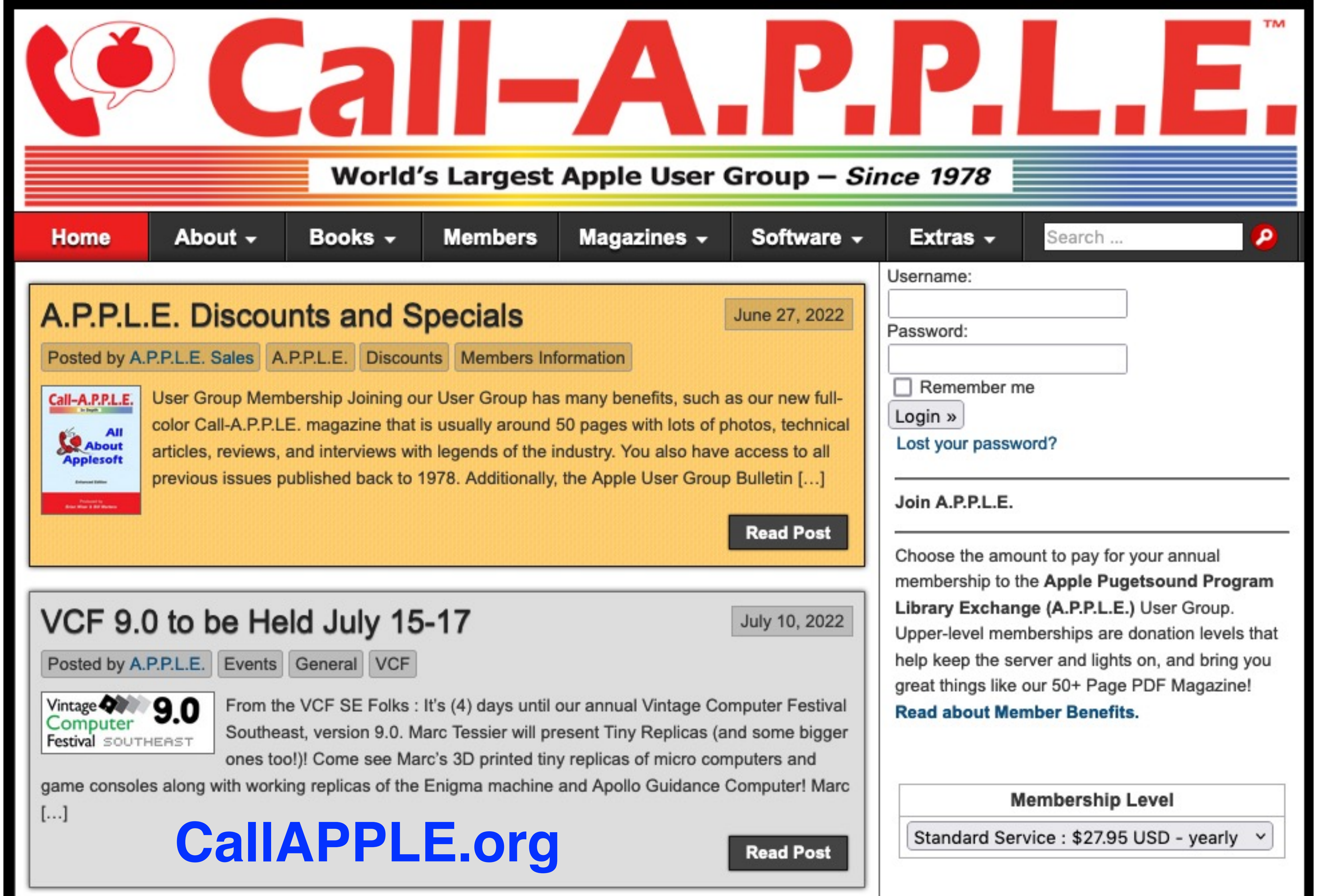
A.P.P.L.E.

Apple Pugetsound Program Library Exchange

www.callapple.org

KansasFest 2022 Update

Brian Wiser & Bill Martens



A.P.P.L.E. Discounts and Specials

June 27, 2022

Posted by A.P.P.L.E. Sales | A.P.P.L.E. | Discounts | Members Information

User Group Membership Joining our User Group has many benefits, such as our new full-color Call-A.P.P.L.E. magazine that is usually around 50 pages with lots of photos, technical articles, reviews, and interviews with legends of the industry. You also have access to all previous issues published back to 1978. Additionally, the Apple User Group Bulletin […]

Read Post

VCF 9.0 to be Held July 15-17

July 10, 2022

Posted by A.P.P.L.E. | Events | General | VCF

From the VCF SE Folks : It's (4) days until our annual Vintage Computer Festival Southeast, version 9.0. Marc Tessier will present Tiny Replicas (and some bigger ones too!)! Come see Marc's 3D printed tiny replicas of micro computers and game consoles along with working replicas of the Enigma machine and Apollo Guidance Computer! Marc […]

Read Post

Username:

Password:

☐ Remember me

Login »

Lost your password?

Join A.P.P.L.E.

