details of the new object to the bottom of the data in this section.

The next step in setting up the adventure is to list the verbs you are going to recognize. Number them, starting with 1, and set *V9* equal to the total.

Verbs

```
9900 v9=23
9910 dim v$(v9)
9920 for i=1 to v9
9930 read v$(i):next
9940 data n,nor,s,sou,e,eas,w,wes
9950 data go,inv,sco,loo,exa
9960 data taK,get,pic,dro,put
9970 data rea,ope,dig,lig,qui
```

As with the object names, each verb is referred to by its first three letters. In addition, the four directions also appear as single letters. Thus, in my sample adventure, verb number 7 is W and number 8 is WES. Both refer to west.

It is important to have a lot of redundancy in your verb list. Extra verbs cost only a few bytes each, and they make your program seem much more intelligent. Many players become frustrated trying to guess which words an adventure understands. Make it easy for them: include redundant verbs. (For example, in my sample I have included TAKE, GET and PICK UP. Each of these three verbs is dealt with in precisely the same way, but the redundancy means that players are more able to express themselves in their own way.)

Completing the foundation

Now that we have typed in the rooms, objects and verbs, we have nearly completed the foundation of the adventure. All that remains is a little housekeeping:

```
9450 dim d$(3)
9460 for i=0 to 3:read d$(i):next
9470 data north,south,east,west
```

This array is simply a convenient way of printing out directions. It will be used later in the main program loop.

```
9990 r=1
```

This sets the adventure to start in room 1. Feel free to change it to any room you like.

```
9999 return
```

This marks the end of the foundation section and returns to the main flow of the program.

```
100 gosub 9000
```

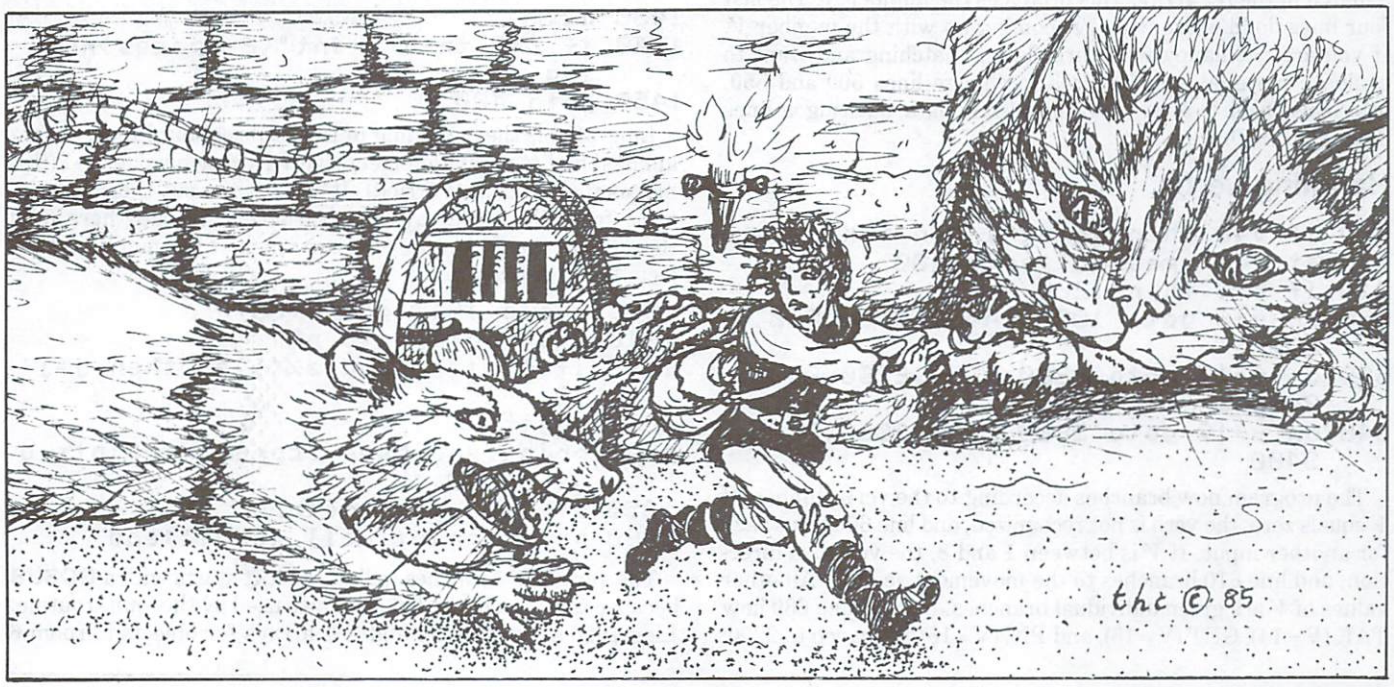
The foundation section is entered by a **GOSUB** from the very first line of the program. If you wish to display a title page or instructions, put them at lines 8000-8999 and change line 100 to **GOSUB 8000**.

Building on the foundation

We are now ready to build the main program loop. It will run between lines 200 and 700, and will be followed by verb branches, starting at line 1000.

```
200 print chr$(147)
210 if r=6 and ca=0 then print"it's t
    oo dark to see anything.":goto 30
    0
220 print"you're "r$(r)"."
230 print
240 v$="you see "
250 for i=1 to n9: if l%(i)=r then
    print v$;o$(i):v$="(8 spaces)"
260 next
```

These lines are executed whenever the player enters a new room. Line 220 prints the description for room *R* from the *R$* array. Line 250 checks the *L%* array for the location of each object; and, if it's in room *R*, prints its description from the *O$* array. Line 210 shows how to implement a dark room by skipping over the descriptions for room 6 if the candle is not lit



(CA = 0). You can use all of these lines without alteration in your own adventure, except line 210.

```
300 print
310 print"you can go:<2 spaces>";
320 for i=0 to 3
330 if r%(r,i)>0 then print d$(i)" ";
340 next:print
```

These lines print the exits for room *R*. Line 330 simply checks the four directions for non-zero values. The *D$* array is used to print north, south, east or west.

Decoding player commands

```
400 print:print"what now";
410 input a$:print
420 n$="":v$="":v=0:n=0:j=0
430 l=len(a$):for i=1 to l
440 if mid$(a$,i,1)=" "then v$=left$(
    a$,i-1):j=i:i=l
450 next:if j=0 then v$=a$:goto 550
460 for i=l to j step-1
470 if mid$(a$,i,1)=" "then n$=mid$(a
    $,i+1):i=j
480 next
```

These lines input a command from the user, and parse it. Line 440 hunts for a space, starting from the front, and line 470 hunts for a space, starting from the back. The first word becomes *V$* and the last word becomes *N$*. Some adventure schemes use the second word for *N$*, but I find the above method gives better results. For example, it understands PUT DOWN THE BAG OF COINS.

```
500 t$=left$(n$,3)
510 for i=1 to n9
520 if t$=n$(i) then n=i:i=n9
530 next
550 t$=left$(v$,3)
560 for i=1 to v9
570 if t$=v$(i) then v=i:i=v9
580 next
```

The first four lines reduce *N$* to three letters, then look for a match in the *N$* array. This produces the number *N*. The last four lines do likewise with *V$*, coming up with the number *V*. If you are not happy with three-letter matching and want to evaluate more letters, feel free to change lines 500 and 550. You'll also have to alter your three-letter data, starting at lines 9550 and 9940.

Handling verbs

```
600 if v=0 then print"i don't know wh
    at "v$" means.":goto 400
610 if v<9 then 1200
620 on v-8 goto 1000,1400,1600,1800,1
    800
630 on v-13 goto 2200,2200,2200,2400,
    2400
640 on v-18 goto 2600,2800,3000,3200,
    3400
```

The program now branches according to the verb number. If *V* equals zero, the verb is not recognized, and line 600 loops back for another input. If *V* is between 1 and 8, the verb is a direction, and line 610 branches to the movement section. All other values of *V* are given individual branches. Note in line 630 how TAK (V = 14), GET (V = 15), and PIC (V = 16) all branch to 2200; and how DRO (V = 17) and PUT (V = 18) all branch to 2400.

The **ON-GOTO** structure handles the verb branches very quickly, so do not hesitate to add redundant verbs. Simply add the verb to the bottom of the list and put its branch address on the end of the last **ON-GOTO**. For example, in my simple adventure the verb OPE (V = 20) is used to open the safe. However, it's quite likely that some players will try to use the verb UNLOCK. Why frustrate them? Just add UNL to the end of line 9970, change *V9* to 24 in line 9900, and add 2800 to the end of line 640.

Verb branches begin at line 1000, with at least 200 line numbers allocated to each verb. Some verbs will require only a few lines; however, it's a good idea to leave room for unexpected developments.

```
1000 if n$="" then print"go where?":
     goto 400
1010 if n=0 then v$=n$:n$="":goto 550
1020 print"use compass directions.":
     goto 400
```

The GO section begins at line 1000. If *N$* is recognized as a noun, line 1020 is executed. However, *N$* usually holds a direction (a verb!), and line 1010 transfers it to *V$*, looping back for re-evaluation. In my sample adventure, I do not recognize any direct use of GO. However, if you wish to deal with commands like: GO STAIRS or GO DOOR, simply begin your routines at line 1020.

```
1200 v=(v-1)/2:i=r%(r,v)
1210 if i then r=i:goto 200
1220 print"you can't go that way.":
     goto 400
```

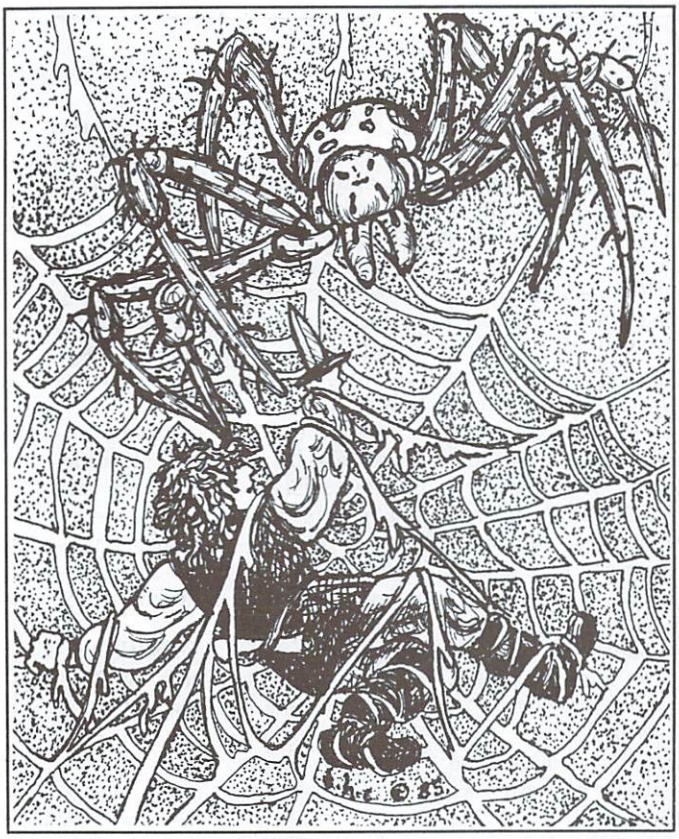
The movement section begins at line 1200. The verb numbers, which range from 1 to 8, are converted to the range 0 to 3. Thus, N (V = 1) and NOR (V = 2) both end up as 0. Then, the connection array for room *R* is read to see if a room lies in that direction. If so, line 1210 sets *R* to that room and loops back to display the details.

```
1400 print"you are carrying:
1410 z=0:for i=1 to n9
1420 if l%(i)=-1 then print"
     <2 spaces>"o$(i):z=z+1
1430 next
1440 if z=0 then print"<2 spaces>noth
     ing
1450 goto 400
```

Line 1400 is the beginning of the INV section. Line 1420 examines the *L%* array for objects with a location value of -1 (this means they're being carried). It then prints out their descriptions from the *O$* array. The variable *Z* is used here (and throughout the adventure) to keep track of the number of objects being carried.

```
1600 gosub 1610:goto 400
1610 j=0:for i=1 to n9
1620 if l%(i)=6 and a%(i)=3 then j=j+
     10
1630 next
1640 print"you have scored"j"points o
     ut of 20."
1650 if j<20 then return
1660 print:print"well done!":end
```

The verb SCO branches to line 1600. It starts with a **GOSUB** because the subsequent lines are also used by the 'quit' routine. Line 1620 scans the *L%* and *A%* arrays for objects in room 6



that are treasures (attribute 3). Every treasure is worth 10 points. In your own adventure, replace the 6 with the number of the room you want the treasures left in. You can also change the 10 if you want a different treasure value. It is also possible to give different treasures different values by defining additional attribute numbers. For example, objects with an attribute of 4 could be worth 20 points.

Lines 1640 and 1650 refer to the maximum score attainable. Substitute your maximum for the 20.

```
1800 if n=0 then 200
1810 gosub 930:if nx then 400
1820 if n=6 and l%(4)=0 then l%(4)=r:
     print"you find a key.":goto 400
1830 if n=12 and l%(10)=0 then l%(10)
     =r:print"something's there.":
     goto 400
1840 if n=8 then print"it is "mid$("u
     nlit",ca*2+1)".":goto 400
1850 if n=9 then print"it's rusty.":
     goto 400
1995 print"it's just "o$(n)".":goto 4
     00
```

Both EXA and LOO branch to this routine. If there is no recognized noun, line 1800 loops back to display the details of the current room. If there is a noun, line 1810 checks to see that it is actually present in the room. This is written as a subroutine because several other verbs will also need to check this.

```
930 nx=0:i=l%(n)
940 if i<>r and i<>-1 then print"i do
    n't see it here.":nx=1
950 return
```

Line 1820 responds to an examination of the pews by revealing the key (unless it's been previously revealed). Line 1830 does the same with the desk and the matches. Line 1840 says whether the candle is lit (CA = 1) or unlit (CA = 0), and line 1850 says that the shovel is rusty. Any item not specifically dealt with falls through to line 1995.

I recommend you include details for most of your objects in this routine. Trivial details are a good way of enriching an adventure, and they also make genuine hints less obvious.

```
2200 if n=0 then 900
2210 if l%(n)=-1 then print"you've al
     ready got it!":goto 400
2220 gosub 930:if nx then 400
2230 if a%(n)=0 then print"don't be s
     illy.":goto 400
2240 if a%(n)=1 then print"it's too h
     eavy.":goto 400
2250 if z>3 then print"your hands are
     <space>full.":goto 400
2260 z=z+1:l%(n)=-1
2270 print"oK":goto 400
```

Line 2200 is the beginning of the section that handles the verb GET (and the redundant verbs TAK and PIC). GET always requires a recognized noun, so this is checked at the start. Since several other verbs will also require this, the routine is written as a subroutine.

```
900 if n$="" then print"you need to s
    ay what to "v$".":goto 400
910 print"i don't know what a "n$" is
    .":goto 400
```

Line 2210 checks the location array to see if the specified object is already being carried; and line 2220 checks to see whether the object is actually present in the room. The next two lines check the object's attributes, rejecting it if it's scenery or immovable. Line 2250 checks the variable *Z*, rejecting the command if four objects are already being carried. Finally, at line 2260, the object is picked up (by changing its location from *R* to -1).

```
2400 if n=0 then 900
2410 if l%(n)>-1 then print"you have
     <space>no "n$".":goto 400
2420 l%(n)=r:z=z-1
2430 if n=8 then ca=0
2440 print"oK":goto 400
```

This is the branch for the verbs DRO and PUT. First, line 2400 checks that the noun is recognized. Line 2410 then checks the location array to see whether or not the object referred to is actually being carried. Line 2420 drops the object, by changing its location from -1 to *R*. Line 2430 is an example of an extraneous line inserted to achieve a special effect — it turns off the candle when it is dropped.

```
2600 if n=0 then 900
2610 gosub 930:if nx then 400
2620 if n=3 then print"it says: leave
     <space>treasure here.":goto 400
2700 print"there's no writing.":goto
     <space>400
```

Line 2600 is the beginning of the REA section. The verb requires a noun, and the object referred to must be present. These are checked in the first two lines. Most objects have no writing, and they fall through to line 2700. The sign, however, has writing, and it's dealt with in line 2620.

```
2800 if n=0 then 900
2810 gosub 930:if nx then 400
2820 if n=12 then print"try 'examine'
```

```
    ":goto 400
2830 if n<>13 then 2900
2840 if r%(4,1)=5 then print"it's alr
    eady open.":goto 400
2850 if l%(4)>-1 then print"you don't
    <space>have the key.":goto 400
2860 print"ok":r%(4,1)=5:goto 400
2900 print"that's not necessary.":
    goto 400
```

The verb OPE branches to line 2800. Like several other verbs, it needs a noun, and the noun must refer to an object in the room. The first two lines check this. Most objects do not need to be opened, and they fall through to line 2900. The safe is handled by lines 2840-2860. Line 2840 checks to see whether the bank (room 4) already has a south connection. Line 2850 checks to see whether the player is carrying the key (object 4). Line 2860 opens the safe by making the connection to room 5.

Line 2820 responds to the command OPEN DESK. It is there simply to be helpful. I recommend that you put many such helpful responses in your adventure. They are just like redundant verbs: they make it easier for players to express themselves and thereby reduce frustration. Adventure designers should try to anticipate the commands players might give and channel them into understandable expressions.

```
3000 if l%(9)>-1 then print"you have
    <space>no shovel.":goto 400
3010 if r<>1 then print"you can't dig
    <space>inside buildings.":goto 4
    00
3020 print"you find ";
3030 if l%(11) then print"nothing.":
    goto 400
3040 l%(11)=r:print"gold!":goto 400
```

The DIG routine starts at line 3000. The first line checks for the shovel, while the second line checks that the player is on the village green. If the gold has already been found, line 3030 is executed; otherwise, the gold is revealed.

```
3200 if n=0 then 900
3210 gosub 930:if nx then 400
3220 if l%(10)>-1 then print"you have
    <space>no matches.":goto 400
3230 if n=10 then print"it burns brie
    fly.":goto 400
3240 if n=8 then print"it burns brigh
    tly.":ca=1:goto 400
3300 print"it doesn't burn.":goto 400
```

Line 3200 is the beginning of the LIG section. The first two lines check whether the noun is recognized and whether it is present in the room. Since matches are necessary for burning, line 3220 checks this. A match is lit on line 3230 and the candle is lit on line 3240; the rest of the objects fall through to line 3300. No special comment is made when someone lights an already lit candle, but this could easily be added by checking *CA* before executing line 3240.

```
3400 gosub 1610:end
```

The verb QUI branches to 3400. There it prints out the score and ends the program.

My sample adventure is now complete. It's only 4K long and it's not very thrilling, as a game: however, it should provide you with a sound skeleton upon which to build your own adventure. Simply rip out the bits that do not apply to you, and fill in your own details.

Adding frills . . .

In order to make things easy, I have left the frills out of my sample adventure. There are no instructions, no title page, no colours, no fancy screen layout, and no pictures. You will probably want to put some of these things in your adventure, but you should be able to add them to my skeleton without too much trouble.

A title page and instructions are a good idea. Not only do they set the stage for the adventure, but they also provide something to read while the arrays are being set up. Instructions are particularly important for inexperienced adventurers, who may not be aware of words like INVENTORY.

A good screen layout is also desirable. Each new room provides several items of information, and it's helpful if this information is structured in some way. I like to use colours and reverse field: for example, if colour control characters are inserted at lines 220, 240 and 310, the room descriptions, objects and exits will be displayed in different colours. Similarly, the WHAT NOW? prompt can be put in a reverse-field box to set it off from the rest of the screen.

As you develop your adventure, keep in mind that it's supposed to be entertaining. Fill the adventure with interesting places and interesting tasks. Put in some humour and some mystery. Add a touch of the bizarre. Certainly there should be puzzles in your adventure, but they should not be too difficult. The purpose of puzzles is to give players the satisfaction of solving them.

Think of your adventure as a novel: you provide the setting, the plot and the supporting characters. The player takes the role of the main character. Since he doesn't know what's supposed to happen, it's up to you to channel his activities until he completes the adventure. If the player gets stuck on one of your puzzles, you have failed.

Testing your adventure

When your adventure is complete, it is essential that you test it. This means getting people to play it, while you sit silently taking notes. Such testing is the only way to determine whether the adventure is playable or not. What's more, seeing how people react to the adventure will enable you to fine-tune it.

At the lowest level, testing will uncover syntax errors and other minor bugs. No matter how many times you run through the program yourself, another person is bound to hit something you've missed. Moreover, sometimes your tester will input unexpected verb-noun combinations and produce unforeseen results.

A more important function of testing is to determine whether your tester moves smoothly through the adventure. If he has trouble finding words the program understands, make a note of the expressions he uses; and later try to incorporate them into your verb list. Similarly, if he gets stuck on one of the puzzles, think of a way to make it easier. It doesn't matter how absurdly simple the adventure seems to you; if your tester has problems, you must make changes.

The most important function of testing is to give you new ideas. As you sit watching someone play your adventure, you'll see him attempting things you never thought of. Most of these are unimportant, but occasionally you'll see something worth adding to the adventure. It may be just a sarcastic reply to a particular input, or it could be a major modification involving five new rooms and eight new objects. Whatever it is, take the opportunity to expand and enrich your adventure. When your adventure has been tested and you have finished the resultant fine tuning, it will be ready for release. I hope you will put a copy in the TPUG library, so that we can all enjoy it. □

Adventure And Other Adventures

by Jim Butterfield

Adventures are good for you. They increase literacy and exercise your mind. They can be fun to create, and fun to solve. And (don't quote me on this) they're fun to cheat on, too — sneaking the answers out of the program when you can't figure out what to do next.

It's worth making comments on the 'sneaking' part. When I'm asked, "How do you get past the snake?" or some other question spawned from utter frustration, I tend to have a standard reply: "Ask me again, and I'll tell you the answer... but first, think and be sure you really *want* it". It's annoying to be told something that you would have thought of yourself... any moment now.