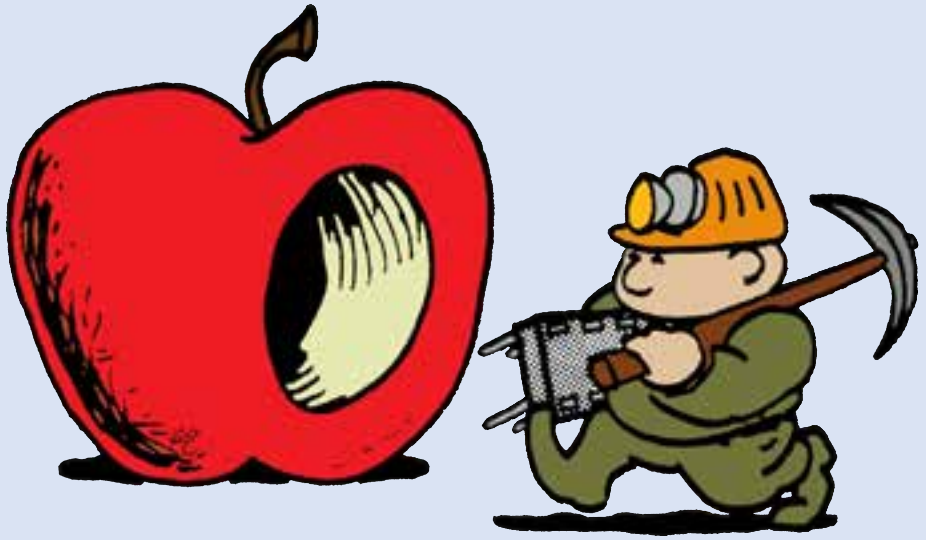
Choose the amount to pay for your annual membership to the **Apple Pugetsound Program Library Exchange (A.P.P.L.E.)** User Group. Upper-level memberships are donation levels that help keep the server and lights on, and bring you great things like our 50+ Page PDF Magazine! **Read about Member Benefits.**

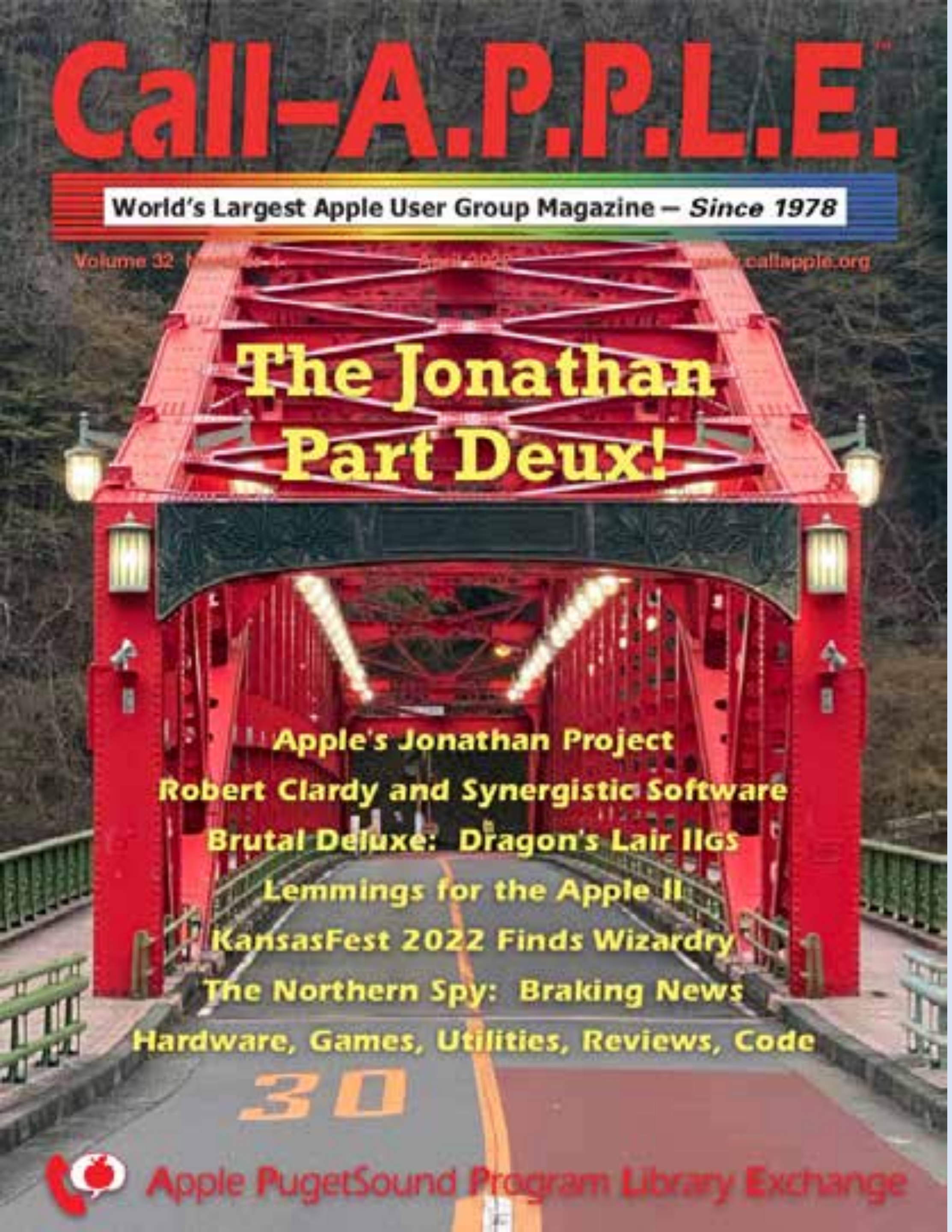
Membership Level
Standard Service : $27.95 USD - yearly

CallAPPLE.org

Membership: CallAPPLE.org/members

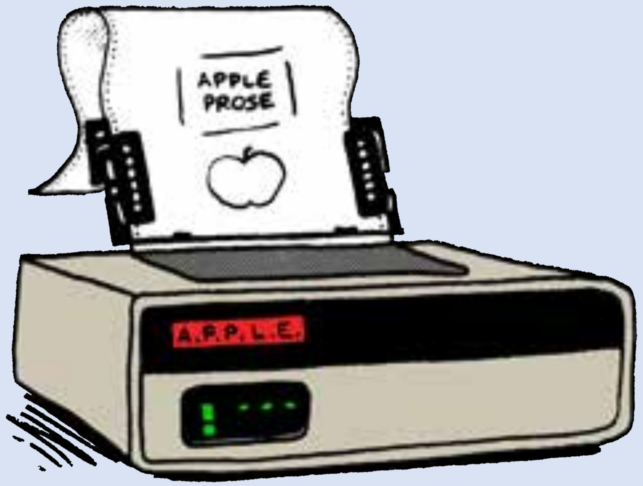
- Official Apple User Group, founded in 1978
- Base-level Membership: $27.95 per year. New “Donate” option.
- Current Magazine & Legacy PDFs
- 3rd Party User Group Discounts
- A.P.P.L.E. Software Library



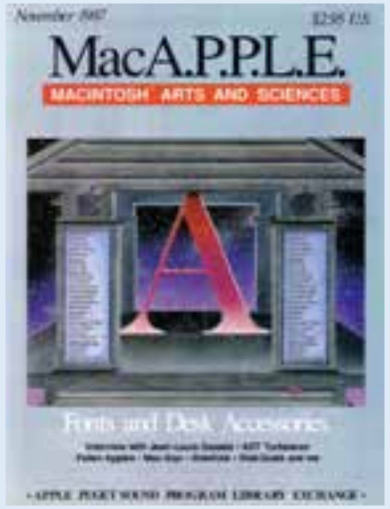

Call-A.P.P.L.E.
World's Largest Apple User Group Magazine — Since 1978
www.callapple.org
The Jonathan Part Deux!
Apple's Jonathan Project
Robert Clardy and Synergistic Software
Brutal Deluxe: Dragon's Lair IIGS
Lemmings for the Apple II
KansasFest 2022 Finds Wizardry
The Northern Spy: Braking News
Hardware, Games, Utilities, Reviews, Code
Apple PugetSound Program Library Exchange

CallAPPLE.org/magazines

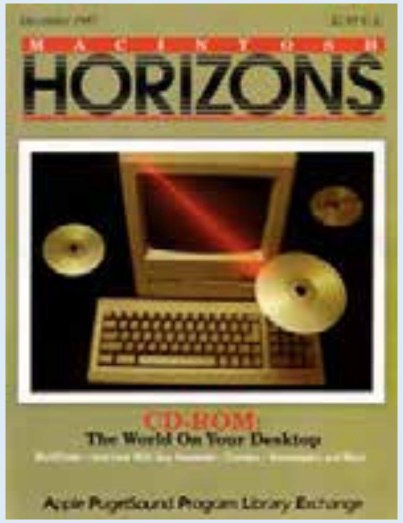
- **40th Anniversary Free PDF**
- Magazine Since 1978. PDF since 2015
- Legacy & New for members
- Now Monthly



Other A.P.P.L.E. Magazines

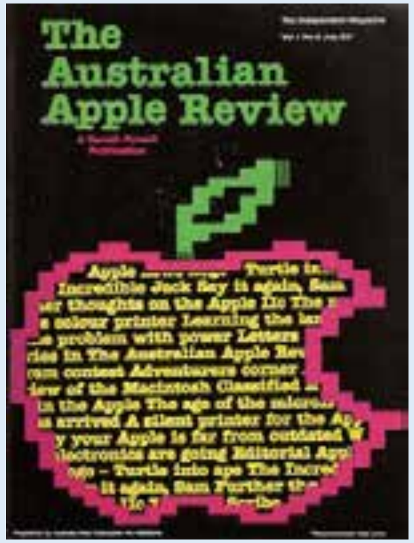


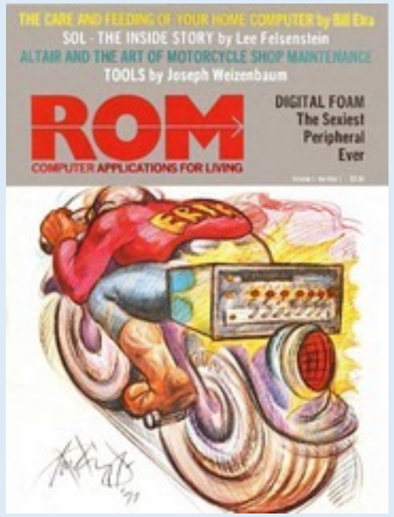
MacA.P.P.L.E.
MACINTOSH ARTS AND SCIENCES
Fonts and Desk Accessories
APPLE PUGET SOUND PROGRAM LIBRARY EXCHANGE



MACINTOSH
HORIZONS
CD-ROM
The World On Your Desktop
Apple PugetSound Program Library Exchange