Similarly, in anticipation of clever adventurers listing the program and browsing through my files, I carefully encrypted the command list. This is particularly Machiavellian on my part, I think. An experienced programmer can find out what might happen, but still not how to make it happen.

I'm talking, of course, about the original **Adventure**, which was written by Crowther and Woods. It grew — in an organic manner — at M.I.T., and quickly spread to all Digital Equipment Corporation (DEC) computer systems, worldwide. Many big computer users didn't even know they had it, but file **advent** was there, all right. It took up a huge amount of memory. The architecture of large systems allowed such programs to be restricted to quiet times, so that the command **HOURS** would list — from the system log — the non-priority hours during which **Adventure** could be played.

When micros started to gain attention, **Adventure** became a common case in point for computer users. Many users said: "These small computers are all very well, but I can tell you one program that will never fit..." — meaning the original **Adventure**. And when 8K machines expanded to 16K, and then 32K, I decided I needed to take a shot at it. I viewed the game as a 'classic'; I didn't want to change it or put in my own cute things. For memory and run time considerations, I abbreviated a few small parts of the game.

Let me tell you a little about **Adventure**, if you've never been there. You

begin by finding yourself at the end of a road before a small brick building. A stream runs out of the building. From that point on, you're on your own — except for a rumour that somewhere nearby is supposed to be a Colossal Cave, filled with fabulous wealth.

When you find the cave, you're confronted by dozens of events. The cave is populated by dwarves, a troll, a dragon, a giant clam, a tiny bird, a shadowy figure off in the distance, something that rustles as it follows you, a little plant that whispers 'water', a fierce green snake, and a bear. Some of the treasures seem to have special problems: a delicate Ming vase can be picked up, but breaks when it's dropped; a gold nugget makes the exit stairway vanish if you pick it up; and a platinum pyramid is not only invisible in a dark room, but won't fit through the tiny exit hole. But — as is often said — there are no problems, there are only challenges.

Even in the early days, **Adventure** inspired a host of emulators. TPUG President Michael Bonnycastle wrote **Caves** and **Stygian Tombs**, so as to devise an interesting adventure that would fit within 8K of memory. As a response, I wrote an 'array' adventure called **Explore** to demonstrate how to fit the maximum number of rooms within limited memory.

As systems grew in memory size and general capability, even the classic **Adventure** was overtaken by successors. The **Zork** series allows much more flexible syntax: where **Adventure** permits only two words, a verb and a noun, **Zork** will deal with a sentence such as: "Put the cake on the table".

A few words of caution: it's very easy to make adventure writing an ego trip: putting in clues that nobody will be able to solve if they aren't 'in'. Players tire very quickly of a game if they feel that it's not a challenge, just an exercise to show how clever the writer is. On the other hand, an adventure shouldn't be a dull recital. Nothing is more boring than proceeding along and killing everything in sight. Maybe your strength will give out, or maybe you'll make it through, but there's little exercise for the mind, here.

I recall having a discussion with an adventure-writer about non-violent adventures. The idea was that you couldn't get killed or hurt, and couldn't lose the game. The thought was laudable, but the game was a little dull; if you knew you couldn't fail, you didn't have the same motivation. In contrast, a preschool game called **Thirsty Nellan** (on disk (C)T7) offers challenge and a good interest level.

Educators have told me that adventure games make a major contribution to reading skills. Kids who are almost illiterate will learn to read better in order to understand the adventure situations... and will learn to spell better.

After all, if you want to take the necklace and type TAKE NECKLISS, the computer won't give you the booty... □

Adventure Games: A Personal Journey

by Peter Archer

You're on an old path made by horses... You're in a dense dark forest. To the south there seems to be light...

So begins **Castlemaze**, the first in Steven Darnold's adventure game trilogy. These words deserve to become just as famous as the opening quote from the original **Adventure**, written way back in the mid-seventies on a mainframe computer, and translated since then onto many microcomputers (including an excellent version for Commodore machines by Jim Butterfield, which will work on the C-64, all 32K PETs, and even the VIC-20 with 24K expansion).

Having played both Steven's series and the 'original', I can say that while I did enjoy **Adventure** very much, I liked **Castlemaze** and its successors, **Tunnelmaze** and **Valleymaze**, even better.

Steven Darnold first encountered adventure games around 1980, soon after he acquired his first computer, a 16K PET. This machine (when expanded to 32K) allowed him to become acquainted with Scott Adams' first two adventures (**Adventureland** and **Pirate Adventure**), plus (when he obtained a disk drive) Butterfield's version of **Adventure**. These all had a definite influence on him, when he later started producing his own adventures.

After joining TPUG, Steven naturally tried out all of the adventure games available from the TPUG library. He soon became acquainted with the efforts of Greg Hassett. Hassett had written several adventures for the TRS-80, and Commodore translations of some of these were available from the library. Among these were **Trip to Atlantis** and **Sorcerer's Castle**.

Steven indulged in a "little bit of reworking" and turned **Trip to Atlantis** into his own **Atlantis Adventure**, and **Sorcerer's Castle** into **Castlemaze**. He also used Hassett's basic structure to produce a major work of his own — **Tunnelmaze**, a sequel to **Castlemaze**, and intended to be the second part of a trilogy.

Atlantis Adventure

My own first encounter with a proper adventure was with Steven's **Atlantis Adventure**. This was in mid-1982, in the pre-Commodore 64 days, when my computer was a VIC 20 with 16K expansion. Steven's computer was his trusty 32K PET. The only people who had played Steven's adventures at this time were his wife, a few local PET owners, and some high school students at the school where he taught part-time.

Steven made a few changes to **Atlantis Adventure** to allow it to run on a VIC 20. About the only thing that needed to be altered was his input routine, which was machine specific. Changing a few pokes, however, was all that was required, to write a working VIC version. Of course, the screen looked pretty funny with the text (designed for the PET's 40-column screen) wrapping around the VIC's twenty-two columns. But after a while I hardly noticed this.

And what did I think of **Atlantis Adventure**? Well, I was hooked! So were my wife and eldest son (then aged 11). What better way to spend a Saturday evening than gathered around the computer over a few beers, enjoying the challenge of trying to unravel the puzzles set for us by our friend Steven?

Of course, when I first played **Atlantis Adventure**, I was a mere novice. But I found that there was just the right degree of difficulty — hard enough, without being too hard. As Steven says: "Any idiot can write an adventure that's so hard, no-one can solve it... The real challenge to the adventure writer is to make things just difficult enough without being unfair on the player." I found that **Atlantis Adventure**, as well as his other games, contained just that correct degree of difficulty — enough that the player has to really stretch his mind, but by no means unsolvable.

Looking back on **Atlantis Adventure**, I now realize that there are some very nice original ideas. Things like the cannon and the jail stick in the mind, but also little things like the comb and the relationship between the train and the cliff show evidence of a very inventive mind. And I mustn't forget the good old discus...

Castlemaze and Tunnelmaze

Once we had finished **Atlantis Adventure**, we tried **Castlemaze**, and then **Tunnelmaze**. We found that they, too, contained just the correct degree of difficulty.

I believe that the narrow crack and large painting will become a classic example of adventure programming. The vase, as well; and the... oops, I mustn't give the game away to those of you who have yet to experience the pleasure of **Castlemaze** for yourselves. Also, I will never forget the snake (or the turtle) in **Tunnelmaze**.

Incidentally, a word of advice for all you would-be adventurers when you tackle any of Steven's masterpieces. Always examine *everything*, and note very carefully, and think hard about the implications of, even the *slightest* remark made about the nature or position of every object. On several occasions that I can recall, I have missed a vital but very subtle clue, and hence blundered around blindly for much longer than necessary.

The C-64 Hits New Zealand

Once the C-64 became available in New Zealand in early 1983, we abandoned the old VIC for adventuring and switched wholeheartedly to the C-64. Steven and I obtained two of the very first C-64s sold in New Zealand (both very low serial number models, from the West German Commodore factory, air-freighted direct to New Zealand from England before they were available from the normal channels via Australia).

Steven set to work to produce very nice C-64 versions of **Atlantis Adventure**, **Castlemaze** and **Tunnelmaze** with very effective screen colours, and so on. The latest versions of these are compiled with the DTL compiler, and run very fast on the C-64. Copies are in the TPUG library (and in some U.S. groups' libraries).

These adventures helped to fill two disks of public domain programs that Steven put together for the C-64 in early 1983, when C-64 software was scarce. The programs have since become very well known here in New Zealand, and also in some parts of the USA.

In The Public Domain...

At this point I feel that a word (or three) about the subject of public domain adventure game quality would be in order.

I have seen many so-called adventures being sold commercially

that are vastly inferior to the public domain ones I have been speaking about so enthusiastically.

One of the reasons Steven decided not to sell **Tunnelmaze** commercially was because the leading publisher of software for hobby computers in New Zealand was only interested in adventures if they 'contained graphics'. Text-only adventures were not wanted, no matter how superior the quality. A very short-sighted policy indeed, considering that the two so-called 'graphics adventures' that were subsequently released onto the New Zealand market by that same firm were pure garbage!

The other reason that **Tunnelmaze** never appeared as a commercial product was that Steven had sent an early (PET) copy to an Englishman, with whom he used to correspond. Some considerable time later, this person published a book about adventure games for the C-64 that included a full printed listing of **Tunnelmaze**. Steven was not very pleased (to say the least) about this, and decided forthwith to place **Tunnelmaze** in the public domain.

Valleymaze

Steven had intended from the first that **Castlemaze** and **Tunnelmaze** would be the lead-up to his real masterpiece, which he had named **Valleymaze**. But the sheer size of the project had caused him to defer starting work on it.

After thinking about it for a long time, however, and after much urging, he finally made a serious start on it in the (southern) summer of 1983-84. (Steven's job as a part-time high-school teacher gives him a long summer vacation, from mid-December until the beginning of February).

To start with, he used his trusty old PET, with its well-used package of toolkit utilities, and so on. But **Valleymaze** was intended to be as big as the C-64's memory would allow. Eventually he ran out of room in 'Old Faithful', and had to proceed with actually writing it on the C-64.

He soon found that the sheer size of **Valleymaze** was a problem. To give himself as much room as possible, he rewrote his parsing routine in machine language and placed it in the C-64's 'C block'. This also greatly increased the speed and eliminated the 'garbage collection' problem. He also rewrote his input routine in machine language and put it into the cassette buffer. When the finished program was compiled with the DTL compiler, this also helped the run speed.

Talking of size, **Valleymaze** is actually bigger than **Adventure**. It has more locations (or rooms), but by keeping the text much less wordy, Steven has still (just!) managed to cram the entire program into the 38-odd K of the C-64's BASIC memory with no need to resort to disk files. Hence **Valleymaze** is available on tape, as well as on disk (it must surely be just about the largest adventure available for any computer on tape).

But what is it like? Well, I had to wait some time to find out. After his unfortunate experience with the **Tunnelmaze** 'piracy', Steven was understandably reluctant to trust anyone (even me!) with a copy of **Valleymaze**. So, as we then lived over four hundred miles apart, I had to wait until we could get together to actually try it out. Steven likes to have someone test-play his new adventures while he watches them, noting any possible difficulties for later correction in the final version. In this way, he can accurately gauge the degree of difficulty of the various problems that the player has to solve.

Steven kept **Valleymaze** under wraps for almost a year before eventually deciding to place it in the public domain too. By the time you see this issue, **Valleymaze** will be officially in the public domain (for the Commodore-64 only, at this stage, because of memory limitations) *(see 'Library Additions')*.

Steven's long-suffering wife, Helen, had been (as usual) his first 'guinea-pig', and he had already polished up **Valleymaze** to a very high standard before my son Jonathan and I finally had a chance to play it in Steven's motel, in Christchurch, August 1984.

Well, what can I say? Many adjectives come to mind. Large. In fact, huge! It certainly merits the inclusion of the 'save game' option that he had thought unwarranted in his earlier efforts. Another adjective — brilliant! There are many nice touches completely original to him. Like the shop. And the way the train fits in. And the oven...

But don't take my word for it! Try it for yourself. I would advise that you play **Castlemaze** first. Then tackle **Tunnelmaze**. And finally take on **Valleymaze**. You will have countless hours of enjoyment (and frustration!) ahead of you. □

Adventure Games: Text versus Graphics

by Peter Archer

There are basically two types of computer adventure games: the straight 'text only' adventures, and the ones with graphics. The 'text only' adventures are (in my opinion) the true classic computer adventures. Playing a top-class text adventure is very similar to reading a quality work of literature: your mind is free to construct your own pictures of the scenes described, whereas in a graphics-type adventure, what you see is what you see.

One problem nowadays with graphics adventures is that high-quality graphics consume large amounts of memory. This position will eventually change. The next generation of home computers will have much more memory than today's machines, based as they are on 8-bit processors, with their somewhat limited memory addressing capacity. Also in the not too distant future, the combination of the video-disk and the home computer will allow the use of genuine hi-res 'real live scenes' in computer games — as is already done on the more advanced coin-operated arcade machines.

At present, however, there is no comparison between the two types of adventure games. As long as the player is endowed with a good imagination, a well-written text adventure wins hands down every time over all but the very best graphics adventures. □

Inside *Inner Space*

by Jim Butterfield

It was December 31, 1979. A day like any other day... except for the arrival of *The Transactor*, Volume 2, Number 7. The front page announced: "This month's *Transactor* is a collection of ... charts and tables concerning PET..."

There were 21 pages of memory maps, edge connector tables, hexadecimal conversion charts and other useful stuff. Just about half of it was my stuff (much of it reprinted from previous issues). It was useful to have everything in one place. I carried around my copy for a couple of years, until...

Volume 4, Issue 5, appeared around October 1983, although the issue carries no date. The reference section now covered 54 pages, and the type was smaller so that more information could be jammed in. There was more professional typesetting and numerous diagrams: for example, there were pictures of the edge connectors. By this time, the VIC 20 and Commodore 64 were popular, and there were many special charts covering sound, graphics and sprites. The original 'Superchart', showing character representations of various styles of memory (screen, ASCII, etc.), was now split into two charts: one for PET/CBM, and one for VIC/C-64. The B series wasn't yet available, but maps, command summaries and error lists were provided for this range of machines. New tables such as a side-by-side comparison of disk drives (still valid) were included, as well as, surprisingly, a book list and a glossary.

This *Special Reference Issue* became a hot item, and quickly sold out. It could have been (should have been) reprinted, but the editor, Karl Hildon, was busy gathering new material for his next magnum opus ...

The Complete Commodore Inner Space Anthology is now available for $14.95 from *The Transactor*, 500 Steeles Avenue, Milton, Ontario, Canada L9T 9Z9. The price, by the way, is an introductory special: newsstand and bookstore price will be somewhat higher.

If you have any technical interests at all — and maybe even if you don't — you need this one. There will certainly be something you'll need to look up... and almost anything you can think of is in the *CCISA*. If you're not a techie, you can still find a comprehensive set of BASIC commands. If you're beyond BASIC, you'll find extensive material on COMAL and Machine Language. If you don't program at all, you'll be able to use the Word-processing Reference Guide, which has a side-by-side comparison of the commands of **Superscript**, **EasyScript**, **Paperclip**, **Speedscript**, **WordPro** and the new **WordPro 64**. If you'd rather work with spread sheet programs, a briefer section gives a summary of the **Calc Result** commands.

There's even more in terms of general information. Bulletin board numbers and network numbers (for Datapac, Compuserve, Tymnet and GTE) are given for the communications enthusiast. And for everyone — including the traveller — there's a list of computer clubs.

This is a book without text. It's all reference tables. Because of this, it's less a book for learning and more for checking up on the details of something that you know, but can't remember. But even if you can't understand some of the tables, they're interesting to look at and may draw your attention to new areas of computer utilization, and new ideas.

There are 122 pages of information in *The Complete Commodore Inner Space Anthology*, and on some pages the print is fairly small. One of the objectives is to allow you to see a complete section of information laid out in two adjacent pages. It's spiral-bound, by the way, to allow it to lie flat for this type of use. Complete memory maps — RAM and ROM — are available on such double pages, and other sections show careful planning to give the same useful effect.

It's difficult to summarize the contents of this volume. There are extensive charts to assist and give inner details on BASIC, COMAL and Machine Language. The Supercharts give character sets; the hardware section gives everything from board layouts to chip specifications. The printer section gives details on many printers; the disk drives section supplies both technical and general information on Commodore disks. This includes detailed memory maps for the 4040, 8050 and 1541 units.

The video section gives detail on video for all current machines including the B series. Oddly, Greek alphabet character matrix codes are given for printers, but not for screen characters... it's surprising to see them at all, but that's one of many little surprises in the book.

Karl seems to have gotten enthusiastic over sound generation. Not only does the book contain details on note generation and sound chip registers — as usual, for all computers — but it also contains a glossary of music notation symbols and chord composition. I'd half expected to see a catalogue of favourite weapon sounds — laser blaster, phasers, gunshots, explosions, sirens, and so on — plus an ADSR list for common instruments such as piano, flute and oboe... but these are absent.

There are many unexpected items. 'Checking Semiconductors with an Ohmmeter' is a surprise, although it's interesting to see the wide range of technologies listed. Metric units are there. I wish it would tell me, though, how to pronounce the 'g' in 'gigahertz': hard as in 'gift', or soft as in 'gin'. The table of propeller thrust for a power boat makes me suspect that Karl has nautical leanings; and you may find it useful to know that the US septendecillion is the same as the British nonillion... either way, it's a big number.

The unit conversion table is lengthy. If you want to change firkins to liters, or roods to hectares, you'll find it all there. I had some trouble reading (twice) that 1 cubic foot per minute equals .0011 quarts (liquid), but maybe that somehow reflects the editor's drinking capacity. And I was frustrated in trying to convert Atmospheres to Pascals; although they are both units of pressure, the conversion isn't readily supplied.

The Complete Commodore Inner Space Anthology ends with a couple of lavish tables that don't seem to have much to do with day-to-day computing: Geometric Areas and Volumes, and an ornate Periodic Table of the Elements. I asked the editor about this and, as I understand it, they were just too pretty to be left out. In any case, you never know when you might need one.

It's a rich and useful collection of information. You'll just have to have one for your bookshelf. □

The Ultimate Shuffle

by Gordon Campbell

Shuffling a deck of cards should be a fast, simple process for a computer. But many programs spend excessive time doing it.

The best algorithm is fairly simple, although the mathematical proof that it generates a well-shuffled deck is not obvious. The following steps are required:

- Consider a deck of 52 cards.
- Select a card at random, and switch it with the top card. (If the top card was selected, leave it alone.)
- Reduce by one the number of cards being considered.
- Repeat steps two and three until only one card is left.

Simple enough so far, and this algorithm can be coded in one line of BASIC. The tough part is making it run quickly. The demonstration program below will run on all Commodore computers.

The 'tricks' used to speed things up are:

- Move numbers, not strings. (The numbers don't really move in memory, but the strings do.)
- Use as few lines as possible.
- Include no spaces in the code.
- Use variables, not literals.
- Shorten zero to period. (The BASIC interpreter sees it as 0.0, and does it quickly.)
- Place the subroutine as close to the front of the program as possible.
- Predefine the variables, with the first one mentioned first, etc.
- Make best use of single-digit line numbers.

As is obvious from the demonstration program, I do not advocate these measures for all of a program, just the parts where speed is important.

Demonstration Program

In the demonstration program, line 6 is the routine that does the shuffling. Everything else is either initialization — up to printing the 'menu' of options — or window dressing to check the shuffle subroutine.

One of the menu options is to shuffle a deck twenty times and report how long it took, while the other displays the shuffled deck as seven hands of seven cards each. This is just to prove that typical hands are being produced.

Notice that the array *D$* contains the visual representation of the cards, but is untouched by the shuffle subroutine. The numeric array *D* contains pointers into the *D$* array.

Application

Several years ago, one of the players in my poker circle introduced a new variation of the game. The variation lent itself extremely well to computerized analysis, so I wrote a program to do just this, accumulating statistics on winning and second-best hands. (As every poker player knows, the only really expensive hands are the ones that wind up second best.)

Within a week, I had accumulated statistics on twenty thousand deals. Ethics decreed that I show the statistics to the other players, but the game still proved profitable.

The Bottom Line

The demonstration program shuffles a deck of cards in well under one second. This isn't quite instantaneous, but it certainly shows that there is no need for programs to print the message, *Please wait — shuffling.*

```
1 goto16
6 fora=ctodstepe:b=rnd(.)*a:f
  =d(a):d(a)=d(b):d(b)=f:next
  :return
11 rem initialization
16 print "ultimate shuffle?
21 print"by gord campbell
26 print
31 a=.: b=.: f=.
36 e=-1: c=51: d=1
41 dim d(51), d$(51)
46 for j=1 to 51: d(j)=j:
  next
51 for j=0 to 51: read d$(j):
  <space>next
56 print "press 't' for timin
  g test
61 print "<6 spaces>'d' for d
  isplay
62 print "<6 spaces>'q' to qu
  it
66 geta$: ifa$="t" goto 81
71 if a$="d" goto 101
73 if a$="q" then end
76 goto 66
81 n=20: t=ti
86 t=ti:forj=1ton:gosub6:next
  :x=ti
91 print "tooK"; (x-t)/(n*60)
  ; "seconds per deal"
96 goto 56
101 gosub 6: for j=0 to 6:
  for K=0 to 6
106 print d$ (d (j*7+K)); " "
  ;: next
111 print: print: next: goto
  <space>56
116 data "aX", "2X", "3X", "4
  X", "5X", "6X", "7X"
121 data "8X", "9X", "10X", "
  jX", "qX", "KX"
126 data "aZ", "2Z", "3Z", "4
  Z", "5Z", "6Z", "7Z"
131 data "8Z", "9Z", "10Z", "
  jZ", "qZ", "KZ"
136 data "aS", "2S", "3S", "4
  S", "5S", "6S", "7S"
141 data "8S", "9S", "10S", "
  jS", "qS", "KS"
146 data "aA", "2A", "3A", "4
  A", "5A", "6A", "7A"
151 data "8A", "9A", "10A", "
  jA", "qA", "KA"
```

The C-128: A Programmer's Playground

by Nick Sullivan

The sleek new Commodore 128 PC, with its Concorde profile and rococo architecture, is one of five new computers that Commodore is planning to introduce this year (the others: the IBM clone PC 10, a Unix micro, the LCD lap computer, and the Amiga). With the possible exception of the Amiga (still, apparently, at least six months from release), it is also the one that has generated the most excitement in the Commodore computing community.