Related Magazines: CallAPPLE.org/magazines




THE CARE AND FEEDING OF YOUR HOME COMPUTER by Bill Etra
SOL - THE INSIDE STORY by Lee Felsenstein
ALTAIR AND THE ART OF MOTORCYCLE SHOP MAINTENANCE
TOOLS by Joseph Weizenbaum
ROM
COMPUTER APPLICATIONS FOR LIVING
DIGITAL FOAM The Sexiest Peripheral Ever

AppleArchives.com

The Best Vintage Apple & Mac Websites

Repository of Apple & Mac sites, Manuals, Software, Magazines

BEAGLE BROS EST 1980

beagle.applearchives.com

APPLIED ENGINEERING

ae.applearchives.com

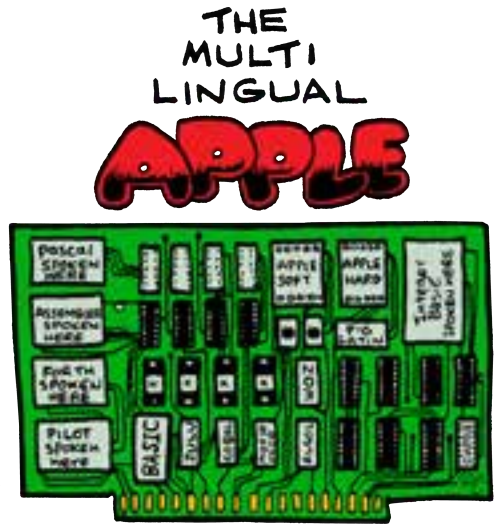
- Largest, Official Archive – 250+ Disks & Manuals
- Contributors: Kenneth Brumbaugh (1st CEO), Steve Taffee (Last CEO), Philip Bouchard (*Oregon Trail*), Wayne Studer (*Oregon Trail II*)
- Enhanced & Redesigned PDF Product Glimpses, Teacher Guides, Student Handouts.

CallAPPLE.org/store

T-shirts
Posters
Mugs…

CallAPPLE.org/store

GOT A PROGRAM TO RUN ON THE FIRST ATTEMPT.



2022 Books

PEEKing at Call-A.P.P.L.E.

APPLE PUGETSOUND PROGRAM LIBRARY EXCHANGE

1978

Compendium

of Magazine Articles

Produced by

Brian Wiser & Bill Martens

PEEKing at Call-A.P.P.L.E.: 1978 Compendium of Magazine Articles represents a nostalgic look back at the early years of the Apple computing revolution through the eyes of the Apple Pugetsound Program Library Exchange (A.P.P.L.E.) user group.

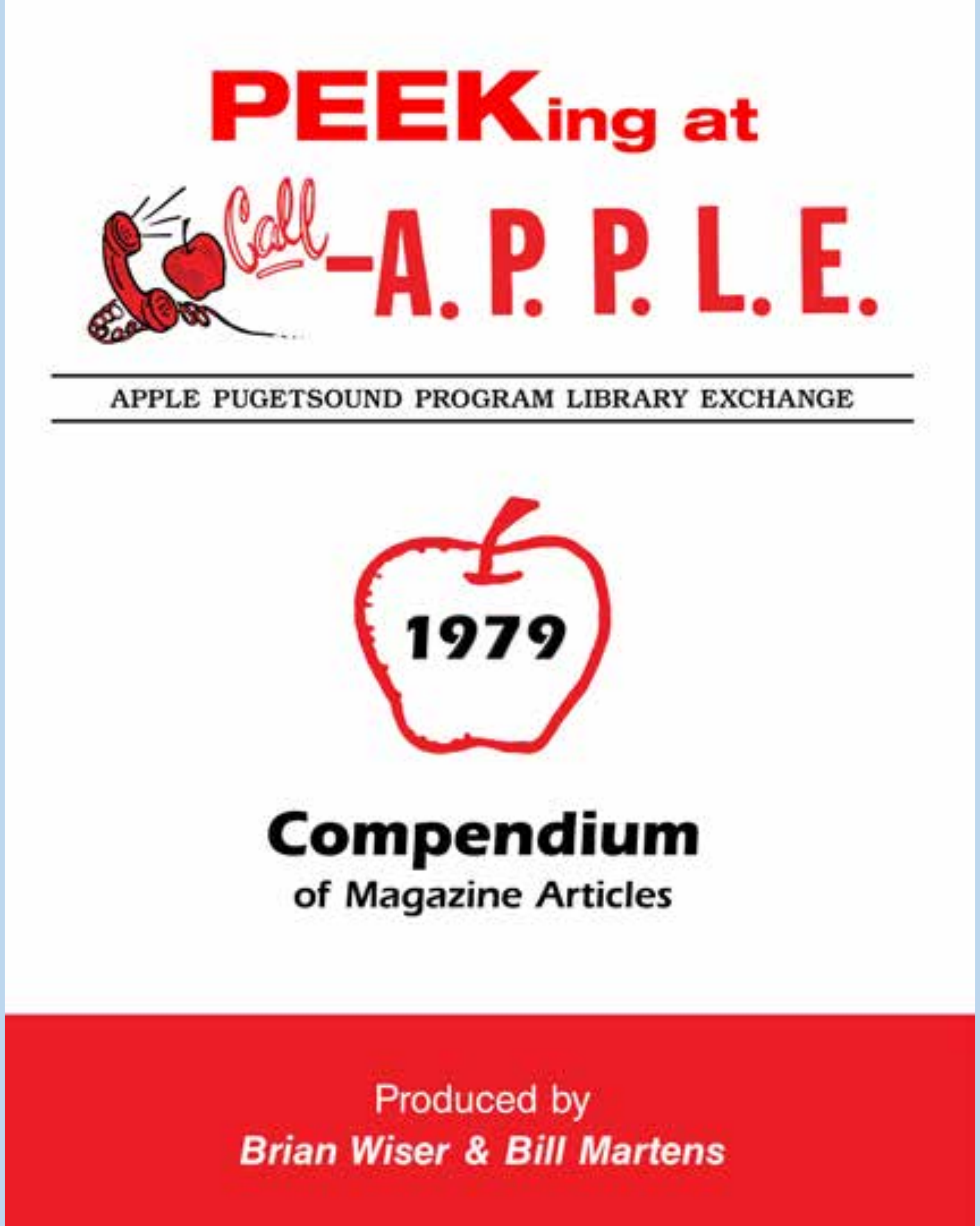
Articles from all 10 issues from the first year of *Call-A.P.P.L.E.* magazine are organized into categories such as: BASIC, Color Graphics, Disk Fundamentals, Printing and the Monitor. Over 100 pages are enhanced and restored, complete with advertisements and program listings. Relive the earliest days of the Apple II computer through the eyes of the creators and users.