The hallmarks of the 128 are versatility and compatibility. In one mode, it is a Commodore 64, ostensibly 100 per cent compatible (though one well-known user has reportedly found a commercial C-64 program that *won't* run on his prototype machine). In another mode, it is a CP/M computer, with very fast disk access, and a mountain of popular software readily available. Documentation for this mode is not yet included in preliminary versions of the C-128 manual. Both C-64 and CP/M mode will likely appeal to users, rather than programmers.

Then there is the 'native' or 'C-128' mode, with 128K of built-in bank-switched RAM and 44K of ROM, a sophisticated new BASIC (numbered 7.0), 40 and 80 column video output, all the C-64's graphics and sound capabilities, a lot of new keys, and the ability to communicate with the fast new Commodore disk drive, the 1571. C-128 mode is not 100 per cent compatible with any other computer, but the BASIC 7.0 owes a lot to the BASIC 3.5 dialect in the Plus/4 and C-16. No doubt there will be lots of great software written for this mode, too, in the months and years to come. Yet it is also ideal for those who like to do their own programming.

The overlooked virtue of the Plus/4 and C-16, as Jim Butterfield likes to point out, is that they are friendly to programmers. First there's the big vocabulary of BASIC commands, which makes most pokes obsolete. Next is the addition of three commands — **RENUMBER**, **DELETE** and **AUTO** — specifically for making program development more convenient. Lastly come the improvements to the already excellent Commodore screen editor, which now features a variety of useful two-stroke commands (the **ESC** key followed by a character) to perform such functions as inserting and deleting screen lines, erasing lines to and from the cursor, defining screen windows, and blanking the screen from the cursor to the bottom. You can even get an insert mode, such as one finds on word processors, for typing without overstriking.

The friendliness of the Plus/4 and C-16 carries over to the Commodore 128, and is enhanced with such items as a numeric keypad, a **CAPS LOCK** key and a **TAB** key. But this machine boasts other features that programmers will also cherish.

The Monitor

Like many earlier Commodore computers (all except the C-64 and VIC 20), the C-128 has a built-in machine language monitor. This monitor works with 5-character hexadecimal addresses, not the 4-character addresses most of us are used to. The first character is the bank number (0 through F). BASIC programs and graphics data are stored in Bank 0; variables and strings are stored in Bank 1. The segregation of program from data gives plenty of room for code and for big arrays, and also simplifies program chaining. Banks 2 through E are available for memory expansion (which will be available later this year) up to 512K. Bank F holds the ROMs. The lowest 16K of memory seems to be common to Bank 0 and Bank F; Bank 1 shares only the lowest 1K with the others. Bank switching is controlled at memory location $FF00 in each bank, and there are lots of little routines in pages 2 and 3 of memory for getting around between banks.

The monitor itself features a simple assembler and disassembler, memory display, hunt, transfer and compare functions, load and save, and so on. It is bilingual in hex and decimal. Since the monitor's command checking routine jumps through a RAM vector ($032E in our preproduction machine), it should be possible to add new monitor commands if required. By the way, other new vectors in RAM have also been provided, allowing for various other extensions to the operating system, particularly in the area of editing functions (the escape and control sequences, for example).

High Resolution Graphics

The Commodore 64 has excellent hi-res and medium-res graphics capability — in hardware. Unfortunately, it is a programming challenge to draw so much as a simple straight line on the hi-res screen, let alone anything more complex. Moreover, to accomplish hi-res tasks at reasonable speed normally requires either machine language, or a non-resident language such as Logo or COMAL.

This is not true of the C-128, which provides a battery of commands to simplify hi-res work. Now we have the **DRAW** command to generate points, lines or polygons, and the **BOX** and **CIRCLE** commands for more specialized tasks (though both of these are more general than their names suggest). In addition, you can now get a split text/graphics screen, with the graphics part either multicolour or hi-res,

BASIC 7.0 Keywords

Compiled by Chris Bennett

The BASIC 7.0 language on the new C-128 computer (in C-128 mode) is the richest BASIC dialect yet offered on a Commodore machine. On this page, Chris Bennett has compiled a list of BASIC 7.0 commands in three sections. The first section contains those BASIC 4.0 commands that were present on the PET/CBM series, but dropped for the VIC 20 and Commodore 64. In the second section are the commands that were new with BASIC 3.5 (on the Plus/4 and C-16); these commands have been retained in BASIC 7.0. The third section gives the commands that are new with BASIC 7.0. Commands that are found in all Commodore BASIC dialects (**PRINT**, **POKE**, and so on) are not included in this list.

BASIC 4.0 Commands

APPEND	Add new data to an existing file.
BACKUP	Duplicate an entire disk in a dual drive.
CATALOG	Display the disk directory on the screen.
COLLECT	Validate a disk.
CONCAT	Add one file to the end of another.
COPY	Copy files on disk.
DCLOSE	Close disk files.
DIRECTORY	Display the disk directory on the screen.
DLOAD	Load a program from disk.
DOPEN	Open a disk file.
DSAVE	Save a program onto disk.
HEADER	Format (or 'new') a disk.
RECORD	Position to any record in a relative file.
RENAME	Change the name of a disk file.
SCRATCH	Delete a file from the disk.

BASIC 3.5 Common Commands

AUTO	Generate line numbers automatically.
BOX	Draw a rectangle of any size.
CHAR	Display text at a given screen location.
CIRCLE	Draw circle, ellipse, arc, triangle or octagon.
COLOR	Set background, foreground, border colours.
DEC	Return decimal value of a hexadecimal string.
DELETE	Delete a range of lines.
DO	Start of a **DO-LOOP** command.
DRAW	Draw individual dots, lines and shapes.
ELSE	Part of **IF-THEN-ELSE** statement.
ERR$	Return the error description string after a **TRAP**
EXIT	Cause termination of a **DO-LOOP command**.
GRAPHIC	Set graphics mode: hi-res, split screen, etc.
GSHAPE	Display a saved shape to the screen.
HEX$	Return hex equivalent of decimal number.
HELP	Display line where a BASIC error occurred.
INSTR	Return the position of one string in another.
JOY	Return the position of either joystick.
KEY	Display function key definitions, or define new ones.
LOCATE	Position pixel cursor on the screen.
LOOP	Part of the **DO-LOOP** command.
MONITOR	Enter the machine language monitor.
PAINT	Fill an area with colour.
PUDEF	Redefine symbols in the **PRINT USING** command.
RCLR	Return colour of background, foreground, or border.
RDOT	Return the position of the pixel cursor.
RENUMBER	Renumber the lines of a BASIC program.
RESUME	Return to execution after **TRAP**ping an error.
RGR	Return current graphic mode value.
SCNCLR	Clear the current screen in all modes.
SOUND	Produce a sound using one of three voices.
SSHAPE	Save a screen shape.
TRAP	Intercept BASIC errors.
TROFF	Turn trace function off.
TRON	Begin trace, displaying line numbers executed
UNTIL	Part of **DO-LOOP** condition clause.
USING	Define format for **PRINT** statement.
VOL	Set the volume level for the sound command.
WHILE	Part of **DO-LOOP** condition clause.

C-128 New Commands

BANK	Set bank number for **PEEK**, **POKE**, etc.
BEGIN	Used in multi-line **IF** statement.
BEND	Part of the **BEGIN-BEND** construction.
BLOAD	Load file into a specific bank location.
BOOT	Load and run the program named.
BSAVE	Save memory from a specific bank location.
COLLISION	Used in sprite collision detection.
DCLEAR	Initialize disk drive.
DVERIFY	Verify file on disk with program in memory.
ENVELOPE	Set sound envelope (ADSR on SID chip).
FILTER	Set the sound filter (SID chip).
MOVESHAPE	Move a defined shape around the screen.
MOVSPR	Move one of the sprites around the screen.
PEN	Read the light pen port.
PLAY	Play music.
POT	Read potentiometer on joystick port.
RREG	Return register value.
RSPPOS	Return sprite position.
RSPRITE	Return sprite attribute.
RSPRCOLOR	Return colour of any given sprite.
RWINDOW	Return position of current window.
SLEEP	Delay execution for a specified interval.
SPRCOLOR	Set a sprite's colour.
SPRDEF	Define a sprite.
SPRITE	Set up the attributes of a sprite.
TEMPO	Set the tempo of the music.
WIDTH	Set width for the pulse wave form (SID chip).
WINDOW	Define a screen window.
XOR	Exclusive OR.

with a single line of BASIC.

A lot of thought has gone into the creation of these commands to make them as powerful as possible. The **CIRCLE** command, for instance, can draw, in addition to circles, part circles, ellipses of any eccentricity, and even polygons like triangles, rectangles and pentagons. Commands with that kind of power are going to make hi-res programming a lot less forbidding than on the 64.

Yet there is a penalty, too. The **CIRCLE** command takes no fewer than nine parameters: colour, centre (x), centre (y), radius (x), radius (y), beginning of arc (in degrees from vertical), end of arc, amount of rotation, and segment size (in degrees between vertices). Many of the parameters in this and other commands have sensible default values — it is rarely necessary to specify all nine. Even so, the conveniences offered by BASIC 7.0 do not come without a price in memorization.

The assignment of memory to the hi-res screen is interesting. At power-up, the start of BASIC address, where programs begin, is hex 1C01, decimal 7169. This leaves you with a little over 58,000 bytes of space for program text. As soon as you invoke a hi-res screen, though, with the **GRAPHICS** command, the start of BASIC is moved up to hex 4001, decimal 16385, and the intervening space is used for graphics data. This rearrangement, invisible to the user except for a momentary delay, automatically resolves the conflict between program text and graphics data that so frequently has to be dealt with manually on the 64. It also reduces program space by 9,000 bytes or so, but that will rarely be a problem.

Sprites

There are two main difficulties with sprite programming on the Commodore 64. One is designing the sprite, which requires either a sprite editor or a bucketful of tenacity; the other is handling all the poking and peeking needed to turn on sprites, to position and move them, to set their colours, sizes, priorities and graphics modes, and to detect collisions amongst themselves and with other graphics data.

The Commodore 128 programmer experiences neither difficulty. The first is solved with a built-in sprite editor, which one invokes with the command **SPRDEF**. This editor is not as powerful as some that have been written for the 64, but it certainly beats fiddling with graph paper, binary math and **DATA** statements. (It is unfortunate that Commodore has not provided a RAM vector for linking in extensions to this editor, at least in the prototype 128s we have seen.) Sprites are stored automatically in a dedicated area of memory. They can be easily saved to and loaded from disk with the **BSAVE** and **BLOAD** commands used for transferring binary data, or they can be defined as strings with another special command, **SPRSAV**.

The second difficulty mentioned above, that of manipulating a sprite already defined, is solved with a series of commands that allow all sprite parameters to be readily set. Collisions can be detected from BASIC (not possible on the 64), and can even be set to jump automatically to a specified subroutine when detected. (Sprite-sprite collisions can be detected separately from sprite-data collisions, and light pen interrupts may be detected also.) Sprites can even be set to move automatically with a specified speed and direction. Once set in motion, they continue about their business with no further attention from the programmer.

Sound

Programming sound effects and music on the Commodore 64 again requires an encyclopedic knowledge of the relevant i/o registers. Once you have mastered the mechanics of getting sound out of the machine, you can start turning your attention to figuring out parameter values that actually provide a decently musical result.

The C-128 again shows just how far short the 64 falls when it comes to programmer friendliness. As usual, no pokes are necessary. Musical notes can be entered using a simple code that is both concise and flexible. Voice number, note value, octave and so on, can all be set along with the notes in the **PLAY** command. The wave form and ADSR envelope, which give the sound most of its character, can be selected from ten preset instrumental sounds, or can be set by the programmer for a particular need.

More Commands

BASIC 7.0 boasts a vocabulary of more than 140 commands, of which I have mentioned only a few that will be of particular interest to 64 programmers. But it is hard to stop there. The structured **DO-LOOP** command-set ought to be mentioned, as should the convenient disk commands (including a transparent directory) that were present in Commodore's old BASIC 4.0 machines, but not in the VIC 20 or C-64. In fact, BASIC 7.0 has all the old BASIC 4.0 commands plus a couple more.

Other features that will make life easier or more interesting for programmers are the predefined (and redefinable) function keys, the numeric keypad, and the well-situated reset button. There are lots of things to explore and experiment with on this machine also. For instance, programs can be made to autoboot by placing the characters 'CBM', the filename and some other data on track 1, sector 0 of a disk in the drive at power-up. Programs can be moved back and forth between C-64 and C-128 modes by pressing the reset switch either with the logo key (selects 64 mode) or without it (selects 128 mode). The RAM used for the 64 mode is the 128 mode's bank zero.

A Programmer's Playground

The Commodore 128 is undoubtedly a fine *user's* machine, given its ability to make use of possibly more already existing software than any other computer in any market. But with its power, its complexity, and its convenience features, it is also going to be a source of aid and inspiration to programmers of all persuasions, and a continuing challenge to exploit and explore to the fullest. □

C-64 & C-128* OWNERS:

Stop playing "Hide-and-Seek" with your Disk software! Let your Commodore 64™ "memory" do all the work for you with **MASTERDISK©** or **MASTERDUAL©**. Experience fully-integrated power for maximum productivity. Easy Menu-Guides make everyone a "pro" in 1 day! Both feature fully-sorted Master Catalog of your TOTAL Software Library (even those unlistable "protected" disks).

MASTERDISK© works with 1 or 2 Single Disk Drives (1541, MSD, etc.) **PRICE: $34.95. (U.S.A.)**

MASTERDUAL© utilizes all dual-drive functions (4040, MSD, etc.) for maximum speed and ease of use. **PRICE: $29.95. (U.S.A.)**

*Both are also compatible with the NEW C-128 (in C-64 mode)!

INTEGRATED-SOFTWARE SYSTEMS
VISA/M.C./C.O.D., call **(515) 233-2992**
9 a.m.-9 p.m., (Central Time) Mon.-Fri.
Add $3 Shipping/plus $2 Foreign, C.O.D.
IA Residents add 4% Sales Tax.

IN CANADA:
Cricket Distribution Co.,
616 Mackechnie Crescent, Cobourg,
Ontario, Canada K9A 4X8
Call (416) 372-3692 for prices
and shipping information
9 a.m.-6 p.m. (Eastern Time) Mon.-Fri.

COMAL INFO
If you have COMAL— We have INFORMATION.

BOOKS:
- COMAL From A To Z, $6.95
- COMAL Workbook, $6.95
- Commodore 64 Graphics With COMAL, $14.95
- COMAL Handbook, $18.95
- Beginning COMAL, $22.95
- Structured Programming With COMAL, $26.95
- Foundations With COMAL, $19.95
- Cartridge Graphics and Sound, $9.95
- Captain COMAL Gets Organized, $19.95
- Graphics Primer, $19.95
- COMAL 2.0 Packages, $19.95
- Library of Functions and Procedures, $19.95

OTHER:
- COMAL TODAY subscription, 6 issues, $14.95
- COMAL 0.14, Cheatsheet Keyboard Overlay, $3.95
- COMAL Starter Kit (3 disks, 1 book), $29.95
- 19 Different COMAL Disks only $94.05
- Deluxe COMAL Cartridge Package, $128.95 (includes 2 books, 2 disks, and cartridge)

ORDER NOW:
Call TOLL-FREE: 1-800-356-5324 ext 1307 VISA or MasterCard ORDERS ONLY. Questions and Information must call our Info Line: 608-222-4432. All orders prepaid only—no C.O.D. Add $2 per book shipping. Send a SASE for FREE Info Package or send check or money order in US Dollars to:

COMAL USERS GROUP, U.S.A., LIMITED
5501 Groveland Ter., Madison, WI 53716

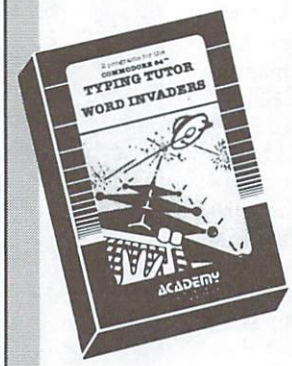
TRADEMARKS: Commodore 64 of Commodore Electronics Ltd.; Captain COMAL of COMAL Users Group, U.S.A., Ltd.

TRIVIA PLUS™

- Over 3,500 questions
- 8 fascinating categories
- No typing required
- Excellent graphics and sound (includes over 100 songs!)
- For 1 to 6 players
- In 'Trivia Plus' a good bluff may be the next best thing to the right answer

Commodore 64™ Disk $24.95

TYPING TUTOR + WORD INVADERS



REVIEWERS SAY:
"This is the best typing tutor we have seen yet;★★★★+" **INFO-64**

"Computer aided instruction at its best." **Commander**

"This is an excellent program that makes typing practice an enjoyable pastime instead of boring drudgery." **DILITHIUM PRESS**

Rated the BEST educational program for the VIC 20 **Creative Computing**

CUSTOMERS SAY:
". . . delighted with my son's progress . . . he is the only one in his second grade class who touch types at the computer."

"Your Typing Tutor is an excellent program . . . our 4 children literally wait in line to use it."

"Thoroughly satisfied, can't believe how fast I've learned to type. I've never typed before."

In daily use by schools across the USA.

NEW! Commodore Plus/4 or 16 . . . Tape $21.95 Disk $24.95
Commodore 64 . . . . . Tape $21.95 Disk $24.95
VIC-20 (unexpanded) . . . . . Tape $21.95

IFR (FLIGHT SIMULATOR)

REALISTIC AIRCRAFT RESPONSE

"Has a quality of realism which sets it apart from others, even those I've tested in flight school." **Compute's Gazette**

"Great program!" **INFO-64**

"It is tremendous fun." **Compute's Gazette**

"Flight tested by an air traffic controller, two skilled pilots and an elementary school class. Highly recommended by all." **Midnite Gazette**

"This is an unbelievably realistic simulation of the difficulties facing a pilot in instrument flying. I'm a 747 pilot and I think that this simulation could do a lot to improve the reactions and instrument scan habits of even very experienced pilots." **747 pilot**

NEW! Commodore Plus/4 or 16 . . . . Tape or Disk $29.95
Commodore 64 . . . . . Tape or Disk $29.95
VIC-20 (unexpanded) . . . . . Cartridge $39.95

ACADEMY SOFTWARE

Shipping and handling $1.00 per order. CA residents add 6% tax.

VISA MasterCard

P.O. Box 6277 San Rafael, CA 94903 (415) 499-0850

The Creation of *Sky Travel*

by Frank Covitz

*The **Sky Travel** program, reviewed elsewhere in this issue, was two years in the making. In this article, one of the authors of the program describes the creation and development of **Sky Travel**, and the problems faced by him and his co-authors in bringing the program to the market.*

It is 1982, and I am discussing with my colleague and good friend, Dr. A.C. Ashcraft (Clif to you and me), the marvels of the 6502 home computer, the PET (we had both graduated from the KIM). After spending a good deal of time and brain-busting, Clif and I had recently completed a very versatile music program that spat out 4-voice music with user-defined instruments through a Digital to Analog Converter (DAC). SID sounds are totally primitive compared to what our DAC software could do, although SID does have a better high frequency response. In any case, I suggested to Clif that another 'natural' for a computer application would be in astronomy, where computer graphics could conveniently represent the results of what we knew would be some rather complex computation.

Both Clif and I were (and still are) amateur astronomers. Some years back, another good friend (Keith Sproul) had obtained for us a star catalogue in computer-compatible form through his connections with ARPANET. So we had a good start (although later it turned out that much more time was spent on the programming effort than would have been spent keying in the star data from catalogues). After deciding that it should be possible to get the 6502 to present graphically a view of any part of the sky, I spent some time describing what could be done in the astronomy area to Steve Murri, who was at Commodore at the time. He suggested that I propose such a software product to Commodore, which I promptly did. Both Clif and I began developing the math routines we knew were going to be needed: for example, a spherical trigonometry package, since there is no way you can accurately represent a portion of the sky by horizontal or vertical 'scrolling' through a data base, as is normally done with game-style background graphics.

Spherical trig is tough! — and full of mathematical pitfalls. I could go through an entire chapter on the inside-out universes we created, the ones with 'black holes' in various regions, or even 'wormholes' through space whereby an object could be seen on both sides of the sky. After about a month of ploughing through such problems, however, we had a reasonable window on the sky, viewable on a PET computer equipped with an MTU 'visible memory' (a 320-by-200-pixel monochrome bit map). At about this time, the Commodore 64 was just starting to emerge, and Steve Murri obtained one of them for us. Before launching into a full-featured 'Home Planetarium' (as our proposed product was originally called), we decided to wait until my formal proposal to Commodore was officially accepted. Our personal goal of having a computer-generated sky had already been met, and we had little incentive to pursue it further without assurance that it would be a widely useful and marketable software product. At this time there, was no horizon, no planets, Moon or Sun, no world map, and no user-friendly interface.

Much to our dismay, no one (except Steve) at Commodore thought much of a home planetarium product. Typical comments from the decision makers were: "Why do I need a computer? If I want to know what the sky looks like, I can just go outside." My comments, "If it's raining out, you can just turn on the windshield wipers on your telescope, right?" and "I suppose that when you go outside and point at something in the sky, a voice coming from the Almighty tells you what it is you are looking at, and draws constellation lines in the sky," were met with appropriate but condescending smiles. No one with decision-making authority seemed to understand that the proposed program was not supposed to take the place of actually looking at the sky, but to en-

courage it. Clif and I felt that most people have enough curiosity to want to know the stars and planets, but don't have the patience and time to gain this familiarity through books and charts.

The project was put on hold for about six months. In May of 1983, Steve Murri and I were driving together to attend a TPUG symposium, at which I was going to present a talk on graphics and music. Near sunset, Steve happened to spot a bright star-like object in the western sky, and wondered out loud what that 'star' was. When I told Steve it wasn't a star at all, but the planet Venus, he was extremely interested and amazed at how I knew this. We began discussing the home planetarium project anew, and Steve grew more and more enthusiastic about it. When I reminded him that the proposal had been on hold for quite some time, he promised he would try to do something to activate the proposal when he got back to West Chester. This he certainly did, and arranged for Clif and me to meet with Commodore's new software director, Sigmund Hartmann. Sig (as he likes to be called) immediately recognized the potential of what we were proposing as a fine example of a product in the educational area. Things started to pick up immediately: a contract was agreed upon, hardware support was generous, and we were asked to finish the product by the January 1984 CES show.

Now the real work began. By this time, another colleague and friend, Dr. Fred Ancker, had joined the team, and was trying to organize Clif and myself into adopting a more business-like approach to software development than either of us were used to. Fred, who is Danish, was our chief negotiator with Sig, and they seemed to talk the same kind of language (Sig apparently has more trust in people with European accents). Fred consistently attempted to convince Clif and myself to 'farm out' more work to other people: the specifications for the home planetarium had gotten quite ambitious by this time, and it was clear that it was going to be an enormous effort. The problem, as Clif and I so often pointed out, was that by the time we would be able to specify exactly what we wanted in enough detail for it to be farmed out, we felt that we would be 95 per cent of the way to having it in source code (it was clear from the beginning that essentially all of the planetarium would have to be written in assembly language).

Nevertheless, we did get help in two important aspects. We wanted the system to have an informative piece of text on every object that the system could display, and we also wanted the user to be able to get a hard copy of any graphic screen. The amount of text on each object could differ enormously in length (a few facts on some of the fainter stars, a lot of text on the planets and some of the more prominent deep-sky objects). We needed a random access/random length disk access routine, but neither Clif nor I was well versed in the intricacies of the disk operating system. A young and brilliant fellow by the name of Paul Kriss (an avid TPUGer) was enlisted to do the disk work as well as the printer dump routine. We were helped in other areas as well — not as much as our business mentor, Fred, would have liked, but enough to consider ourselves at least beginners in the 'agricultural' method of programming.

. . . Clif and I felt that most people have enough curiosity to want to know the stars and planets, but don't have the patience and time to gain this familiarity through books and charts. . .

We started from scratch on our star data base, and were in touch with the NASA department that dealt with public domain information. It turned out that the Bright Star Catalogue was available on mag tape for only the copying and media cost. The positions, brightness, and other data for the brightest 250,000 stars was obtained, and a Fortran mainframe program was written by Keith Sproul to weed out data for stars brighter than 5th magnitude — those visible to the naked eye. By the time we had most of the program developed, data for about 1200 stars could fit into the C-64's memory; in addition, about 300 deep-sky objects (nebulae, galaxies, clusters, and so on) were added, also from NASA data tapes.

While I was working out the details of the user interface, Clif was wrestling with the problem of how to manage the computation of the Solar System objects (Sun, Moon, planets). At first approximation, the planets seem to trace out neat circular or slightly elliptical orbits with the Sun at one focus. After all, Sir Isaac Newton and Johannes Kepler had worked out the two-body gravitation problem long ago, so it couldn't be too complicated, right?. . . Wrong! It turns out that in order to be accurate enough to compute positions to a minute of angle (the limit we had decided upon as consistent with 16 bit coordinate data), the equations for planetary motion have to include the secondary effects (or perturbations) due to objects other than the Sun. The Moon is especially difficult, since it has significant perturbations from both Earth and Sun, and is an object whose position is observationally known over long periods of time with good accuracy.

By this time, Fred had gotten fascinated with historical accounts of such phenomena as lunar and solar eclipses, planetary alignments and transits, and we all desired to be able to simulate these reasonably well. The stars must have been favourably placed for us, because Clif was able to locate a journal article that seemed tailor-made for our purpose. Not only that, but it turned out that we knew one of the authors. The article presented a method and gave the data needed for computing the positions of the planets and Moon with no short or long-term drift with an accuracy of about one minute of arc. Understanding how to use the system was no simple task because it entailed a fair knowledge of astronomy, math and computer programming.

To save valuable RAM space, Clif and I decided to use the ROM floating point routines out of the BASIC area. Although Jim Butterfield cautioned us to avoid tapping into the non-Kernal area, he also was quite sure that these routines had survived essentially intact since the early days of the first PETs, and were unlikely to change for any machines on the horizon. Clif and I decided that the needs of meeting our deadline, and the savings in RAM space justified our use of the ROM routines. After falling into all the traps associated with the floating point accumulators, normalized representation, the sign comparison byte and so on, we finally got the planetary ephemeris under control. Fred had a field day asking us to check out some of the more interesting historical events he was incorporating into the tutorial sections of the manual he was writing (he wasn't able to farm that job out, either!). As a result, we had to modify the equations subtly to account for the slow deceleration of the Earth's rotation rate due mainly to tidal friction with the Moon. The Moon's orbital elements also had to be corrected for this effect.

By now, the user interface was reasonably complete, and we had met our goal of not forcing the user to have to key in any numerical information, as was the case for various astronomy programs

then available. The observer's location on Earth, and his direction of view were cursor (or joystick) controlled; the corresponding coordinate data tracks the cursor and is continuously displayed in a 'data window' on the right edge of the screen. The text access portion was not quite finished, but we felt that we had a demonstratable program in time for the CES show in January 1984.

During our preparations for CES, Clif and I were invited by Frank Winter of Commodore Canada to participate in the Commodore Anniversary meeting and show in Toronto, in mid-December. Naturally, we were going to preview the home planetarium program, and Frank creatively entitled our talk 'Magical Mystery Tour Through The Universe'. In keeping with the spirit of this unusual title, I prepared a few surprises specifically for the show. As the audience entered the auditorium, I had arranged to have the Beatles' song 'Magical Mystery Tour' playing in the background. Previously, Clif and I had been doing some last minute programming so that the CN tower in Toronto would be visible on the horizon, when the system was put to the correct latitude and longitude. Thus the machinery for the 'map tokens' was created.

I had carefully arranged things so that after the opening screen default conditions (Washington DC, January 1, 1985) I could quickly demonstrate the MAP mode by going to Toronto, and the SET mode by going to the show date and time, which was in the late afternoon. The sky colour came back blue, of course, since it was still daytime. To demonstrate the FIND feature, I had the program find the Sun; there it was, low on the western horizon, and just above the CN tower, as had been so carefully planned. I activated the clock to 64X so the audience would get to see the sunset. As the clock started to tick, I secretly activated a recording I had made of the opening music from 'Also Sprach Zarathustra' (you know, the one from 2001). The music peaked just as the Sun was passing behind the CN tower, and setting below the horizon — perfect timing, if I say so myself. Needless to say, the audience loved it. "Aha!" heckled Brad Templeton (a TPUGer of note for his brilliant programming abilities), "I bet you still see the CN tower before 1970." (The tower was completed in 1970). To the audience's delight, I was able to retort truthfully "Of course you don't see the tower before it was built, Brad," and proceeded to demonstrate.

By May 1984, all the minor bugs had been removed, and the manual was finished by Dr. Ancker and delivered to West Chester. We had previously demonstrated the planetarium program to a group from the Franklin Institute, a museum in Philadelphia, and the staff of the Hayden Planetarium, where the resident astronomer was skeptical until I successfully simulated an eclipse of the Sun in New Caledonia in November of 1984, an event to which he was making an expedition.

. . . As the clock started to tick, I secretly activated a recording I had made of the opening music from Also Sprach Zarathustra . . .

The project was still far from complete. All the details of preparing the manual for publication, and getting the disk 'protected', lay ahead of us. By this point, we were being helped by Commodore staff, notably Sandra Ritter, Doreen Carson, Diane Lebold, Barbara Feldman, Joyce Wetmore and Steve Beats. I also must give special thanks to Elizabeth Deal, who is not a Commodore employee, but is an avid PET user and super 6502 programmer. Liz lives near West Chester, and graciously played host to Clif, Fred, and myself after most of our many meetings with Commodore. Meanwhile, Dr. Daniel Kuntz, who was heading the educational software division, had gotten an appointment to the council of the newly formed Young Astronauts Program in Washington, DC. Much to our delight, Dan recommended and got acceptance of **Sky Travel** (the final name chosen for the product by Commodore) as officially approved software for the program — the first one to get such approval. After seemingly endless delays, the product was officially launched in late December 1984, just too late to have any impact for the Christmas season.

The story is not over. Commodore appears to be trapped in a vicious circle in the software area: they deal mainly with distributors who, in turn, bring the products into the retail stores. The catch is this: Commodore software has (to put it mildly) not always been of top quality, and many distributors have gotten stuck with piles of marginally saleable Commodore products. Distributors were not exactly jumping to carry **Sky Travel**, since it was to them an unknown. People who wanted it couldn't get it because it wasn't appearing in the stores. Worse yet, the very existence of **Sky Travel** seemed to be a carefully guarded secret. It seems a Commodore software product has to already be a success before it can be a success!

We have now reached the present, and the apparent paradox shows some signs of being alleviated, since the appearance of an excellent review article and back cover ad in the April *Commodore* magazine. Was it worth all the effort? Remember, we are now close to two years since the start of the programming effort, and much of the time was spent in hard work. Fred, Clif and I put our all into making **Sky Travel** as complete and user-friendly as possible: the INFORM text occupies over 400 blocks on the disk, the program itself utilizes over 60K of the 64K RAM space of the C-64, not counting overlays, and the manual is well over 100 pages long. As of this writing (April 1985), the answer is: definitely not! The rewards have been minuscule compared to the effort up to the present time. However, the story goes on, and we may eventually see our claim come true — that even people only casually interested in what's in the sky will want **Sky Travel**.

Remember . . . *the comet is coming!*□

A Beginner's BBS Guide: Part Three

by Ian A. Wright

Bulletin boarding has become very popular in recent months, and modems are among the hottest-selling peripherals. To some extent, this popularity results from a multi-media blitz caused by stories about 'hackers' presented in the movies, on television and in print. Real telecommunication, however, is an enjoyable and often very useful activity, not — as the media often portray it — a solitary and destructive one. In the first two articles of this series, Ian Wright discussed some of the uses of bulletin boards, and how to logon and read messages and bulletins. Now he explains the procedures for entering messages, and for uploading and downloading, using examples from TPUG's own Bulletin Board System (BBS). Since the download function is not fully supported by the club's board, Ian uses the Bradley Brothers' Bulletin Board System (BBBBS) for examples of that process.

The Message Editor

The story is that Steve Punter, while working on the earliest version of his text editor, decided to test it in his BBS program. I don't believe it, because **WordPro** is far superior to the adequate but annoying line editor of the BBS.

Entering a message to another BBS user or to *all* involves typing the letter **E** at the **command** > prompt, and following directions. The difficult part, for most users, is the use of the built-in editing functions. The message editor works *after* you have typed in your material — so don't format your message neatly as you enter it. Typos, spelling and grammatical errors should be checked by selecting **F** to preview your message from line one to the end. Correct your errors using **L**ist, **C**ontinue, **R**eplace, **E**dit line, **I**nsert or **D**elete line, and place **P**aragraph marker. Finally **S**end your message to the BBS disk when it's correct.

Entering a single space, followed by a carriage return, will leave a blank line between paragraphs as you enter your message, making it easier to read. Please use upper/lower case if your terminal supports it — it's much more pleasant for tired eyes at 3:00 am. The BBS will automatically add your name to the end of your message (no anonymity!) — but it's good form to sign off anyway.

Until you select **S** to send your message, you can continue to enter or correct it. You can delete any message you sent by typing **DM** at the **command** > prompt, if necessary. Messages sent *to* you can be deleted each time you sign off the board, and this helps clear the clutter of old messages.

Some terminal programs allow you to compose a message while off-line (in your word processor), and then upload it directly from disk just as if you had typed it in live. A few terminal programs have an 'editor' function that will allow you to compose the message and edit it, using full-screen editing, while you are still online, then switch to terminal mode and send it.

Private messages are 'restricted' so that only the sender, receiver and SYSOP can view them — all others cannot. Note that the SYSOP can read everything in all sections of the board. This is a check on illicit activities — it is *not* an invasion of privacy. The TPUG SYSOPs have seldom had to deal with problems such as illegal trading of copyright software, abusive language or illicit operations, because they delete any offender from the user list. You are given one free error — one warning should be enough. TPUG's BBS has a spotless record, because each of our SYSOPs has taken his responsibilities seriously.

Uploading

TPUG's BBS does allow uploading of material to the office, so that (for instance) you can send a program to the various librarians, for inclusion on a monthly disk. The process of sending a file to the BBS is simple, but many novices find it confusing at the start because it involves 'protocol'.

Protocol refers to a process of sending, receiving and checking data transmissions between computers. As most of you are aware, an IBM computer will not run Commodore programs, because the two machines 'don't speak the same language'. Even their BASICs are different. Many BBSs use a common 128-character code called ASCII to transmit text files between dissimilar machines. Common protocols include *Punter C1*, *Vidtex B*, *X-Modem* and *.img*. Each involves a different method of transmitting information between computers.

TPUG's BBS still uses the original Punter protocol — it will send three blocks of information (255 bytes each) per

Selected BBS Commands Continued

UP/DOWNLOADING

LIST	show available programs
LOAD	download a program
SAVE	upload a program
STAT	save status variable
NEXT	continue message reading
DP	delete program (you must have the deletion code) Send a 'start signal' by exiting from terminal mode to main menu, select receive/send a program and follow the prompts of the terminal program.

OTHER

E	leave a message on BBS
DM	delete a message
CAT	list message categories
DUP	change duplex
EXP	expert mode
FWD	reprint private message

Message editor: H = help, L=list,
R = replace/original/new/line #, S = send,
F = preview, E = edit, I = insert line,
P = paragraph, Space & Return = leave blank line

Note: TPUG BBS does not support DOWNLOADING

minute from your computer to TPUG's. The usual speed of home telecommunications is 300 bps (bits per second) — sometimes incorrectly called baud rate — which is the same slow speed at which the 1541 disk drive operates. Up and downloading is even slower. The delays in the Punter protocol are a result of careful checks for incomplete or inaccurate transmissions, so Steve has recently developed version C1 for the C-64, which is twice as fast and much more accurate. Unfortunately, it does not work with PET BBS systems like TPUG's. *(Since this article was written, TPUG has in fact switched to a C-64 system using the new Punter protocol. —Ed.)*

Your terminal must support the protocol of the BBS you are using. You can't upload to TPUG using a terminal program that has only X-Modem protocol. The selection of an appropriate terminal program thus becomes a major problem. Most BBSers have accumulated a number of different terminal programs because they cannot find one terminal program that does it all. I have seven!

The only tricky part to uploading a program, bulletin or message to TPUG's BBS comes when you are asked for the 'start signal', after typing **SAVE**, and entering the file name. The start signal can be sent only by exiting from terminal mode, going to the main terminal menu, and then selecting the option that lets you upload (or send or transmit) a file. Depending on your terminal, you may be asked to give the filename again and to tell the terminal whether the file to be uploaded is a program, a sequential file or a **WordPro** file. *You* don't send any start signal — your terminal program does it for you.

As the file is sent, you should see a - on the screen for each good block transmitted, and a : for blocks that were not successfully sent. These bad blocks will be retransmitted until acceptable, but a general rule is that any five bad blocks constitutes a questionable upload (or download) and you should try again later.

Downloading

Since TPUG's BBS has elected to have almost 700 users, the downloading section has been removed from the club board. I will describe downloading as it applies to local Punter-protocol systems. **LIST** is the command that will give you a listing of programs available for downloading, allowing you to select from the various menus that appear (pause/abort works the same here as in the message section). Not all files can be downloaded by everyone. Sometimes files will be uploaded with a password so that only those with the password have access to them. This is shown with a * beside the name when you list the files. Write down the name of your selection *exactly* as it appears in the list. At the **command >** prompt, type **LOAD**, and then the name of your selection as prompted. Wait while the BBS looks for the file. You'll see the file name repeated, and the estimated time of transmission (how long it will take to download the data).

At this point you can abort the download (use **SHIFT-A**) or send a start signal. The start signal is the same as that described above for uploading — with one difference. Enter the name of the file to be written to disk as **"0:filename"** to write to drive 0, even with a 1541 drive. I have found that this eases the download process greatly. At the end of a successful download, you will see your name, the file name, the blocks downloaded, and the time.

Up/downloading from Compuserve, DOW Jones, or the company VAX is no different in principle to using a BBS, and every attempt is made to ensure a simple and accurate exchange of data. Electronic communication can be a doorway to a whole new world of people and events — try it!

In the next article, I will give some tips on the selection of modems, terminal programs and phone lines for use in telecommunicating. If you have any questions please write to me c/o *TPUG Magazine* — or send E-mail to TPUG Office on the club BBS (416-782-9534). □

Important message to all BBS users

TPUG BBS

The telephone number is:

(416) 782-9534

Operating hours:
24 hours per day
7 days per week
The password is . . .

MONO

Not Just A Pretty Picture

```
100 rem Koala splitter by nicK sullivan
110 :
120 if sw=1 goto 230
130 poKe 251,peeK(55): poKe 55,0
140 poKe 252,peeK(56): poKe 56,96
150 clr: sw=1
160 close 1: open 1,8,15,"i0"
170 gosub 320
180 input "Koala file";Kf$
190 Kf$=chr$(129)+Kf$+"*"
200 open 2,8,0,Kf$: gosub 320: close 2
210 load Kf$,8,1
220 :
230 print "change disKs, press return"
240 get a$: if a$<>"" goto 240
250 get a$: if a$<>chr$(13) goto 250
260 for i=1 to 3: gosub 380: next i
270 poKe 55,peeK(251)
280 poKe 56,peeK(252)
290 clr: end
300 :
310 rem checK error channel
320 input#1,e,e$,t,s
330 if e<20 then return
340 print "<down>disK error:";e;e$;t;s
350 close 2: close 1: goto 270
360 :
370 rem save a file
380 read a$: print "saving " a$ "..."
390 print#1,"i0": gosub 320
400 open 2,8,2,a$+",p,w": gosub 320
410 read a: print#2,chr$(0);chr$(a);
420 read a,b
430 for j=a to j+b
440 print#2,chr$(peeK(j));
450 next j
460 close 2: return
470 :
480 rem hi-res (bitmap)
490 data file1, 32,24576,7999
500 :
510 rem low-res (video matrix)
520 data file2,<2 spaces>4,32576, 999
530 :
540 rem colour memory
550 data file3,216,33576, 999
```

*The **DATA** statements in lines 490, 520 and 550 of this program each contain four items. The first is a filename, which you can change to suit your own taste. The second is the page at which the file will be loaded in memory — multiply this number by 256 to find the actual load address. You should change this number only if you are going to use bank switching as described in the article. The third number, which represents the start address of the data as loaded in the **KoalaPainter** file, and the fourth number, which represents the number of bytes to be stored minus one, should not be changed.*

by Dave Neale

The other day I was sitting in front of my terminal wondering what sort of programming I could do to change the looks of one of my programs. I'd seen the multicolour hi-res creations that have come out of the **KoalaPainter** program, and I thought, "Why not use those pictures to enhance programs?" After poking around for a while, I found that it is a lot easier than one might think.

The first step is understanding how **KoalaPainter** stores all of the information for a multicolour hi-res screen in one 40-block file. From the quick loader program in the **KoalaPainter** manual, I found that the 40 blocks include four separate data storage areas. The first 32 blocks are the actual bitmap data for the hi-res screen. Next come two 4-block sections containing data for the video matrix (low-res screen memory) and colour memory areas respectively. These contain colour information for the multicolour screen. The final byte of a **KoalaPainter** file is the background screen colour.

Before you start fooling around with files you should ask yourself how you want to use the hi-res picture. You could put it in a loading program for a title page, as I did in my **Star Trekking** program, or use it as a background in some sort of game that uses sprites. What you choose will help you decide where to put your files for their best usage.

If you just want to use one of your **KoalaPainter** pictures on the screen and not worry about video banking (we'll be getting into that later), you have to solve one little problem — that of splitting the compound **KoalaPainter** file into three files, each with its own load address. **Program 1** *(see box)* accomplishes this task for you. Make sure the disk with your **KoalaPainter** picture is in the drive when you run the program.

All that's left now is to load the files, and to set up the video chip for multicolour hi-res graphics. See **Program 2** for an example of how to do this.

The first line of the loader should always clear the screen and set the border and background colours, so set them according to the way the hi-res picture was created. Remember that the **LOAD** command works differently in a program than it does in direct mode — after a pro-

grammed load, execution resumes at the beginning of the program, not at the statement following the **LOAD** command. This is not a bug — it is designed to allow the chaining of BASIC programs. The function of the variable *F* in **Program 2** is to dodge around lines that have already been executed.

Before you load your file into the low-res screen area, it's good practice to stop the screen scan. That way you won't get any garbage showing on the screen while the load is in progress. Line 20 in **Program 2** accomplishes this.

Lines 60 through 80 tell the computer to turn on multicolour hi-res mode, and set the address for the hi-res map. Note variable *X* in line 80. *X* is the offset within the current 16K bank to the bitmap storage area. Since we are loading our bitmap to address 8192 (8K, hex 2000) we set *X* equal to 8.

The graphics loads completed, we switch the screen scan back on with line 90. Now — assuming the graphics screen is a title page for another program — we load the main program in with line 100. Execution will begin with the first line of that program as soon as the load is complete.