Highlights Include:

- Articles and programs from Apple computing legends including: Darrell Aldrich, Ron Aldrich, Val Golding, Bob Huelsdonk, Randy Wigginton, Don Williams, and Apple co-founder Steve Wozniak.
- A brief history of Apple by former Apple Inc. CEO Michael Scott.
- The first documentation of Integer BASIC and Applesoft outside of Apple.
- Reviews of the Disk II, DOS 3.1, Applesoft, the first printer for the Apple II, and many other revolutionary products.

Apple PugetSound Program Library Exchange

www.callapple.org



PEEKing at

Call-A.P.P.L.E.

APPLE PUGETSOUND PROGRAM LIBRARY EXCHANGE

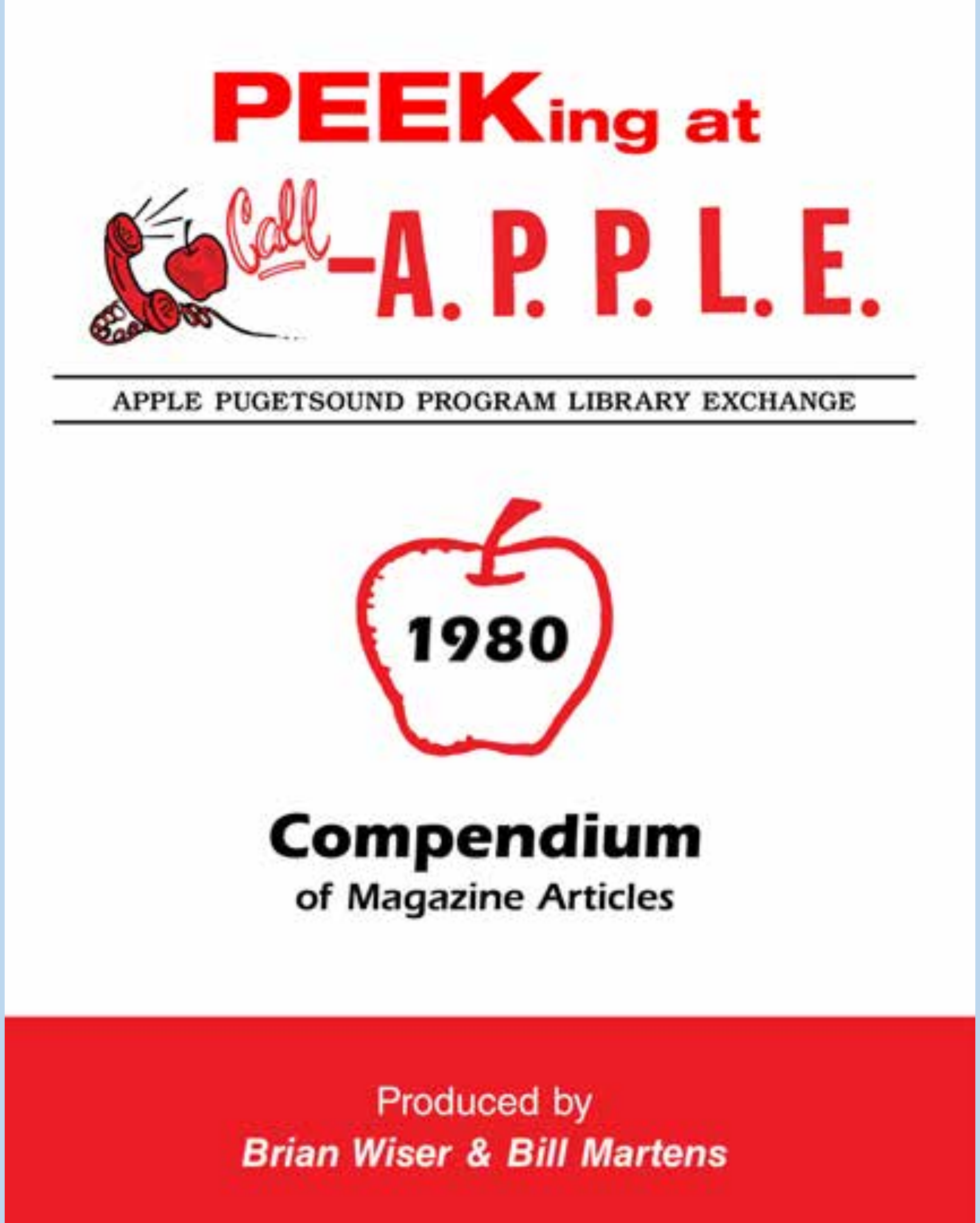
1979

Compendium

of Magazine Articles

Produced by

Brian Wiser & Bill Martens



PEEKing at

Call-A.P.P.L.E.