It's that simple. Now all that's left is to tell the VIC-II chip that you won't need hi-res graphics after the load is finished. Just start your main program with these lines:

```
10 clr:print"<clr>":poke
   <space>53280,bordc:
   poke 53281,backc
20 poke 53265,peek(53265
   )and239:rem turn off
   <space>screen scan
30 poke 53265,peek(53265
   )and223:rem turn off
   <space>bit map graphi
   cs
40 poke 53272,(peek(5327
   2)and240)or4:rem rese
   t character data addr
   ess
50 poke 53265,peek(53265
   )or16:rem turn screen
   <space>scan back on
60 rem start of main pro
   gram
```

Creating a simple loader program is fairly easy, but there can be a few drawbacks. If the main program is longer than 24 blocks, it will overrun the bitmap data area, causing your picture to look... a little different. There are two ways around this. The first, video banking, allows you to put the screen and bitmap way up in Bank 3, well out of the way of your main program. Since only Banks 0 and 2 have character images available, you will have to create your own character set and store it somewhere in that bank. The second method is relocation of the start of BASIC. Moving the start of BASIC up to 16384 (hex 4000) will give you 96 blocks of RAM to use, with no risk of interference with the bitmap.

Since it's an unfamiliar area to many, let's take a quick look at the steps necessary to switch video banks. With four banks (16K each) to choose from, you must find an area in one bank to hold your bitmap data. The banks are numbered 0 to 3, with the default being bank 0. Here is a chart showing the memory that each bank occupies:

Bank	Memory Range
3	49152-65536 ($C000-$FFFF)
2	32768-49151 ($8000-$BFFF)
1	16384-32767 ($4000-$7FFF)
0	0-16383 ($0000-$3FFF)

Since the VIC-II chip can only access 16K at a time, you will have to put the low-res screen in the same bank as your bitmap. After you've chosen the bank to use, you switch it in by **OR**ing a value into the two low bits of Port A of CIA chip #2. But first you should make sure that these bits are set for output. Though this is the default condition, better safe than sorry. It only takes one line:

```
poke 56578,peek(56578)or
3:rem confirm port a set
<space>for output
```

Now change banks:

```
poke 56576,(peek(56576)
and252)or(3-bank):rem pe
rform bank switch
```

These two lines should be added to the loader either before the data load or just after. Also, the bank must be switched back to the normal values before the main program starts running; use these two lines again in the added section of the main program. Don't forget to set the start address of each file to somewhere in the bank you choose (by altering the first number in the **DATA** statements of **Program 1**).

With a little bit of practice, you can set up the system to allow switching between hi-res and low-res screens, or between several hi-res screens for an even better effect. The only drawback to screen banking comes when you want to use the ROM image of the character data. Only Banks 0 and 2 have this image, and if you wanted to use the Commodore characters you would have to transfer them into the bank you're using. Also, try to stay away from putting a graphics area on top of one of the ROM character images.

Assigning areas of memory for graphics screens is an art as well as a science, and there are tradeoffs involved. With a bit of thought, you should be able to come up with an arrangement that fits the needs of your program. Luckily, colour memory never moves from its area at 55296 (hex d800), and is accessible in any bank, so there's nothing to worry about there. Understanding graphics can improve your programs and impress your friends. Good luck and happy programming. □

```
10 if f=0 then f=1:print"<clr>":poke 53280,bord
   c:poke53281,backc
20 if f=1 then f=2:poke53265,peek(53265)and239:
   rem turn off screen scan
30 if f=2 then f=3:load"bitmap data",8,1:
40 if f=3 then f=4:load"screen char data",8,1
50 if f=4 then f=5:load"colour ram data",8,1
60 poke53265,peek(53265)or34:rem set bit map gr
   aphics on
70 poke53270,(peek(53270)and223)or16:rem turn o
   n multicolour graphics
80 poke53272,peek(53272)or x:rem tell vic chip
   <space>where to find bitmap data
90 poke53265,peek(53265)or16:rem turn the scree
   n scan back on
100 load"main program",8,1
```

This program will load the three files created by Program 1 into the appropriate areas of memory, then load and run your main program. Remember to change the dummy filenames in this program to the filenames you have chosen.

Keyboard Out Joystick In

by Michael Quigley

Many games written for the VIC 20, and a few for the Commodore 64, use keys such as I, J, L, and M (up, left, right and down), rather than a joystick, to move objects around on the screen. I've always found this curious, since just about everyone I know has a joystick.

Converting keyboard programs to joystick is not particularly difficult — you just have to know what to look for. Most of these programs use memory location 197 ('current key pressed') to direct the cursor, rocket ship or whatever, around the screen. Each key on the keyboard (except the logo key, the **SHIFT** keys, **CTRL** and **RESTORE**) has a value in this location when it is pressed. Type in the following two lines, type **RUN** and then press some keys.

```
10 print peek (197)
20 goto 10
```

Try typing I, J, L and M, and note the values. They should be 12, 20, 21 and 36 respectively on the VIC, and 33, 34, 42 and 36 for the C-64. Note that when no key is pressed, the value is 64 on both machines.

Now comes the detective work. Look through your program and try to find lines which are something like the following:

```
100 if peek(197)=64 then.. [no input]
110 if peek(197)=12 then.. [up]
120 if peek(197)=20 then.. [left]
130 if peek(197)=21 then.. [right]
140 if peek(197)=36 then.. [down]
```

Since the results of pressing these keys vary from program to program, the four example lines here are purposely left incomplete. The first line, corresponding to 'no key pressed', may or may not be present. Of course, some programs also allow movement in only two directions, not four. In such cases, the lines for horizontal or for vertical movement may also be missing. Another thing to watch for is if **PEEK(197)** is converted earlier in the program to a numeric variable, for example: **P = PEEK(197)**. Then the lines above would read **100 IF P = 64 THEN...**", and so on.

Now that you've found the place where keyboard inputs are interpreted, what is the magic joystick formula to use? It's as follows:

```
0 goto 10: rem vic 20 version
1 qq=37154:q1=37151:q2=37152
2 poke qq, 127:q=peek(q2)and128
3 r0=-(q=0)
4 poke qq,255:q=peek(q1)
5 d0=-((qand8)=0)
6 l0=-((qand16)=0)
7 u0=-((qand4)=0)
8 fb=-((qand32)=0)
9 return

0 goto 10: rem commodore 64 version
1 q=peek(56320)
2 r0=((qand8)=0)
3 d0=((qand2)=0)
4 l0=((qand4)=0)
5 u0=((qand1)=0)
6 fb=((qand16)=0)
7 return
```

This standard routine should be placed somewhere in your program, preferably at the beginning to speed things up. The line numbers can be changed if you wish. Notice that I have used some variable names that suggest the direction they will take — *U0* for 'up', *D0* for 'down', and so on. Be sure that none of the variables in the joystick formula are the same as those already in your program, otherwise things will get really messed up!

Now go back to those lines you found earlier — the ones for interpreting keyboard input. Make the following changes:

```
100 gosub 1:if u0 or l0 or d0 or r0 o
    r fb then 110
105 rem perform action for 'no input'
110 if u0 then... [up]
120 if l0 then... [left]
130 if r0 then... [right]
140 if d0 then... [down]
```

The new line 100 first **GOSUB**s to the joystick subroutine. If a joystick input is made, it jumps to line 110, where the input is decoded and acted upon. If *nothing* happens (the equivalent of **PEEK(197) = 64**), then the 'no key pressed' routine, if any, may be accessed in line 105.

In the last couple of examples, we've used a new variable, *FB*, for 'fire button'. This input may not be needed, but if it is, the process for making the conversion is analogous to the cases we have already discussed.

Another way in which keyboard inputs are handled on the VIC and C-64 is with a **GET** statement. Assuming we are using 'S' for fire, 'I' for up, 'J' for left, 'L' for right, and 'M' for down, a typical keyboard checking routine might look like this:

```
100 get q$
110 if q$="s" then 1000 [go to fire b
    utton routine]
120 if q$<>"i"andq$<>"j"andq$<>"l"and
    q$<>"m"then300 [no input]
130 x=x+(q$="j")-(q$="l") [determine h
    orizontal direction]
140 y=y+(q$="i")-(q$="m") [determine
    <space>vertical direction]
```

Here's the same routine, converted for joystick operation:

```
100 gosub 1 [go to joystick formula]
110 if fb then 1000 [go to fire butto
    n routine]
120 if u0+l0+r0+d0=0 then 300 [no inp
    ut]
130 x=x+(l0)-(r0) [determine horizont
    al direction]
140 y=y+(u0)-(d0) [determine vertical
    <space>direction]
```

I find using a joystick highly preferable to groping around the keyboard, and if you're really lazy, there's yet another trick to be learned. Instead of the usual keyboard input to rerun a program:

```
100 print"play again? (y/n)"
110 get a$:if a$<>"y"and a$<>"n"then1
    10
120 if a$="n"then end
130 if a$="y"then run
```

try the following:

```
99 rem vic 20
100 print "play again? press fire but
    ton"
110 wait 37137,32
120 wait 37137,32,32
130 run

99 rem commodore 64, port #1
100 print "play again? press fire but
    ton"
110 wait 145,16
120 wait 145,16,16
130 run

99 rem commodore 64, port #2
100 print "play again? press fire but
    ton"
110 wait 56464,16
120 wait 56464,16,16
130 run
```

With this routine, your hands need never leave your joystick, except when you want to stop the game with **RUN/STOP-RESTORE!** □

The Better Way

by Chris Johnson

Is there a better way to set off sections of a program in listings? The usual method is to enclose **REM** statements in asterisks. They take up a lot of space and they don't really stand out that much.

A much better way is to put your **REM** statements in reverse type: they really stand out, whether listed to the screen or to a printer. Try it some time.

But how? The BASIC screen editor will strip out a reverse video code from your **REM**, no matter how hard you try to fool it. To get what we want, we'll have to bypass the screen editor and poke the code (whose PET ASCII value is 18) directly into BASIC program space, specifically into the byte following the **REM** token.

The following program is a BASIC loader that will poke a machine language routine into memory. A **SYS** to the location of the routine will put your **REM**s into reverse at machine language speed — that is, quickly!

The **REM** statements at the beginning can be your first test of the routine. You cannot put **REM**s into reverse from the keyboard, so when you have typed in this program (and have saved it, to be on the safe side), run it and save a copy with reversed **REM** statements.

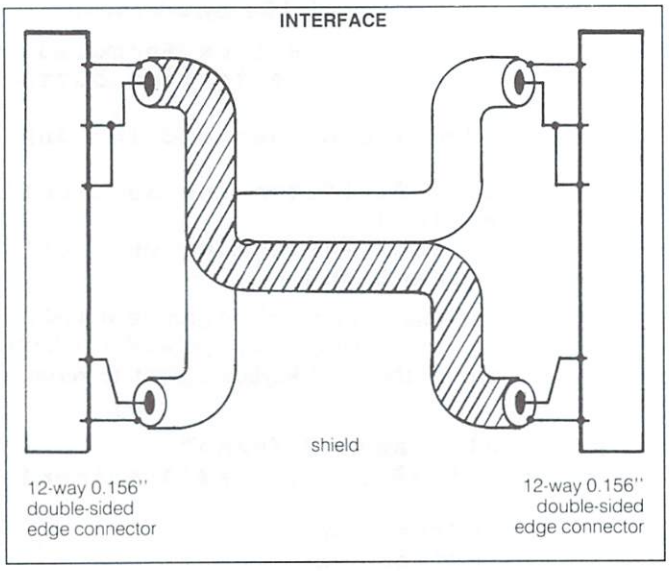
The machine language itself can be put anywhere in memory that you choose. Simply change the value of *S* in line 9. (C-64 utilities such as **Power** or **Basic Aid** use the cassette buffer as a workspace, so 49152 might be a better location.)

You can use this routine to put codes other than RVS into your listings. If you wanted the comments to appear on a new line, you could have the routine insert a 13 (carriage return) instead of an 18 (RVS). Run the loader, then type **POKE S+81,13**. Now **SYS** to the starting location, and the text after your **REM** statements will start a new line.

This loader was created with **Datacheck** (from *TPUG Magazine*, May 1985), a program that creates a BASIC loader with a checksum at the end of each line. If you make a mistake in typing, the checksum will catch it, stop the program and tell you the number of the line containing the mistake. □

```
0 rem<33 spaces>:
1 rem<2 spaces>highlight your rems
  <12 spaces>:
2 rem<33 spaces>:
3 rem<2 spaces>to change machine lang
  uage<5 spaces>:
4 rem<2 spaces>location, change value
  <space>of 's'<2 spaces>:
5 rem<2 spaces>in line 9<22 spaces>:
6 rem<33 spaces>:
7 rem<2 spaces>by chris johnson
  <15 spaces>:
8 rem<33 spaces>:
9 s = 828
10 fori= s to i + 95 step8
11 forj=0to7:readа:poke(i+j),a:c=c+a:
   nextj:read cs
12 ifc<>csthenprint"<2 down>error in
   <space>line";i:forz=1to25:a=abs(a-
   15):poke54296,a:next:stop
13 c=0:nexti
14 print "<clr><7 down> to put revers
   e rems into your program,
15 print tab(6) "<down>load your prog
   ram and type:
16 print tab(15) "<down>sys" s "
   <6 down>"
17 end
828 data<2 spaces>165, 43, 56, 233, 1
    , 133, 25, 165, 821
836 data<2 spaces>44, 233, 0, 133, 26
    , 169, 0, 141, 746
844 data<2 spaces>153, 3, 160, 5, 230
    , 25, 208, 2, 786
```

```
852 data<2 spaces>230, 26, 136, 208,
    <spare>247, 160, 0, 177, 1184
860 data<2 spaces>25, 208, 12, 238, 1
    53, 3, 173, 153, 965
868 data<2 spaces>3, 201, 3, 240, 47,
    <space>208, 227, 140, 1069
876 data<2 spaces>153, 3, 201, 143, 2
    40, 2, 208, 28, 978
884 data<2 spaces>165, 25, 24, 105, 1
    , 133, 27, 165, 645
892 data<2 spaces>26, 105, 0, 133, 28
    , 160, 0, 177, 629
900 data<2 spaces>27, 208, 5, 238, 15
    3, 3, 208, 4, 846
908 data<2 spaces>169, 18, 145, 27, 2
    30, 25, 208, 197, 1019
916 data<2 spaces>230, 26, 208, 193,
    <space>96, 0, 0, 0, 753
920 rem<2 spaces>version 3, 12 may 19
    85<7 spaces>:
```



Talking to Other VICs

by Mark Hopkins

Is your VIC 20 feeling a bit lonely? Teach it to talk to other VIC 20s, using a simple program and a couple of connectors!

The interface described below makes use of the RS 232 communication facility provided on all VIC 20s. However, you do *not* need the RS 232 cartridge in order to make it work.

Things you will need

In order to try this experiment, you will need two 12-pin double-sided 0.156" edge connectors, a metre or two of figure 8 shielded cable, and — of course! — a second VIC 20. These are cheap enough now that everyone can afford two, but if you can't, talk a friend into bringing his or hers.

What to do

Wire the connectors exactly as shown below, then carefully plug one into the user port on the left-hand side of each VIC 20.

Next, type the following program into *each* computer, and then **SAVE** it on tape or disk (be careful to include the semicolons at the end of lines 130 and 160):

```
100 open 2,2,0,chr$(6)+chr$(160)
110 get m$
120 if m$="" then 140
130 print#2,m$;
140 get#2,y$
150 if y$="" then 170
160 print y$;
170 goto 110
```

Now **RUN** the program on both computers. It doesn't seem to do much, does it? Well, try pressing some keys on one computer: do they appear on the screen of the other? To get characters to appear on *your* screen, you'll have to type them on the keyboard of the other VIC 20. Confused yet?

This is like a telephone, only you *read* the other person's words, instead of listening to them.

You could increase the length of the interface cable to several metres and have the VIC 20s in different rooms (a computer 'intercom'!)

With this set-up, you can also try the following. Press **RUN/STOP** on one computer only, and type in this line (no line number necessary):

```
cmd4:list
```

Now hit **RETURN**, and watch the screen of the other VIC 20. You should see the listing appear on the screen.

To make the listing appear on *your* screen, type the following line (again, use no line number):

```
print#2
```

Now hit **RETURN**. Then type **LIST**, and the listing will appear on your screen.

Program Explanation

For those of you who are interested, here is a brief explanation of how the program works.

Line 100 opens a channel to the RS 232 port (device 2), setting the format and baud rate. In **Line 110**, a character is fetched from the keyboard and printed to the RS 232 port by **Line 130** (if it is not a null character). **Line 140** fetches a character from the RS 232 port and **Line 160** prints it to the screen (if it is not null). The process repeats for further characters.

For those of you who want to know more about the RS 232 port and files, the *Programmer's Reference Guide* (pages 251 to 260) will prove useful. □

C-16 and Plus/4 library disks

Much of the friendliness of the C-128 (see page 18) has been available for some time in the BASIC 3.5 on the C-16 and Plus/4.

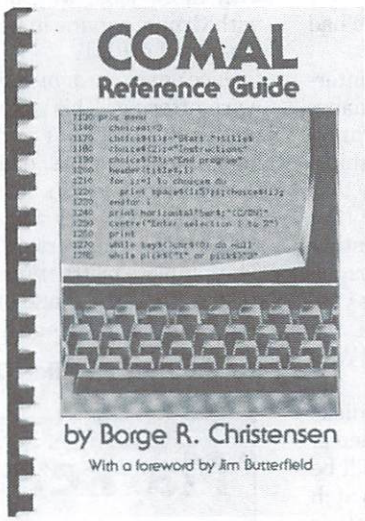
Many programs in the TPUG library can be converted easily to run on the C-16 and Plus/4. If you have done this, please submit the conversion to the TPUG library. We hope to announce at least one combined C-16 and Plus/4 disk in the next issue of TPUG magazine. Original programs are encouraged, too, of course.

Chris and Colin Johnson, C-16 and Plus/4 librarians

COMAL

REFERENCE GUIDE

Sixty-four pages outlining all the C64 COMAL keywords, with sections on the language's superb String Handling, Procedures and Parameters, Expressions and Standard Functions. **$9.95**



Send cheque or money order to:

TPUG Inc., Dept. A., 1912 Avenue Rd., Ste. 1, Toronto, Canada M5M 4A1

NEW FOR HIGH SCHOOL ADMINISTRATORS

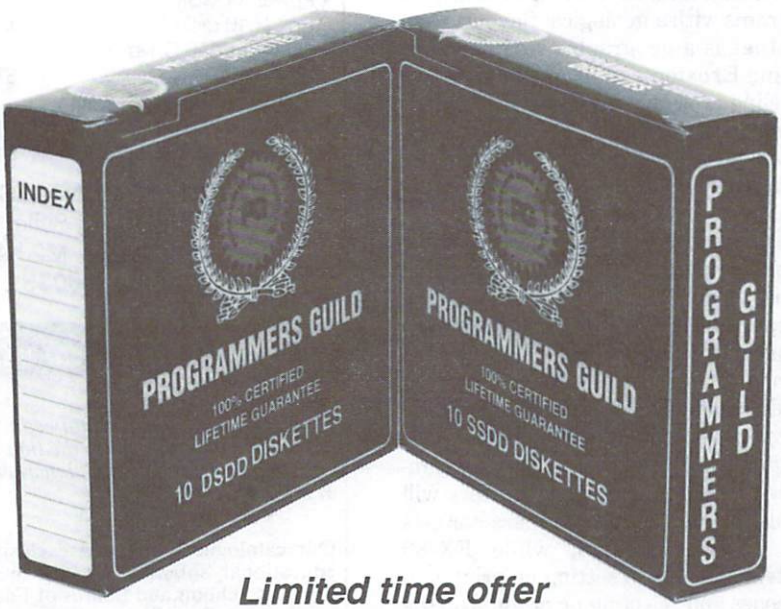
IF A STUDENT WANTS TO CHANGE HIS TIMETABLE . . . CAN YOU DO IT?

YES YOU CAN!

The WYCOR Class Scheduling System lets you:

- Add new students, change existing timetables.
- Checks all available classes, class sizes.
- Displays 32 best timetables that minimize class sizes.
- Lets YOU pick the timetable.
- Prints the new timetable.
- Updates class size.
- Prepares a teachers change list.
- Other reports avaliable.

WYCOR BUSINESS SYSTEMS
170 The Donway West, Suite 401
Toronto, Canada M3C 2G3
Tel. (416) 444-3492



Switch

to the highest quality, lowest priced brand name diskettes available in Canada.

Q. Who has switched?
A. Professional and educational institutions, national accounting firms, major engineering firms, software houses, computer retailers and the Toronto Pet Users Group.

Limited time offer

SSDD - $15.00/box
DSDD - $20.00/box
(Minimum purchase - 5. Delivery extra.)

Programmers Guild Products Ltd., 255 ManuLife Place, 10180 - 101 St., Edmonton, Alberta T5J 3S4 Phone: (403) 428-6229

VISA MasterCard

Library Additions

TPUG's library of public domain software grows month by month. Hundreds of disks containing thousands of programs are available to TPUG members at the nominal cost of ten dollars per disk. Considering that each disk is packed with good programs, at today's software prices this is a fantastic value.

In order for the library to keep growing, our librarians need a constant supply of new programs. If you have written a program or a collection of programs that you think might be an asset to the library, please send it to: TPUG Program Library, 1912A Avenue Road, Suite 1, Toronto, Ontario M5M 4A1, Canada. If your contribution is accepted you will be sent the library disk of your choice.

Commodore 64 April Disk: (C)TI

This disk has lots of interesting stuff. As promised last issue, **Directory Assistance** appears here, along with instructions. This program will let you do just about anything you could ever want to do to the directory of a disk. Remember, this is a freeware program, so please send the author what he requested if you like the program.

Adventure game fans will enjoy **Diplomat's Dilemma**. It, too, is freeware, and is very challenging. I have played it for a while, but I always seem to fall into a pit or get captured.

There are a number of interesting disk utilities as well. One allows you to search an entire diskette for any combination of characters; another will compare the contents of two disks; another will allow you to compare the contents of two files. I think you'll find them useful.

A few months ago some local talent put together quite a demo program. It plays music, it manipulates sprites, and it does a mean moon walk too! I could say more, but this program must be experienced.

For you game players who have been looking for a very different type of game, **Ghoul Dogs** is here at last! Make sure you read the instructions that have been included on the disk or you will miss a lot of what is going on.

Watch also for the adventure game disk, (C)GG, which should be out by the time you read this. Anyone that even slightly enjoys adventure games will want it. Luckily I was able to get my only copy away from TPUG's resident adventure game enthusiast, Marya Miller, and get it to the copying complex before she had worn the disk out.

By the way, I recently met a very interesting TPUG member from Burnaby, British Columbia. I was out there watching the National 5 Pin Championships, and he happened to work at the alley that I was hanging around. It turns out that he has developed some very interesting software to keep track of bowling league statistics. If this application interests you, feel free to write to: Doug Jewell, c/o Lougheed Lanes, 4199 Lougheed HWY, Burnaby BC, V5C 3Y6.

The re-organization of the older disks is coming along and should be finished in time for the conference — which will be over by the time your read this. Watch future editions of the magazine for details about these new disks.

Presented by David Bradley

PET April Disk: (P)TI

This disk features several educational programs with a geological flavour. **Geo-Weather** is a geographic weather quiz; **Marine Erosion** uses graphics to model shoreline erosion; and **Open Pit Mine** is a game in which you try to open pit mine the Athabasca oil sands. This game includes cave-ins, floods, and other problems, such as pollution costs.

Business programs this month include **Graph Sprint80**, a utility for drawing and editing graphs on your MT Sprint 80 printer; **Loan Payments v2.8**, a program to calculate interest paid and current balance under non-uniform payment schedules; and **Receipts.8**, a program to print and record customer deposits.

Those whose printers have programmable character and image buffers will be able to define their own character sets with **Printer Char.8**, while **FX-80 Fonts.8** will help in setting up print control codes and selecting print fonts on the Epson FX-80 printer. With **TapeLabler** you will have a convenient way to print cassette tape content lists.

Two specialized utilities for specialized people are **Guitar Frets**, for calculating fret positions when you're setting up your own guitar; and **Payload V2**, which assists you in calculating the fuel consumption and flying time with reserve of your airplane. And when you're finished with those jobs, why not take a break with **Groan**, a game in which you roll dice against the PET.

Once again our monthly disk has a freeware offering. This one is called **Archimedes**. It's a set of programs, for junior high school teachers, dealing with Archimedes principle. Six lessons are included. They can be run on either the PET/CBM computers or the Commodore 64. Complete instructions and a teachers' guide will also be found on the disk, along with a message from the author.

Presented by Mike Donegan □

SOFTWARE ORDER FORM

TORONTO PET USERS GROUP

1912A Avenue Road, Suite 1
Toronto, Ontario
M5M 4A1

416-782-9252

416-782-8900

Name _____ **Membership No.** _____

Street Address _____

City/Town/P.O. _____

Province/State _____

Postal/ZIP Code _____ Telephone _____

DISKS

To order club disks by mail, send $10.00 for each 4040/2031/1540/1541 disk (4040 format), and $12.00 for each 8050/8250 disk (8050 format). We do honour purchase orders from school boards.

These disks are for use with a _____ computer and a _____ disk drive.

Please send me the following:

3 Letter/No. Code	Description	4040 or 8050 Format	Price

Total $ _____ .00

TAPES

To order VIC 20 or Commodore 64 library tapes, send $6.00 for each tape.

To order PET/CBM or Commodore Educational Software, send $10.00 for each tape.

These tapes are for use with a _____ computer and a datasette.

If for a PET computer, what model _____ - BASIC- 1.0( ), 2.0( ), 4.0( )?

3 Letter/No. Code	Description	Price

Total $ _____ .00

The prices indicated include postage and handling as well as Ontario Provincial Sales Tax (if applicable).

☐ Cheque/money order enclosed (payable to TPUG) ☐ Visa/Mastercard # _____

Expiry date _____ Signature _____

Reviews

Flexidraw 4.0 and Flexifont
from Inkwell Systems
Light pen and associated software for Commodore 64

Review by Dave Neale

Flexidraw, the interactive light pen and software package by Inkwell Systems of San Diego, opens up a whole new area of graphics for the Commodore 64. **Flexidraw** combines simplicity with state of the art hardware to produce superb high resolution pictures.

When I first saw what **Flexidraw** was capable of doing, I thought that it would take quite a while to pick up some of the advanced features of the system. I couldn't have been further off track. **Flexidraw** comes with the best documentation I have ever seen for any piece of hardware or software. The 64 pages of the manual are filled with simple and easily read explanations, as well as detailed pictures of almost every function of the system. The Table of Contents is comprehensive, allowing you to zero in on any area of confusion. A glossary of terms rounds out the manual nicely.

The **Flexidraw** Master Disk is double-sided, and contains several graphic manipulation programs. These include **Flexidraw 4.0**, the main drawing and design program; **Pen Palette**, for colouring **Flexidraw** creations; **Transgraph**, for transmitting **Flexidraw** files via modem; **Sprite Editor**; **Sprite Animator**; **Display Picture**, a printer dump program; and **Follow Me**, a game.

The main program, **Flexidraw 4.0**, divides the screen into two areas: the Work Area and the Menu. Unlike other systems where you have to switch back and forth between screens to change modes, **Flexidraw** neatly packs everything onto one screen. The package of commands or modes available allows you to do virtually anything with the standard hi-res screen. Thanks to a 'dynamic' menu system, only the currently appropriate modes are displayed. The many options include: *Pixel*, *Grid*, *Draw*, *Point*, *Rubber*, *Sketch*, *Spray*, *Arc*, *Circle*, *Ellipse*, *Box*, *Line*, *Invert*, *Crosshair*, *Zoom*, *Fill*, *Shade*, *Copy*, *Paste*, *Test*, *With*, *Over*, *Rotate* and *Flip*.

Many options are powerful and imaginative. *Grid*, for example, allows shapes and text to be placed only at character block locations, which makes neat design work much easier. *Rubber* mode creates a rubber-banding effect between two points on the screen. This lets you move the free end of the line around the screen till you find the positioning you want. *Sketch* produces a continuous line, while *Spray* activates only the occasional pixel, allowing you to use the light pen as an airbrush. *Crosshair* is important if you want images to line up across the screen, horizontally or vertically. *Copy* and *Paste* allow you to move a block of the screen from one position to another.

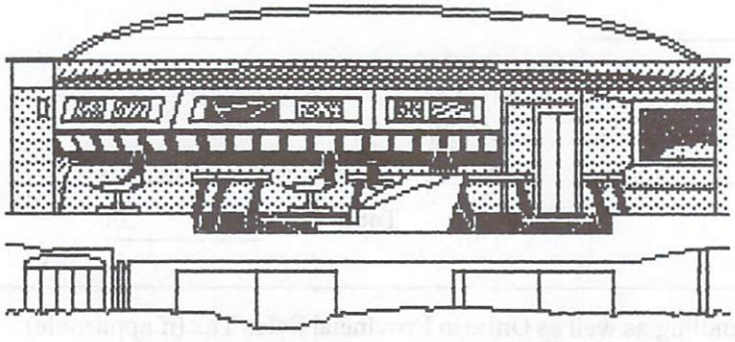
As with most systems, two work areas are available at a time. **Flexidraw** goes one better, though, by allowing you to use a split screen effect (horizontally or vertically) to have the screens overlap one another. This effect helps to produce one large hi-res screen, which can then be carried over to the printing option to allow you to produce a quality high resolution printout of virtually any size. The **Flexidraw** Master Disk contains several printer files, and I was told by a representative of Inkwell that they will create or supply any file not presently available on the disk.

Inkwell Systems has released another program, for use with **Flexidraw**, to help create custom fonts. Called **Flexifont**, this utility is used with the light pen to design your own fonts, special characters or symbols. Included on the disk are thirty-three different fonts. As with **Flexidraw**, **Flexifont**'s manual is very well laid out, and there should be no difficulty getting to know the system.

I found the system extremely easy to use. The only drawback I came across was during the *Zoom* mode. I found it sometimes difficult to activate an individual pixel without simultaneously activating others nearby, causing a bit of frustration. This problem may have been caused by static electricity; the **Flexidraw** manual does recommend wiping the screen off occasionally with a damp cloth to prevent the build-up of charge.

Flexidraw 4.0 and **Flexifont** are available from:

Inkwell Systems
P.O. Box 85152 MB 290
7677 Ronson Road
San Diego, CA 92138
Telephone (619) 268-8792 □



Edumate Light Pen with 'Peripheral Vision' drawing program
from Futurehouse
for Commodore 64

Review by Mike Martin

The Edumate Light Pen is an inexpensive data input device that reads the monitor and plots the position of the pen against the screen. This makes it possible to use the pen, in combination with suitable software, as an alternative to the keyboard or a joystick for interacting with the computer. The device plugs into joystick port 1.

The pen includes several demonstration programs that work quite well. They include a **Menu** program that uses the light pen to select from your directory. Due to the large boxes to aim the pen at, only four programs are listed on the screen at one time. Long directories seem even longer. A **Tic-Tac-Toe** game is included, a drawing program, and a simple, but impressive musical scale program displaying about six octaves. Touching the spot next to the note plays it, but no provision is included for playing or saving actual songs. A **Disk Utility** program is included for using the light pen to scratch, validate or format a disk. While all the programs work well, their usefulness is questionable. To use the pen, you touch it to the screen, then hit either the Commodore or f7 key to activate the pen. Most menu programs have you touch one key to load the program anyway. I don't see an advantage...

The light pen works with several children's educational games, but is currently being marketed with a powerful machine language art program. While the art program is outstanding, the light pen is more of a problem than an advantage. **Peripheral Vision** is an icon-based drawing program in which the icons, or pictures, are accessed by the light pen. That part works well. The drawing part has problems. The sixteen colours are displayed across the top of the screen. Touch one with the light pen, and you draw in that colour. Touch the brush width on the bottom of the screen, and cycle through six widths. Point the light pen at the centre 'drawing pad' part of the screen, and hold down the Commodore or f7 key to draw. But just try drawing an accurate line or curve freehand on the screen. Draw too fast, and you will leave gaps. Even drawing slowly will leave some gaps, as much as half an inch in width. Matching up the position of the pen to fill those gaps is almost impossible. Luckily, you can touch the *Microscope* icon, and magnify any portion of the drawing eight times to clean up the sketch pixel by pixel. In the *Microscope* mode, the lines are wide enough that you can draw fairly accurately, and scroll any direction using the cursor keys.

Other icons on the bottom screen row access disk load/save, print, mirror image draw, sensitivity, move, shapes, fill, lettering and textures. The program will plot squares, triangles, and almost-circles for you. The fill feature includes a safeguard. If the area is not totally enclosed, it won't fill, but will indicate where the fill would leak out. You then use the microscope feature, and block the opening. It plots lines — even jagged lines — but if you try to use the same point for another line, chances are you won't hit very close to your target. You can use all the letters, numbers, and Commodore graphics characters, but you must cycle through the entire set to find the one you want, then use the light pen to position the letter and hit the Commodore or f7 key. How accurate is your spacing and alignment? You can't reach any of the characters through the keyboard. And the manual warns that when you save a drawing, you are limited to 8 characters in the title, and the program doesn't read the z,c,b and m keys.

Probably the most interesting and useful feature offered is the *Texture* icon. Point to this, and cycle through 35 textures. You can use them to fill a shape, or draw with them in the six standard brush widths. The textures are a set size and spacing, but include provisions for creating up to six of your own at a time, and saving them to disk. You don't actually draw the textures, but reveal them, so multiple strokes of the brush will continue the pattern from the previous stroke, not create a new strip. Brushing a second texture over the same spot won't combine the two, but the new texture will replace that portion of the first one. The textures include things like pinstripe lines either horizontal or vertical, bricks, cross-hatch lines, dots, and fabric-like textures. They are beautiful, and even offer the choice of a primary and secondary colour against the background colour. One interesting effect demonstrated by the program is cross-hatch white lines against a blue sky, providing very effective clouds. This is a medium-resolution program using Commodore's multicolour mode, which offers more colour but less resolution than standard hi-res mode.

Besides the accuracy and placement problems, the plastic light pen has a shielding problem. Static electricity normally found on your screen will be picked up by the pen, and cause strange effects. Every third or fourth line I drew would include a spike. While drawing across the screen, suddenly the line would zip up about ten centimetres, and back down, continuing with my line. I can't begin to describe the feeling this gives about the fifth drawing it ruins. The manual is brief, and not very detailed. Instructions *are* included for using the drawings in your own games and programs.

Given all the problems, I find that this program is more like a set of 'High-tech Crayons' than a serious drawing program. This is a pity, since the actual program is brilliant. Its only flaws are caused by the light pen. If only Futurehouse would include the provision of using a joystick and cursor along with the light pen, it would have a real winner. □

Write Now!
from Cardco
Word processing program for the Commodore 64

Review by Michael Quigley

Cardco's **Write Now! 64** word processor is very similar to their like-named program for the VIC 20, which I reviewed in the May 1984 issue.

Some improvements have been made, including:

- An Insert Mode, which makes it much easier to put copy in the middle of previously typed text.
- An 80-column preview mode, useful for correcting errors in formatting before text is sent to the printer.
- The choice of four different screen/text colour combinations.
- The ability to generate multiple copies.
- A 'mail merge' feature that allows you to generate form letters with different names, addresses, etc.
- A non-destructive directory that pauses with the **CTRL** key.
- Four 'help' screens listing most of the program's formatting commands.

Some of the VIC 20 version's features have not undergone a successful transition. The joystick is supposed to be able to scroll copy, but it doesn't work in either C-64 port. Copy is saved with function key **F3**, and loaded with **F4**, which can lead to disaster. And the program's tab positions are only across the 40-column width of the screen.

There are a few bugs, as well. A line beginning with a period won't be printed, because the program recognizes it as a 'dot command', used for sending special information to the printer. The at-sign ('@') should not be used when saving copy to tape, since it is used as an end-of-file marker, and any copy after it will be lost. And, when using tape, prompt messages sometimes don't appear, since they are the same colour as the screen.

These problems aside, **Write Now! 64** is easy to use, has excellent recovery from operator errors and is reasonably friendly. Like the VIC 20 version, I recommend it for the user who needs a good 'basic' word processor. □

HomePak

from Batteries Included

Integrated word processing, data base management and communications package for Commodore 64

Review by John Easton

***HomePak**, by Russ Wetmore, Star Systems Software, Florida. Price: $49.95 (US), $69.95 (Cdn.)*

HomePak, the latest of Batteries' growing list of 'homeware' (for a growing list of computers), is a remarkably powerful, unusually 'friendly' package — combining at a reasonable price three of the most-used home-computer applications. There is no spreadsheet in the package, perhaps because one is available in another fine Batteries Included release, CalKit.

Since **HomePak** is aimed to a great extent at the 'beginning home user', I decided that there was no better way to test it than on my fearless (sounds safer than 'computer klutz') wife, Muriel. Until we got an SX 64, Muriel hadn't really shown much interest in computers at all. Now she's trying her hand at Data Base Management, courtesy of **HomePak**.

HomePak is user friendly (perhaps even idiot friendly), and well-documented with over 60 pages of text and illustrations. The documentation makes easy reading, and tutorials lead one painlessly through the various packages. Well, *almost* painlessly. Muriel had a dickens of a time trying to find keys labelled (as the text indicated) **CTRL B** and **COM +** (remember, she's a real novice) and I left her absolutely alone to make this a fair test. Finally, once I translated those cryptic labels, she was away.

HomeText, the word processor, is definitely not **PaperClip**, but for simple use and handy help windows, it certainly is friendly. **HomeText** files are saved in a non-standard format, but a utility program is supplied that will translate the files to ASCII or PET ASCII, and vice versa. This should ensure compatibility with nearly all other word processors.

Screen formatting is fun to watch. Instead of wrapping around the right screen margin, lines are broken at a word boundary for easier reading. And speaking of screen formatting, should one desire to get some idea of the ultimate printer output, page by page output to the screen (in hi-res dots and dashes) will provide a vague picture of how the text will look when it finally is sent to paper. Text is stored in a 13,728 byte buffer, which compares quite favourably with the capabilities of most other C-64 word processors.

HomeFind is a very unusual, query-oriented data base manager. Those used to 'proper' DBM programs will find this one maddeningly slow, but for my wife — who up to now had thought DBMS was a new type of toothpaste — it makes absolute sense. It operates almost like the venerable *Animal* guessing game, in that as you tell it new information, it remembers it — complete with friendly questions and prompts.

Now, **HomeFind** obviously knows what records, fields and keys are, but it never lets on to the user. As a matter of fact, it never discloses *anything* to the user about its files — they seem to be private. A directory of the data disk reveals only its apparent emptiness — 664 Blocks Free at all times — a mite dangerous unless this disk is prominently identified. A quick attempt to see what was *really* on the disk gave me the strange reply that it started at track 75. Not being into Disk Doctoring, I leave it at that. It is possible to make a readable (at least by **HomeText**) file from within the program — but this mainly gives one files linked to query requests, perhaps useful for something like merging addresses into a form letter.

Let's look at how queries work. There are three parts to every bit of information one wishes to record. 'Subject' refers to your main reference (and would probably form Key Field 1 in your average DBMS); 'Tag' forms a linked reference to your subject; and 'Object' is for whatever information you wish to preserve that pertains to the previous two. These three parts cannot be more than 80 characters each in length, and the entire record (oops — make that 'line' or 'entry' — remember this is a *friendly* program) cannot be more than 144 characters.

How about an example — straight from the manual? So you want to keep a list of birthdays and other important stuff about all of your friends?

We'll just start 'entering' the information:

mike's birthday's april 18, 1958
mike's favourite colour's blue
mike's car's a honda
mike's phone's 929-9909 (home)

. . .and so on.

During each new entry, your computer will either confirm or deny that this is "news to me!", allowing both corrections and new information to be entered at random. If, in the case of a correction, we were to enter **mike's favourite colour's red**, the computer replies with, **mike's favourite colour was blue. add or change? a/c.** At which time you may proceed accordingly, or simply touch **RETURN** to abort the operation. Such interchanges are encouraged by friendly messages from the computer like **never mind!**, and **thanks!**

Now, for information retrieval. Ask **who's mike?**, and you'll get the whole kaboodle back. Narrow it down to **what's mike's car?**, and voila, **mike's car's a honda**. You can even shorten queries to single words: **what's car's?** or merely **car** — and you'll get a complete list of all the cars the machine knows about — including **mike's car's a honda**.

Now, if you've been watching closely, you'll notice the secret of these entries — the magic delimiter is **'s**, which sometimes tends to strange grammar, but to Muriel's way of thinking is perfectly sensible.

HomeTerm is a very complete terminal package, bilingual in ASCII and C-64 dialects, with built-in Vidtex (Compuserve) and X-Modem transfer protocols. Status displays, and even a real-time clock, are constantly available for reference, and a complete set of macro commands is no more than a control key away at all times. Of course, one may upload or download programs or text via built-in buffer transfer to disk or printer. An added feature, which won't show its real value till you're involved in on-line conferencing, is the ability to split the screen, allowing incoming conference messages to scroll merrily above you, while you carefully compose a brilliant reply in your own workspace. A very complete introduction to using **HomeTerm** on the Compuserve system is included as an appendix.

Summing up, **HomePak** is a fine collection of useful and *friendly* programs! In most cases, an entire help menu is no further away than the nearest function key. Now, if I could only get Muriel away from the SX long enough to write this review! Have I lost a computer or gained a wife? Thanks to Batteries Included and Russ Wetmore for a completely enjoyable product!

PS: A (rumoured) minor bug in the X-Modem transfer protocols (necessary for most up/download of program files on CP/M type boards and Delphi) has been fully corrected by Russ Wetmore, **HomePak**'s author. If you have an early version of **HomeTerm**, and have experienced difficulty, contact Batteries Included (or this writer) for the 'real stuff'! □

Wiztype
from Sierra On-Line
Typing tutor
for the Commodore 64

Review by Marya Miller

Put it this way: this is the first typing tutor of any nature that I have been able to stick with. In other words, I have finally learned to touch-type, after (blush) ten years of business and literary experience.

Years ago, I taught myself to type using seven or eight fingers and telepathy. I was pretty pleased with myself at the time, but soon discovered that I could never type faster than 45 words per minute, using this system. I tried correspondence courses, night school courses and manuals, in a desperate attempt to learn to touch-type properly, all with the same result: I soon slipped back into my old seven-fingered habits.

Then (as the commercials say) I discovered **Wiztype** (lured by my enjoyment of the *Wizard of Id* comic strips). I can now touch-type at 60 words per minute and climbing ...

It's actually fun learning to type with the Wizard, the evil Spirit and Bung the Jester to help you. The colour and graphics are good, the sound effects very appropriate, and the entire package skilfully laid out and presented. Sierra has put a lot of care into **Wiztype**, and it has paid off.

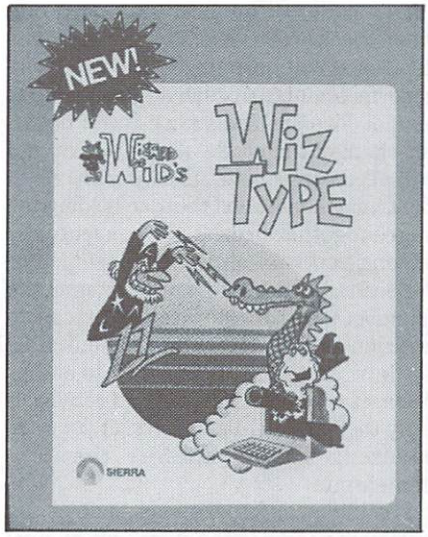
Using **Wiztype** is more like playing a very good game than learning a skill — whether you are just practising drills, words or paragraphs, or playing The Game itself.