APPLE PUGETSOUND PROGRAM LIBRARY EXCHANGE

1980

Compendium

of Magazine Articles

Produced by

Brian Wiser & Bill Martens

DIRECTORY TITLE FORMATTING

SUMMARY
A. CONCEPT OF DIRECTORY TITLE FORMATTING (DTF)
B. DISKETTE DIRECTORY (TR $ 11) DATA FORMATTING
C. L.H OF S AND OTHER USEFUL UTILITY ROUTINES
D. TECHNIQUES OF DIRECTORY TITLE FORMATTING
E. SHORT CUTS (ONCE YOU'VE DONE ALL THE WORK)
F. HIGH CRIMES, LARCENY AND OTHER MISDEMEANORS

by Clifton M. Howard, M.D.

The ultimate purpose of DTF is to make your diskette catalog more attractive and useful. An intermediate benefit is to become familiar with some of the better-known Utility Routines (see C.) and in the process learn about the storage and retrieval of data on the diskette. The latter knowledge is an invaluable tool in understanding the structure of the DISK OPERATING SYSTEM (DOS) and the capabilities of your computer. I shall use, as an illustration, one of my diskettes with a collection of "Utilities". Normally the typed command "CATALOG" displays the directory list of file-names (usually programs) (Fig 1). To see the entire catalog usually two video screen scrolls are required. Compare that to the DTF presentation (Figure 2) which in addition to the title, gives the number of free sectors left on the diskette, formats the file-names in 2 columns on a single video screen display of the catalog, and replaces the words, DISK VOLUME, with personalized and helpful cataloging information.

Fig 1
USUAL CATALOG

```
DISK VOLUME 045

*B 008 IMPR CATALOG
*A 005 KEY
*B 005 CAT
*B 007 VTOC RESTRICT
*B 002 CTRLFIND
*I 026 DOU
*A 032 PROGRAM INFO
*B 005 DISK MAP
*I 007 FASTDOS 3.2*
*B 017 FISHHEAD G
*A 006 MENU
*I 022 I/B WKSP
*A 027 A/S WKSP
*I 002 INTEGER
*I 006 BSTAT
*A 006 LABEL PRINT
*I 006 LABELLER
*B 009 UPDATE 3.2*
*I 053 EDU
 B 02A WORKSHOP
*I 019 DISK HELPER
*I 022 RENUMBER
*B 010 RENUMBER.B
*I 012 F.U.B.S.
*B 021 SUPER COPY
*B 003 SEARCH
*B 009 UNIV COPY
*I 000 LOADER
*I 012 VOL# CHANGE
*I 012 COPY**
 T 008 >CATFILE
*I 012 PLE
 T 002 *
 T 000 <4/22/80>

>
```

Figure 2

```
3  SECTORS FREE                *I 012 I/B WKSP
                               *I 013 IMPR CATALOG
CMH U/DISK  045                *B 014 INTEGER
                               *I 015 KEY
+++++++++++++++++              *A 016 LABEL PRINT
![U/DISK 045D]!                *A 017 LABELLER
!. H A N D Y .!                *B 018 LOADER
!. UTILITIES .!                *I 019 MENU
!.. DOS 3.2* .!                *A 020 PLE
                               *I 021 PROGRAM INFO
                               *I 022 RENUMBER
*A 001 A/S WKSP                *B 023 RENUMBER.B
*I 002 BSTAT                   *B 024 SEARCH
*B 003 CAT                     *B 025 SUPER COPY
*I 004 COPY**                  *I 026 F.U.B.S.
*B 005 CTRLFIND                *B 027 UNIV COPY
*I 006 DOU                     *B 028 UPDATE 3.2*
*B 007 DISK HELPER             *I 029 VOL# CHANGE
*I 008 DISK MAP                *B 030 VTOC RESTRICT
*I 009 FASTDOS 3.2*             T 031 W
*I 010 FISHHEAD G               A 032 WORKSHOP
*B 011 EDU                      T 033 >CATFILE
                                T 034 <4/22/80>
>
```

DIRECTORY TITLE FORMATTING (DTF)
The double column display and free sector count is obtained either by SPLITCAT
(Call -APPLE Vol 2, #4) or P.L.E.
(Call -APPLE's PROGRAM LINE EDITOR)
Alphabetization by DISK SORT - TED COHN[1]
File Length Order by DISK HELP - TED COHN[1]
For title construction itself, see next page.

[1] Programs from SAN FRANCISCO APPLE CORE

Now that we have assigned each row a value, we need to put it into the program. Look at the program listing at the end of this article and I'll show you how I did it there. First look at line # 193. 193 is ASCII for the 'A', so I put it on line #193 to make it easy to access by using the ASCII number to calculate where to go. Now, look at the data on that line. It says A=8, which is the value of row #1. Then, it says B-20, which is the value of row#2. C-34, which is the value of row #3. Then D=C, F=C and G=C because we want them all to be the same. We also have E=64, which is the value of the 5th line. The only row we did not do is row 8 because in text all row 8's are = to 0, which is a blank line. If you want to use that last row in your own characters, you just have to give it a value and put it in. Now look at line #10 in the program. It has the formula to find the location using the X/Y coordinates, and then a series of pokes. The first POKE is into location L with the value that we gave the 1st row. The second POKE is into location L=1024. If you remember back when we were discussing the formatting of the screen, we found that in going from one Hi-res line to the next, the address would increase by 1024 for 8 lines. That is why each of the subsequent POKEs on line 10 will increase by 1024 each time. Notice the last POKE uses the value of 0. That is for the bottom row which is blank. If you are using the bottom row in your characters, you will have to POKE its value, instead of 1, here.

One last thing to mention before closing, is the fact that you may want to put your characters someplace besides exactly within the boxes mentioned. Take a look at figure #2 on chart #6. This represents a graph line that may have been plotted on the screen. Now you want to put a character out of the normal boundaries. You have two problems. #1 You have to have a formula that will shift the character and calculate the new values from those received from the data tables. #2 On the bottom left

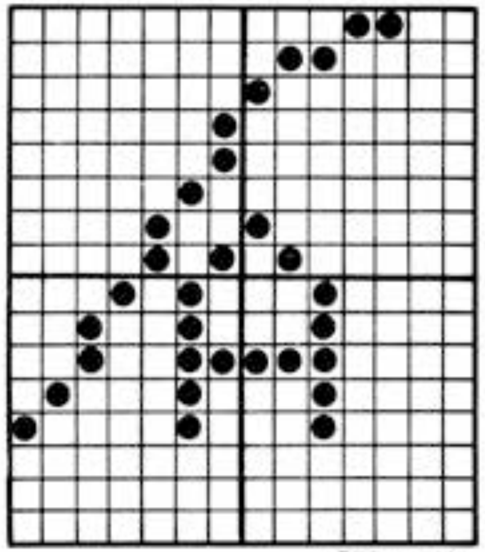


FIGURE # 2

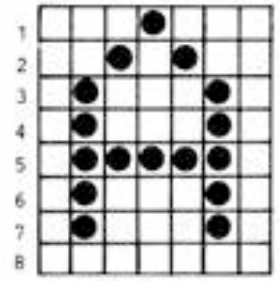


FIGURE #1

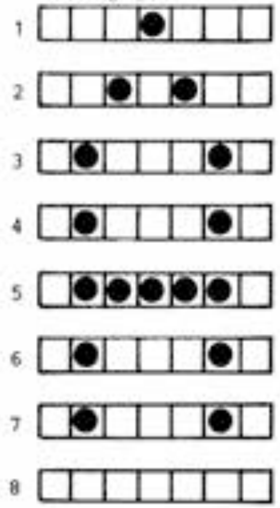


CHART #6

Figure #1 shows the letter 'A' as it would appear on the screen in Hi-res. Notice that the 1st and last columns are not used. The dots as they appear on the screen are so close together sideways that it takes 2 spaces to separate the characters. The dots are already far enough apart up and down so that only one space is necessary to separate them. Each row of the character uses a different byte of RAM in Hi-res, unlike the one character to a byte in text. Therefore, you see below the 'A', each row is separated from the rest. Each box in the row is representative of the 7 bits used in Hi-res to control the screen. By using information on chart #5 we can figure out what the value of each row of the 'A' is. Remember though, the Hi-res screen looks at the bits in reverse order, so that bit #1 is on the left instead of on the right. Try to calculate what you think the value of each row is before reading further. The values of the rows are as follows:

Row #1=8, #1=20, #3=34, #4=34, #5=62, #6=34, #7=34 and #8=0.

Figure #2 shows 4 areas, each the size of a normal character. The area could have come from anywhere on the screen. It depicts a portion of a graph with the letter 'A' beside it. You would not normally put a letter this close to a graph because the letter actually touches it. This is one way to demonstrate that sometimes you may want a letter to be in a place other than the normal squares where they usually go. I do not have the time or space to go into detail, however, there are some clues in the text of this article on how to do this.

Witch Trial

Dan Henderson
Dr. Marc A. Golombeck
Roby Sherman

Produced by
Brian Wiser & Bill Martens

Step back in time to 1600s England! You have been accused of witchcraft and must prove your innocence in this one-of-a-kind Double Lo-Res point and click adventure game for the Apple II. With over 40 locations and lots of medieval villagers to speak with, you have every chance of avoiding an unjust death.

Witch Trial

- Perfect for children and adults over 8-years-old.
- Mouse or Joystick control with a Mockingboard soundtrack.
- Designed and tested to work with the Apple IIGS, IIe, IIc and IIc Plus.
- Artwork and Story by Dan Henderson.
- Programming by Dr. Marc A. Golombeck and Roby Sherman.

Apple PugetSound Program Library Exchange
www.callapple.org
9 781387 893171

Joystick / Mouse

When the game starts up, a screen appears to choose Mouse or Joystick. Type "M" or "J" to make your selection.

Movement: Controls the pointer. Hover to the right of the screen to see the blue navigation menu. Using a point-and-click system, the user navigates through the adventure and interacts with the world.

Button: Mouse or Joystick fire button to select menu items.

Directions: Denoted by the blue blocks on the screen. **N, S, E, W** denote the directions North, South, East, and West.

Items: Individual items can be acquired or "picked up" by clicking on an item within a scene.