The Game is for real. Your score is entered onto the disk each time you tackle it, and every time you load **Wiztype**, you see a graph showing your progress at The Game (and therefore your progress at learning to touch-type).

There are lots of nice little extra touches to this package. I'll only tell you one, though, so you can have the fun of finding out the others yourself. When you successfully complete a level of The Game, the Wizard and the Spirit tell each other corny jokes — using cartoon balloons, of course — while your score is being put on disk.

Incentive, while learning with **Wiztype**? Well, for one thing, it doesn't do you much good if you try to type using any other system — it's set up so that touch-typing is the best way to go. And if you (shudder) fail to type in your words correctly within the time limit (while drilling *or* playing The Game), the Wizard gets an agonized look on his face and the Spirit begins to gloat ... Oops. Within seconds, the evil Spirit has promptly fried the Wizard to a little black heap of cinders. Definitely fun.

Maybe if my typing teachers had been able to fry me into a little black heap of cinders, I'd have learned to touch-type long ago. □



Sky Travel
from Commodore
'Home Planetarium'
for Commodore 64

Review by Nick Sullivan

On May 28, in the year 585 BC, the warring armies of Alyattes of Lydia and the Mede Cyaxeres witnessed a rare event — a near total eclipse of the sun. As the moon's shadow swept across the battlefield, the frightened soldiers abandoned the hostilities, and the warring kings were compelled to make peace. Predicting the eclipse also advanced the reputation of Thales of Miletus, the philosopher, statesman, businessman and mathematician known in later years as one of the Seven Wise Men of ancient Greece.

A couple of hours ago I travelled to the Aegean for a quick look at the sky on that fateful afternoon, and saw the eclipse for myself. It was around five o'clock — already the sun was nearing the horizon. Soon after five, the moon began to encroach on the sun's disk, taking a small, crescent bite from the sun's lower right quarter. More and more of the sun was covered from that point on, except for a short time towards five thirty when the motion of the eclipse appeared briefly to reverse itself. The time of maximum eclipse began some fifteen minutes later. At this point, the sky was quite dark — it was easy to imagine that Alyattes' and Cyaxeres' men might been hampered in skewering each other on account of visibility, in addition to their superstitious awe. Quite a long time later — fifteen or twenty minutes, perhaps — the moon began to slide off the sun's right shoulder, and the sky gradually lightened again. By six thirty or so, the show was over.

I do not know whether conditions were actually dark enough in ancient Greece for Thales and his contemporaries to view the stars and planets while the eclipse was in progress. I myself had no trouble with the observational conditions, and I was able to see at a glance that three planets — Mercury, Mars and Jupiter — were grouped very close together just a little above the sun and to the south; Saturn was somewhat higher in the sky, but still nearby. Stranger still, the remaining four planets — Venus, Uranus, Neptune and Pluto — were at the same time huddled together in an even tighter group about thirty degrees below the horizon.

What do you need to take a journey such as the one I have just described? A time machine? A doctorate in mathematics? A planetarium?

Not any more. All you need nowadays is a Commodore 64 system and an inexpensive program called **Sky Travel**, written by Frank Covitz and Clif Ashcraft, and distributed by Commodore.

We have all looked up at a starry sky at some point in our lives, and felt awe at the infinitude of stars, and the limitless depths of the universe. Multiply that by the vastness of time, from ten thousand years before Christ to ten thousand years after, and multiply that by the expanse of the earth's surface. This is the territory that **Sky Travel** brings home to your Commodore 64.

It is a long and complicated program, heavily computational, and the data files in which it stores the universe occupy most of a disk. The math required to display and to scroll a hi-res screen full of celestial bodies is frightening — the sort of thing you would expect could only be done by programmers who were half computers themselves. Yet **Sky Travel** is friendly, sometimes even witty.

You select (by cursoring around on a Mercator map) your vantage point on the Earth's surface, fine tuning the latitude and longitude with on-screen readouts if you desire. You choose your moment in history — year, date and time of day — with an easy-to-use numerical display. A few seconds of calculation tick by, then

the sky appears.

The major constellations of stars are marked with lines in the traditional way, and their names are given, though you can turn off the lines, the names or both, with a couple of keystrokes. The planets are shown as symbols, rather than star-like dots of light, for easy identification, but you can easily cancel that feature also. If you want to see deep-space objects, such as other galaxies and nebulae, you can call them up. If you want to find a particular planet or constellation, it is done in an instant. If you want information about an object, it's yours in a moment. You can change the rate of passage of time, or its direction. You can adjust your viewing angle to suit your needs, from wide angle to telescopic. There is a chart mode, a printer option and a facility for tracking the planet of your choice as it wanders through the sky.

And there is a fly in the ointment. I saw **Sky Travel**, including the excellent manual, on sale at a computer fair recently for twenty Canadian dollars. Unfortunately, there were no copies remaining. Nor, as far as I can tell, is the program yet available anywhere in Canada, despite its having been in Commodore's hands for a full year. Now that Commodore seems to have decided to give up the software publishing business altogether, the future of **Sky Travel** for the C-64 seems uncertain. At the same computer fair, by the way, the program was being demonstrated for Atari under the name of **Home Planetarium**.

Anyway, keep your eyes open. **Sky Travel** could find its way into your favourite computer store. If you are even a little bit interested in astronomy, snap it up. It is a unique program, and you should have it. □

The COMAL Handbook (Second Edition)
by Len Lindsay
Reston Publishing Co.
467 Pages
Price: $25.95 (Cdn.)

Review by Donald Dalley

In the Commodore world, there are few examples of legitimate copies of any program spreading faster and more widely than the COMAL 0.14 language has done. The original 'COMAL SAMPLER' disk for the C-64 is freely copyable; as a result, most user group libraries include it. This successful ploy teased thousands into at least giving COMAL a chance. Within *one year*, an enormous number of public domain programs were in circulation!

The first edition of the *COMAL Handbook*, covering the earliest versions of the language, was the main reference to guide these ambitious programmers. This edition was helpful, informative, and necessary to read, but there were some problems and additions requiring an updated edition.

Two major differences stand out between the two editions. The new book has about a third more pages and the spine is perfect-bound, instead of plastic-hinge-bound. Two problems arise from the latter change: the book will not lie flat and will need controlling, usually with one of the typing hands; and the glue holding the pages together breaks down, eventually leaving you with a mess of loose pages.

Because COMAL is not stagnant, the *Handbook* had to go through extensive revision as well. Except for a mention in the introduction, all references to earlier releases of COMAL have been removed. Only the CBM 0.14 and 2.00 versions are covered. This simplifies the text immensely.

The introduction explains some of the history of COMAL, the standardized Kernal, and version differences. It also goes into details about using the standardized format of the text, special notes for greenscreen CBM machines, and a brief 'getting started' section, among others. You must read the introduction first to understand some features of the language and the book.

The main body of the *Handbook* is dedicated to the description of over 140 keywords arranged alphabetically. Each word has its own page(s) devoted to explaining its use. An improvement here is that each side of the page header has the keyword in bold print for easier page flipping. Next, the category (command, statement, function, operator, and so on) is specified. Some keywords have more than one category, and each is described separately. If the word is part of the Kernal a **[YES]** is given, if not, a **[NO]**. This is handy for those who wish to write programs for other computers. How fully the keyword is implemented, if at all, is indicated in simple code form for both CBM versions. This is useful for those with the cartridge who wish to write programs for the owners of only the disk version, and vice versa. A clear, concise description of the rules for the keyword's use follows, along with notes usually naming limitations or restrictions.

Next are four very useful parts. The syntax is described in *precise* detail giving *all* parameters! If you want to take full advantage of COMAL's power, this is a necessary list. Next, to make things clear, there are one-line examples of the keyword in context. The variations are useful and informative. A sample program further clarifies the use of the keyword in context. To show the results of the program, output from a run is included. This is where one of the differences from the original edition is made apparent. The first edition had typeset syntax, program listings and sample runs. These were clear and easy to read. In the revision, narrow dot-matrix print has been used, reducing legibility.

The importance of the book lies in its design and usability. You can find what you want, when you need it. If you make a mistake, you can find out why your idea was not allowed. No extra words get in the way of what needs to be explained. Len writes the way manual authors should. Like COMAL itself, the second edition of the *COMAL Handbook* is exemplary. □

Traitex 64
from Logiciel Vision Software
French language word processing program for Commodore 64 and disk drive

Review by Gerry Gold

Traitex 64 International *($149.00 with Dictex, $99.00 without Dictex, $29.00 for the English language dictionary. Logiciel Vision Software, 218 Alexandre, Sherbrooke, PQ, J1H 4S7, Canada.*

In Quebec, Jacques Lebrun's adaptation of Professional Software's **WordPro** and the **Traitex** program are synonymous with professional microcomputer word processing. For this reason alone it is a joy to recommend this program, which combines **WordPro** commands with the **Fleet System 2** program. Unlike other word processors, which may write in French but communicate only in English, **Traitex** accepts its instructions *en français*. If you need **WordPro** or **PaperClip** command compatibility, then **RUN/STOP CTRL Y** enables standard English commands in your document.

Traitex has lots of strong features. For instance, it is available with a 200,000 word (no, that's not a misprint) French dictionary to which 40,000 more words (correct again) can be added in a user dictionary. Moreover, almost 300 blocks

remain on the protected dictionary disk. As if that were not enough, the full **Traitex** can also be purchased with **Fleet System 2**'s 70,000 word English dictionary. The French dictionary loads at 5,000 words per minute (from a 1541!) and offers dynamic spell-checking and other features from within the program.

Convenient **WordPro** features, such as extra text, form letters, file merging and on-screen calculations, are all part of **Traitex**, which also offers many unique features. The **Traitex** keyboard on the 64 is convenient, and maintains a high degree of compatibility with versions that run on Commodore PET computers.

Printer options, screen colours, device numbers and printer interfaces are all set up in a convenient install file that offers a wide array of printer choices. It is impossible, however, to change these features while the program is operating.

Cursor movement and on-screen commands are rapid. As with many of its rivals, **Traitex** supports printing to the screen in either 40 or 80 columns. At startup, users can choose an 80 column or 120 column screen that scrolls sideways as text is entered. Personally, I would rather write and edit in 40 columns, as it is easier to catch errors when you see the entire text. Switch to 80 columns when the text is written and edited.

Another minor problem is that **Traitex** retains some of the weaknesses of **WordPro**. A directory, for example, can only be called by erasing the file in memory. A carriage return erases everything to the right of the cursor. There is no online help command. Also, **Traitex**, like earlier versions of **WordPro**, and like **PaperClip**, does not offer word-wrap (words wrap around the screen). Another missing feature is the column manipulation and sorting ability of **PaperClip**.

WordPro-compatible commands are a plus, but do be careful. **CLR** will invoke the command *Mémorise, Endisque, Intercale*. In this program *Mémorise* invokes recall, not memorize. Failure to observe that difference could be disastrous. Also, **Traitex** creates a USR file that is incompatible with standard word processing files used by outfits such as *TPUG Magazine*. I wrote this document with **Traitex**; printed it to disk as a sequential file; read it into **PaperClip** and then edited the quotes and line feeds!

This is, however, the first fast and full-featured French language word processor that I have used. Since this may be what you need or what you have already purchased, do not forget to send your registration card to Vision Software, who support what they sell. □

Adventure Construction Set
from Electronic Arts
Graphics adventure construction set for Commodore 64, 1541 disk drive and joystick

Review by Marya Miller

There are two words I never want to see on my computer screen again — *Accessing Disk*.

Unfortunately, if you use **Adventure Construction Set**, you will see them a lot. You will spend long, wearisome minutes — hours, even — waiting for your computer to stop *Accessing Disk*. Most of the time, too, you can't go off and do something else while all this accessing is going on, because you have to be there to do frequent disk swaps. The only time you can leave the computer at it is when your adventure is finally being constructed. Then the screen tells you: "The adventure will be ready in about 37 minutes" or so.

Before you get to this blessed stage, however, you will have sat through so many disk swaps and accesses that your thoughts are vaguely occupied with penance, suffering and vegetation. There are lots of disk swaps and accesses merely to load the sample adventures.

The sample games *are* nicely presented, and **Rivers Of Light** and **Deep Dank Dungeons** in particular are fun to play; but when it comes to a 'tutorial', the manual has one on how to play these sample adventures, rather than on how to construct one yourself. Oh, it *is* full of information pertaining to constructing a game, but it doesn't start at A and end up at Z. It kind of flings you into a maelstrom of enthusiastic facts and leaves you to flounder your way out again.

After many hours and many odd things happening, I did manage to create a reasonable adventure game, but the process is rather like Russian roulette, if you let the **ACS** disk have anything to do with it. You end up with regions, characters, objects and pitfalls that you didn't expect to see in your game. Which can be rather fun, mind you; though bewildering to simple souls like me.

If you do everything yourself, the creation part is even more fun but, if you're anything like me, it won't work properly and you'll end up biting your table leg and whimpering (after all those gruelling hours of labour).

The packaging is lovely, and the illustrations in the manual are pleasing. The disk label gives the command line for loading the program — a minor point, but helpful, when you have the instructions for forty-odd different disks rolling about your mind. The graphics are okay — though the characters are a bit small and hard to see — and the available music and sound effects are excellent, but put it this way: I'll let you know how **Adventure Construction Set** and I are getting on in about six months or so.

Fun, if you can stand the length of time everything takes and the user inactivity, but definitely not for neophytes. □

Adventure Writer
from CodeWriter
Text adventure construction program for the Commodore 64

Review by Marya Miller

Unlike **Adventure Construction Set**, **Adventure Writer** actually claims on the front page of its manual: "Your own program — the first time you try!"

Bah! Phooey! Humbug! It's table-leg-biting time again.

There is a sample adventure for you to play and one to construct. This latter comes in place of 'Steps from A to Z'. The idea, I think, is that by constructing the adventure along with the book, you learn to do it *by experience*. An excellent idea, and one of the best ways I know of learning but — alas! — it bombs out, here.

I dutifully did the sample game construction; got foggy about the status table and vocabulary action table, and did it again. Aha! Excellent! Understood it perfectly. Then I mapped out a five-location game of my own, very carefully, and tried that. I got hopelessly mixed up on the status and vocabulary action tables. Several hours later, I finally got things straight, but I certainly wouldn't have done so if I hadn't already had some experience in writing adventure games in BASIC.

I liked **Adventure Writer** better than **Adventure Construction Set** because, for one thing, there were no endless hours of disk swapping. There is more user participation here, too.

This construction set is handy if you want to save a lot of time on standard programming sections, but I found that BASIC was still my best bet, if I wanted to do anything very imaginative.

Not only that, but BASIC was also a heck of a lot quicker! □

Zork and Enchanter

from Infocom

Text adventure games for the C-64

Review by Michael Bonnycastle

If you've only got twenty minutes or so to play one of these games, *don't*. These programs are an experience, and if you get involved with them, you'll be up till all hours of the night. To truly figure them out will take sixteen to forty hours or more of elapsed time. Fortunately, you can save a game part way through, and return to it later. You'll need to do this several times, in order to progress through an entire episode. Sometimes, they can be extremely frustrating, and you'll think you're stuck, so you save the game and go to bed. At 4 am you'll wake up and think, "I haven't tried *this*... or maybe *that* approach will work..." — and sure enough, you're off and running again.

When it works, it's a terribly rewarding experience, and even more so if you can work out the problems yourself. Sometimes, however, a simple hint from a friend who's been there will get you going again, when you've been stuck for several days; or you may return to a section that you haven't visited for a while and — because you've solved a problem somewhere else, or you've read a page from a book at the other end of the game — the solution to this section suddenly comes to mind like a stroke of genius. These games are a fascinating experience, and must be tried to be believed.

Written by Infocom's principal authors who were involved in writing the original **Adventure** (see Jim Butterfield's article elsewhere in this issue), these programs are much more sophisticated, clever and involved than **Adventure**. (The original **Adventure** is on TPUG library disks (C)M5 for the C-64, and (O)G7 for the PET.) **Adventure** allows you to move in various directions, pick up objects and throw axes at nasty dwarfs. **Zork** allows you to interact with a pirate, and **Enchanter** has you teaming up with a not always cooperative adventurer from another game. In **Enchanter**, your success depends not on acquiring treasure, but on your ability to find and use spells to defeat the evil wizard, Krill. In fact, to succeed at **Enchanter**, you must use your understanding of avarice in the adventurer.

Purely and simply, these games are puzzles. You must figure out how to succeed, using the tools available to you. With **Zork**, you may get things, drop things, attack nasties with a sword or knife, turn knobs, use wrenches and screwdrivers, and there is even an inflatable raft that you can blow up with a hand pump. You can get only so far using simple commands, but to advance to the end you must figure out how to open the floodgates on the dam and drain the reservoir, as well as performing other complicated, interrelated tasks with a series of tools. The sequence of events surrounding obtaining the diamond is as bizarre a set of moves as I have ever seen, yet they all make sense.

In **Enchanter**, the spells you need in order to win are found on scrolls and can be 'learned' when you need them, but once you cast them, they are lost from memory. Some spells can be written into your spell book first (using the GNUSTO spell) — these you can use several times. Other spells are too complicated for you to GNUSTO, and so can only be cast once. There are several treasures in this episode (such as an incredibly ornamented and bejewelled egg), but they have other, more important uses than that of mere treasure. You must figure out how best to use them. My favourite portion involves a series of passages within the depths of the castle, a map and a magic pencil. When you draw on the map with the pencil, a passage appears; by erasing a line on the map, the corresponding passage will vanish. This is very useful in avoiding a particularly nasty monster.

Other games in the series that I have *not* played yet include **Zork II**, **Zork III** and **Sorcerer**. I am anticipating many hours of enjoyment when I tackle them, but I do not want to start them prematurely. I will wait until I have lots of time to devote to their particular type of entertainment.

These games are not for the faint of heart, and they require a lot of staying power to get through. I don't recommend them for youngsters, as they require a fair bit of sophistication to solve (although my 12 year old enjoys wandering around them without really being able to solve the entire puzzle). Infocom's games are well worth the time and money you invest in obtaining them and figuring them out. Good luck! □

Products Received

The following products have been received by TPUG Magazine *in recent weeks. While we do not have space to review them in this issue, we would like to describe them briefly in order to bring them to our readers' attention. Please note that these descriptions are based on the manufacturers' own announcements, and are not the result of evaluation by* TPUG Magazine.

Fantastic Filer

Fantastic Filer from SourceCode Technologies, 424 East John Street, Lindenhurst, NY 11757, distributed by Micro-W Distributing Inc., P.O. Box 113, Butler, NJ 07405. Price: $29.95 (US).

Fantastic Filer is an all-purpose data filing system for the Commodore 64 and 1541 disk drive. It allows the user to create an average of 1000 records per disk, each record containing up to 256 characters, with a maximum of 50 fields. Records can be edited, updated or deleted. The user can search for records by record number, index match or by specific search criteria.

A five function calculation routine is built into **Fantastic Filer**, and the program will automatically perform the calculations between fields specified by the user when creating the original layout of a record.

This new version of **Fantastic Filer** includes another program called **Fantastic Forms**, which enables the user to generate columnar reports and prepare mailing labels.

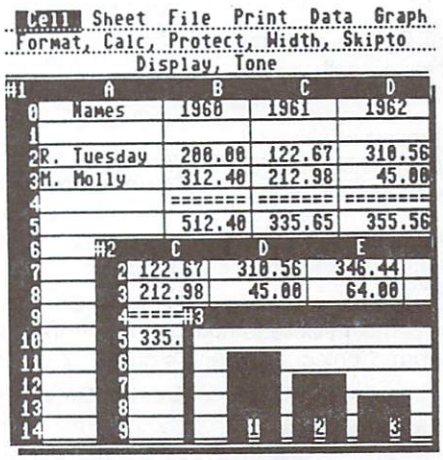
VizaStar

VizaStar from Viza Software, Brompton, Kent, UK. Distributed in Canada by The Soft Warehouse, P.O. Box 1983, Winnipeg MB, Canada R3C 3R3 by direct mail only, and in the US by Solid State Software, 1125 E. Hillsdale Blvd. Suite 104, Foster City, CA 94404. Price: $189.95 (Cdn.); special introductory offer to TPUG readers $179.95 (Cdn.). US price: $119.97.

VizaStar is an integrated Spreadsheet, Database and Business Graphics program designed to "...bring **Lotus 1-2-3** power to the Commodore 64". Created by the UK's Kelvin Lacy, **VizaStar** has sold well in Europe since July 1984, and is now released in North America.

A user-friendly menu system makes **VizaStar** easy to use. The 64,000 cell spreadsheet permits the opening of up to 9 simultaneous 'windows', to overlay distant sections of the spreadsheet onto the working screen. Access between the Spreadsheet, Database and Graphics programs is immediate, and data can easily be moved between these integrated functions. A powerful execution facility allows the user to 'program' any routines necessary to automate their specific application for **VizaStar**. Data storage and retrieval on disk is provided, with immediate interrupts and directory searches possible at any time.

VizaStar is written in 100 per cent machine language. It is totally resident in memory during use, and is compatible with many printers and interfaces. Direct output of printer control commands gives



VizaStar screen with several windows.

even greater flexibility. Import/Export routines also make it possible for **VizaStar** to exchange information with **Easyscript**, **WordPro**, **PaperClip**, **HES Omniwriter** and many other programs.

C-64 Encyclopedia

The Commodore 64 User's Encyclopedia from The Book Company, 11223 S. Hindry Avenue, Los Angeles, CA 90045. Price: $14.95 (US).

The Book Company has published a whole series of encyclopedias for different computers, including this one for the Commodore 64. According to the authors, the *Encyclopedia* "...does not offer detailed coverage of the internal electronic details of the C-64. It does, however, provide a complete reference on BASIC programming, general operation of the computer and its accessories, and available products." The entries in the book cover the following aspects of using the C-64, and computing in general:

- Definitions of general microcomputer terminology, concepts and abbreviations.
- Explanation of BASIC programming, including BASIC statements and commands as separate entries.
- Information on the internal structure and operations of the Commodore 64.
- Brief descriptions of other programming languages: FORTRAN, COBOL, Pascal, etc.
- Description of software packages and hardware accessories.

C-64 Programs for the Home

Commodore 64 Programs for the Home by Charles D. Sternberg, published by Hayden Book Company Inc., 10 Mulholland Drive, Hasbrouck Heights, NJ 07604. Price: $15.95 (US).

This book is intended for two groups of users: first-time buyers who don't want to spend extra money on software, and those who would like to learn BASIC programming by typing, running and modifying program listings to suit their needs.

There are forty BASIC program listings in eight chapters. The programs cover basic household needs in financial planning, recording and retrieval of payments, telephone numbers and maintenance costs, as well as in expense planning, and helping children with schoolwork. To close the list, there are also utility programs relating to outside activities and recreation.

Each listing is preceded with a description of the program, instructions on how to use it and how to enter the data, notes and things to check. Then follow the program listing, sample data, a sample run and a major symbol table. All programs are designed to run using the datasette. However, they can be easily modified by those who own a disk drive, and wish to store programs on diskettes. To help readers modify the programs, as well as improve them, the author has included *Appendix A — Language Features Used* and *Appendix B — Converting Programs for Floppy Disk Usage.*

Presented by Astrid Kumas □

TPUG Associate Clubs

CANADA

C64 North Bay Users Group (Ontario) meets at Cassellholme on Olive St. on the first Wednesday of the month at 7:30 pm. Contact Matt Vautour 705-474-5692.

Chaleur Commodore Club (New Brunswick) meets at the District School Board, Dalhousie, on the third Wednesday of the month at 7:00 pm. Contact Terry Traer 506-684-4852.

Commodore Owners of Muskoka (Ontario) meets at MacAulay Public School, Bracebridge, on the first Wednesday of each month at 7 pm. Contact Mike Wilson 705-645-6300.

Edmonton Commodore Users Group (Alberta) meets at Archbishop Jordan High School, Sherwood Park, on the last Friday of each month at 7 pm. Contact Bob Kadylo 403-465-3523.

Fredericton Commodore Users Group (New Brunswick) meets in Room 105, Administrative Building second floor, at Saint Thomas University, on the third Wednesday of each month at 7:00 pm. Contact John W. Palmer, Comp 53, Site 15 Castle Acres SS#3, Fredericton, NB, E3B 5W9.

Guelph Computer Club (Ontario) meets at Co-operators Insurance Assoc. on the second Wednesday of each month at 7:30 pm. Contact Brian Grime 519-822-4992.

London Commodore Users Club (Ontario) meets at Althouse College of Education, main auditorium on the third Monday of each month at 7 pm. Contact Dennis Trankner 519-681-5059.

Niagara Commodore Users Group (Ontario) meets at Lakeport Secondary School, St. Catharines, on the first Monday of each month at 7.30 pm. Contact Ian Kerry 416-688-6464.

Sarnia C64 Users Group (Ontario) meets at Lambton College on the first Sunday of each month at 7:30 pm. Contact J.C. Hollemans 519-542-4710.

Saskatoon Commodore Users Group (Saskatchewan) meets in Room 2C02, Engineering Building, University of Saskatchewan, on the last Friday of each month (except June, July and December) at 7:00 pm.

Commodore Users Club of Sudbury (Ontario) meets at Lasalle High School in the cafeteria on the last Thursday of each month at 7 pm. Contact Tim Miner 705-566-9632.

PET Educators Group (Windsor, Ontario) meets at Faculty of Education Building, 600 3rd Concession, Windsor, on the third Wednesday of each month (not July and August) at 7 pm. Contact John Moore 519-253-8658.

Winnipeg PET Users Group (Manitoba) meets at Gordon Bell High School, Room 228, on the first Wednesday of each month at 7:30 pm. Contact W.P.U.G., P.O. Box 4096, Station B, Winnipeg, MB, R2W 5K8.

UNITED STATES

Boston Computer Society/Commodore Users Group meets at Minute Man Tech High School, Rt 2A (just off Rt 128), in Lexington, MA, every second Monday of the month at 7 pm. Contact Harvey W. Gendreau 617-661-9227.

C-64 Users Group, Inc. (Chicago, Illinois): seven chapters meet at 7:00 pm. **Northwest Chicago** (Logan Square) on the first Tuesday; **River Grove** on the first Wednesday; **Des Plaines** on the first Thursday; **Park Ridge** on the second Monday; **Calumet Park** on the second Tuesday; **Westchester** on the second Thursday; **Southwest Chicago** (Garfield Ridge) on the second Friday; **Evanston** on the third Wednesday. For exact locations and changes, contact Darrell Hancock 312-588-0334, or David Tamkin 312-583-4629.

Commodore 64 Owners of Petaluma (California) meets in the multi-use room, La Tercera School on the third Thursday of the month at 7:00 pm. Contact Robert Hermann 707-762-1376.

Commodore Computer Club of Toledo (Ohio) meets at Bedford Administration Building on Temperance Rd., between Lewis and Jackman Roads on the second Friday of each month at 7:30 pm. Contact Jim Cychler 419-475-9160.

Commodore Houston Users Group (Texas): **Clear Lake Chapter** — Nassau Bay City Hall, NASA Road #1, on the first Wednesday of each month at 7 pm. **Central Chapter** — Farrish Hall, University of Houston main campus. **NW Chapter** — Bleyl Jr. High School, 10,000 Mills Road (Cypress-Fairbanks SD), on the third Thursday of each month at 7:30 pm. **Klein Chapter** — Hildebrandt Middle School, 22,800 Hildebrandt Road (Klein ISD), on the third Tuesday of each month (except July & August) at 6:30 pm. Contact Mary F. Howe 713-376-7000.

Commodore Users Society of Greenville (South Carolina) meets at the Greenville National Bank, Community Room, 10 Pleasantbury Drive, Greenville, SC, on the last Thursday of each month at 7.30 pm.

Genesee County Area Pet Users Group (Michigan) meets at Bentley High School on Belsay Rd. on the third Thursday of each month at 7 pm. Contact Gordon Hale 313-239-1366.

Greater Omaha Commodore 64 U.G. (Nebraska) meets at South Omaha campus of the Metropolitan Technical Community College, 27th and Q Streets in Room 120 of the Industrial Training Center, on the first Thursday of the month at 7 pm. Contact Bob Quisenberry 402-292-2753.

Manasota Commodore Users Group (Florida) meets at the Florida Power and Light Building, Bradenton, on the second and fourth Thursdays of the month at 7 pm. Contact Robert O. Bronson 813-747-1785.

MAT-SU Commodore-64 Club (Alaska) meets at the Alaska Computer Systems store, Wasilla, on the third Thursday of each month at 7 pm. Contact Terry Maw 907-376-7508.

Michigan's Commodore 64 Users Group meets at Warren Woods High School in Warren, on the third Tuesday of each month at 7 pm. Call 313-773-6302.

Mohawk Valley Commodore User's Group (New York) meets at the Clara S. Bacon School in Amsterdam, at 7 pm on the second Tuesday of the month. Contact William A. Nowak 518-829-7576.

Mountain Computer Society (Sandy, Utah) meets at Murray High School on the second Thursday and last Tuesday of each month at 7:00 pm. Contact Dennis Senior 801-566-5593, or Don Jones 801-967-6641.

TPUG Associate Clubs

Russellville CUG, Inc. (Arkansas) meets at Oakland Heights Elementary School on the third Thursday of each month at 7:30 pm. Call 501-967-1822.

Sacramento Commodore Computer Club (California) meets at Kit Carson High School, on the fourth Monday of each month at 7 pm. Contact Geoff Worstell 916-961-8699.

S.C.O.P.E. (Dallas, Texas) meets at U.T.D., Erik Jonnson Building (Corner Floyd & Campbell Rds.), in Plano, on the second Saturday of the month at 1:30 pm. Contact Betty Clay 817-274-0709.

Southern Minnesota Commodore Users Group meets at Mankato State University on the first Thursday of each month at 7:30 pm. Contact Dean Otto 507-625-6942.

Tri-City Commodore Computer Club meets at Washington Public Power Supply System auditorium on George Washington Way on the second Wednesday of the month at 7:00 pm. Contact George Carpenter 216-946-7746.

Westmoreland Commodore User's Club (Penn.) meets at Westmoreland County Community College, in Youngwood, on the third Friday evening of each month. Contact Bob McKinley 412-863-3930.

INTERNATIONAL

Baden Computer Club (West Germany) meets at CFB Baden-Soellingen on the second Sunday of each month at 7 pm. Contact Ben Brash.

Trinidad Association of Commodore Owners — TACO meets at St. Mary's College, Frederick Street, Port of Spain, every second Saturday of the month at 2 pm.

Classifieds

For Sale: PET 2001 with 4.0 ROM upgrade, 4040 drives with 4.0 ROM addition, 2022 graphics printer. All cables, excellent shape original set, $700.00. Dave Whitaker 615-478-4538 day, 615-476-6634 night.

For Sale: Info-Design Accounting Package, for 1541 drive, G/L, A/P and I/M, all with backups and manuals. $195.00. 416-655-4003.

Wanted: Modem for PET/CBM computer. Gene Clinger, P.O. Box 491, San Clemente, CA 92672, 714-498-6935.

Wanted: 2031/4040 drive, 4010, modem for 4032-N PET. Used OK. 80 column conversion wanted. Ideas? Marc McSwain, 4513 Acushnet, Corpus Christi, TX 78413, USA.

For Sale: B128, 8050 drive, 4023 printer (all with covers). Also P13 USI Monitor, amber black, plus Precision Software's Superscript and Superbase. Leo Grabowski 717-724-5210. Price: $795.00 (US) LOT.

CBM 8096, 8050, 8023P. Silicon Office, CalcResult, manuals, more. Best offer. Call Ron 416-900-1994, days.

Wanted: Tape or cartridge home application and business software for use on VIC 20. G.B. Swan, Box 127, Stewart, BC, Canada, V0T 1W0 or phone 604-636-2378 evenings.

Plans: for inexpensive build-it-yourself computer cabinet; $4.50/set. Flexible, versatile design will hold C-64/VIC 20, 2 disk drives (1541), printer, monitors, and much more. From A.W. Walters, RR#3, Nelson, BC, V1L 5P6.

For Sale: SuperPET or SuperPET board assembly that I will install in an 8032. Offers. Mike Donegan 416-639-0329.

TPUG Magazine Distributors

Dealers: *If you would like to carry* TPUG Magazine *in your store, you may order from any one of the following distributors:*

CANADA

Micron Distributors, Toronto, ON	800-268-1238
Compulit Distributors, Port Coquitlam, BC	604-464-1221

USA

Prairie News, Chicago, IL	312-384-5350
Levity Distributors, North Hollywood, CA	818-506-7958
Whole Life Distributors, Englewood, CO	303-761-2435
M-6 Distribution, Houston, TX	713-778-3002
The Homing Pigeon, Elgin, TX	512-276-7962
Northeast News Distributors, Kingston, NY	914-382-2000
Fred Bay News Co., Portland, OR	503-228-0251
Alonso Book = Periodical, Alexandria, VA	703-765-1211
Cornucopia Distribution, Seattle, WA	206-323-6247
Guild News, Atlanta, GA	404-252-4166
Micro-PACE, Champaign, IL	800-362-9653

OS/9 IS HERE

TPUG has implemented the popular 6809 operating system 'OS/9' on the SuperPET. Super-OS/9 greatly expands software availability and the hardware capabilities of this computer while at the same time preserving access to the Waterloo languages and programs.

The cost of Super-OS/9 to club members will be $195 (Cdn.), which will include the cost of a hardware modification that will not affect the normal operation of the SuperPET. Two board SuperPETs require a simple hardware modification — we provide instructions. To obtain your copy please send $195 (Cdn.) to: TPUG (1912A Avenue Rd., Suite 1, Toronto, Ont., M5M 4A1, Canada).

What does Super-OS/9 offer?

- A true operating system with UNIX features and the simplicity and command style of Commodore BASIC;
- Multi-tasking and multi-user environment;
- Multi-level directories similar to those available in MS DOS 2.0.;
- Time and date stamp for all directory entries (files);
- File access privileges may be restricted by the owner of a file.

Extensive software is available for OS/9, most of which will run on the SuperPET.

The Super-OS/9 package includes an assembler, editor, command (shell) library monitor, symbolic debugger and 25K RAM Disk.

Available Languages (compilers) include BASIC-09, Pascal, CIS-COBOL, 'C' Language, and others.

Available Applications Programs: Word processors, business, inventory and accounting applications.

Public Domain: Software, relational database and spreadsheets, extensive public domain software and documentation.

TPUG will acquire public domain software and assist users in the conversion of commercial software to Commodore format.

Portability and Expandability

- Super-OS/9 programs will run on all OS/9-based microcomputers.
- Super-OS/9 will run hard disks and parallel drives.
- There will be source code compatibility to versions of OS/9 that are planned for the Motorola 68000.

For information call TPUG Inc. (416)782-8900
(416)782-9252

For technical information

Gerry Gold (416) 667-3159/225-8760

advertisers' index

TPUG Contacts

TPUG OFFICE 416/782-9252, 416/782-8900

TPUG BBS 416/782-9534

Board of Directors

President	Michael Bonnycastle	416/654-2381
Vice-President	Chris Bennett	416/782-9252
Treasurer	Carol Shevlin	c/o416/782-8900
Recording Sec.	John Shepherd	416/244-1487
	Rosemary Beasley	c/o416/782-8900
	Gord Campbell	416/492-9518
	Gary Croft	416/727-8795
	Mike Donegan	416/639-0329
	Bill Dutfield	416/224-0642
	John Easton	416/251-1511
	Carl Epstein	416/492-0222
	Keith Falkner	416/481-0678
	Gerry Gold	416/225-8760
	Rob Lockwood	416/483-2013
	Louise Redgers	416/447-4811
General Manager	Louise Redgers	416/782-8900
Library Co-ord.	David Bradley	c/o416/782-8900
	or the TPUG BBS	416/782-9534
	or Compuserve	ID# 70216,414

TPUG Magazine

Publisher	Louise Redgers	416/782-1861
Editor	Nick Sullivan	416/782-1861
Assistant Editor	Marya Miller	416/782-1861
Ad Director	Louise Redgers	416/782-1861

Meeting Co-ordinators

Brampton Chapter	Garry Ledez	c/o416/782-8900
Central Chapter	Michael Bonnycastle	416/654-2381
C-64 Chapter	Louise Redgers	416/447-4811
COMAL Chapter	Donald Dalley	416/742-3790
	Victor Gough	416/677-8840
Communications	Richard Bradley	c/o416/782-8900
Eastside Chapter	Judith Willans	c/o416/782-8900
	Darren Fuller	c/o416/782-8900
Hardware Chapter		c/o416/782-8900
Machine Language	Garry Ledez	c/o416/782-8900
SuperPET Chapter	Gerry Gold	416/225-8760
VIC 20 Chapter	Rick Adlard	416/486-7835
	Anne Gudz	c/o416/782-8900
Westside Chapter	John Easton	416/251-1511
	Al Farquharson	519/442-7000

Librarians

COMAL	Victor Gough	416/677-8840
Commodore 64	David Bradley	c/o416/782-8900
	Richard Bradley	c/o416/782-8900
French	Baudouin St-Cyr	c/o416/782-8900
PET	Mike Donegan	416/639-0329
SuperPET	Bill Dutfield	416/224-0642
VIC 20	Daryl King	c/o416/782-8900
	Walter Holowatenko	416/782-8900
BBS Sysop (voice)	Richard Bradley	c/o416/782-8900
Assistant Sysop	Sandi Paquette	c/o416/782-8900

TPUG
LIBRARY DISKS

10 months of the latest, fabulous public domain software

ONLY

$79.95* CAN

SAVE 20% or more by using this offer

Subscribe now to start receiving the ***TPUG disk of the month*** *in September*

*US orders US$59.95

Name _____ Membership # _____

Address _____

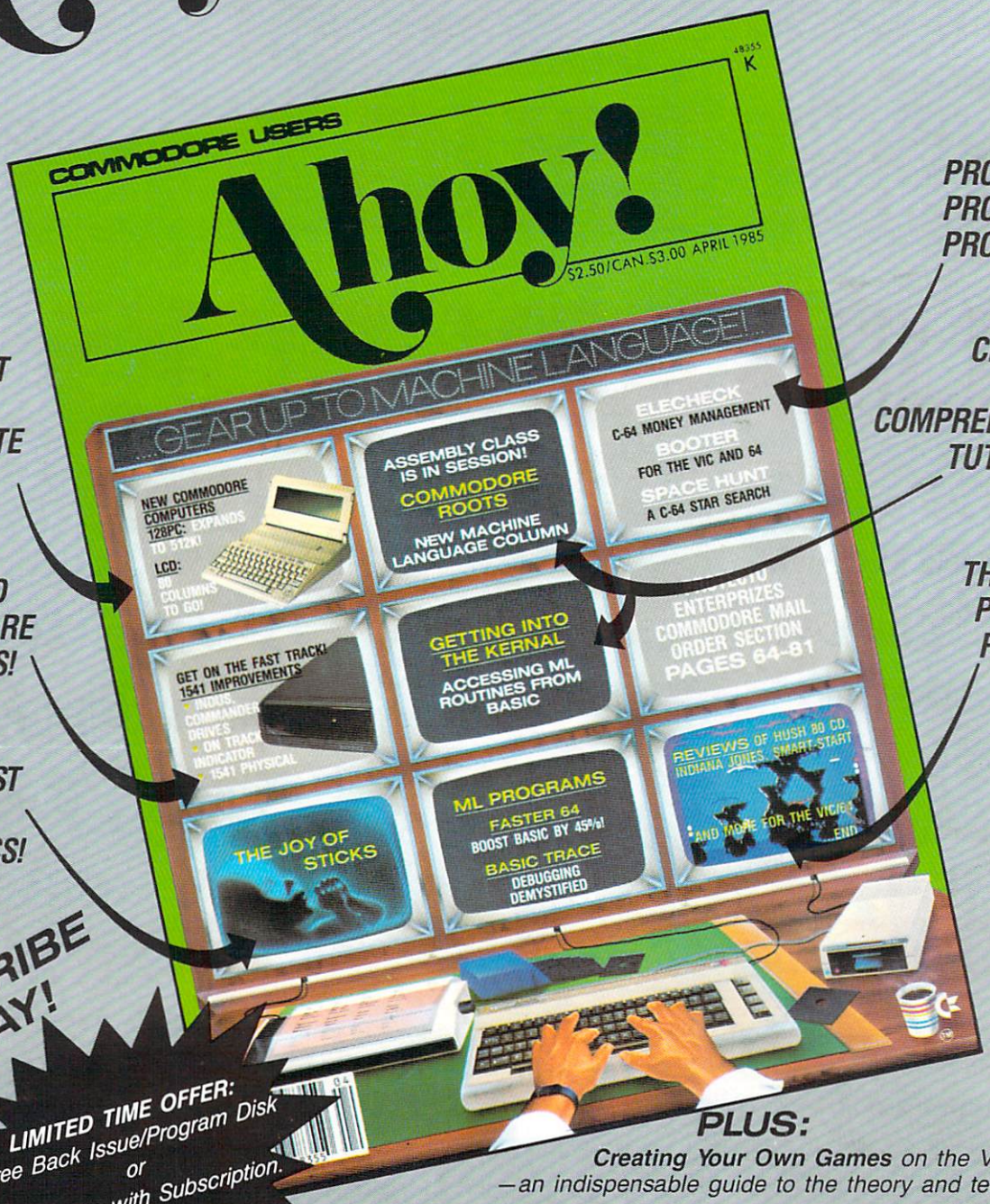
_____

☐ Cheque ☐ Money Order ☐ VISA ☐ MASTERCARD Amount $_____ Currency ☐ Can. ☐US

Credit Card # _____ Expiry date _____ Signature _____

Type of disk: ☐ C-64 ☐VIC 20 ☐ PET ☐ SuperPET (10 disk set)

Ahoy! The *Other* Commodore Magazine



PROGRAMS, PROGRAMS, PROGRAMS!

CLEARER, MORE COMPREHENSIVE TUTORIALS!

THE MOST PRODUCT REVIEWS

MOST CURRENT AND COMPLETE NEWS!

DETAILED HARDWARE ARTICLES!

THE MOST COLOR GRAPHICS!

SUBSCRIBE TODAY!

LIMITED TIME OFFER:
Free Back Issue/Program Disk
or
Cassette with Subscription.

PLUS:

Creating Your Own Games *on the VIC and 64—an indispensable guide to the theory and technique of game design.*

Commodares*—programming challenges to toggle the bits in your own random access memory.*

Ship to Shore*—the latest on the telecommunications front.*

SOS*—your programming maydays answered.*

	U.S.A.	CANADA & ELSEWHERE
❑ 1 year *Ahoy!* subscription (12 issues).....	$19.95	$26.95
❑ 2 year *Ahoy!* subscription (24 issues).....	$37.95	$49.95

❑ My payment (in U.S. dollars) is enclosed.
❑ Bill me.

Please check:
❑ Disk
❑ Cassette

Make check or money order payable to **Ion International.**
Send coupon or facsimile to: ***Ahoy!*** **T-PUG 3**
Ion International Inc., 45 West 34th St., Suite 407, New York, NY 10001

NAME_____

ADDRESS_____

CITY_____ STATE_____ ZIP_____