Inventory: The Inventory block **[I]** will show the adventurer the items they are currently carrying.

Talk: The Talk block **[T]** will enable the adventurer to talk with the characters in the scene.

Walkthrough

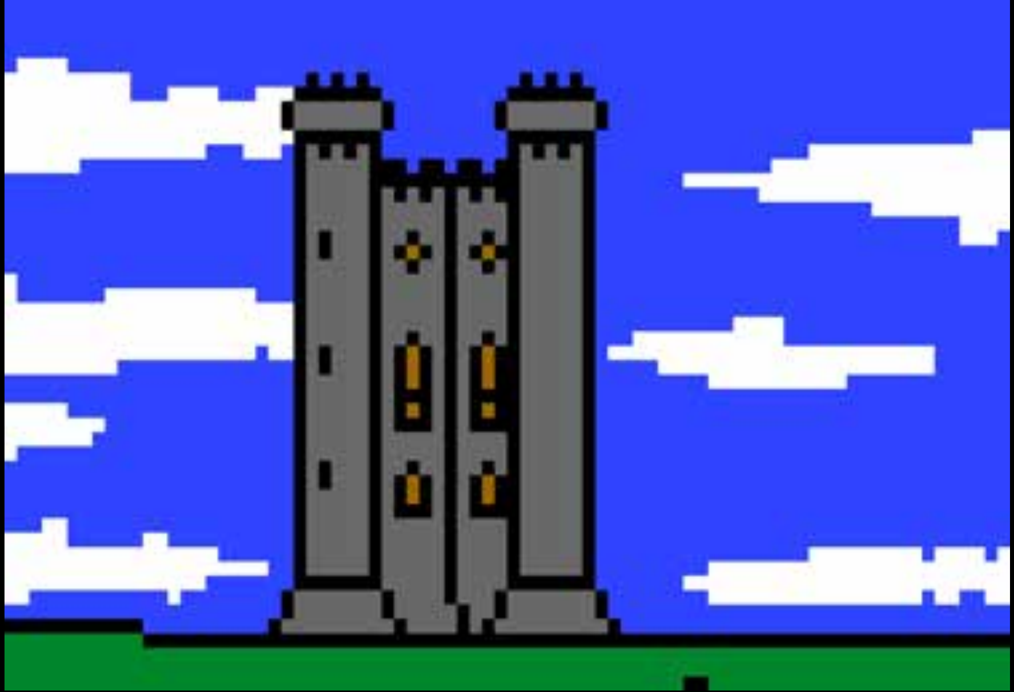
Spoiler Alert: If you don't want a walkthrough, then stop reading now. Otherwise, be forewarned, the following pages provide a full run-through of the necessary items and steps to complete the game – complete with pictures.

Afterwards, pictures are shown of the Ending, an Easter Egg, and an Art Gallery featuring the remaining game screens.

Stage 1

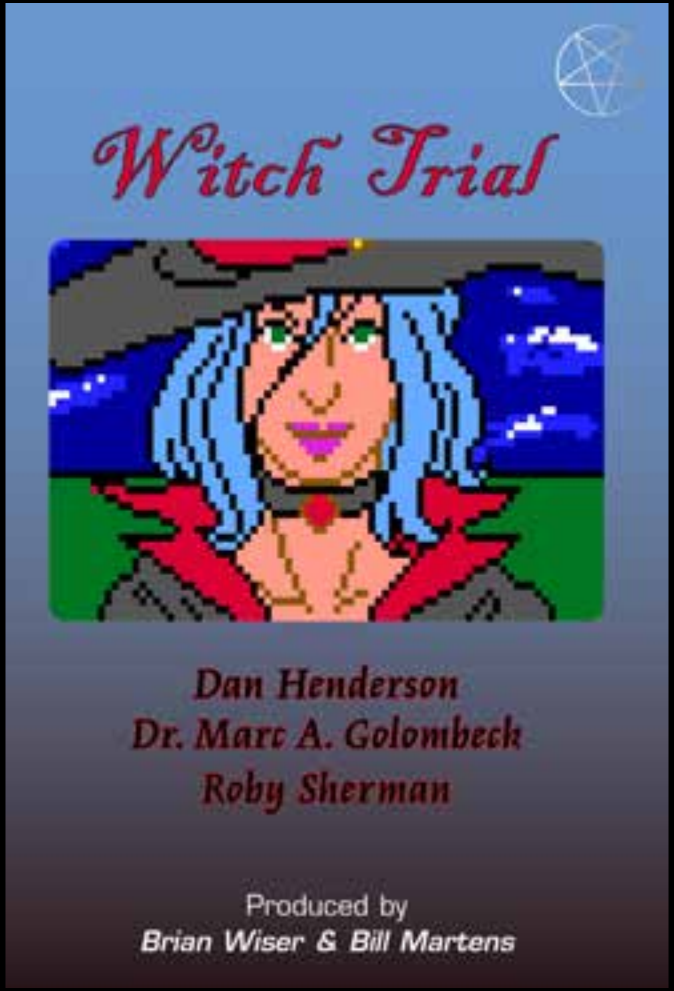
These pictures are scenes and people you can interact with. They don't help you with your quest, but are part of the big puzzle you have to solve (to find those people you really need to interact with).

Most imporant, the pictures represent the remaining beautiful *Witch Trial* art by Dan Henderson – preserved as it should be for artistic enjoyment.



Limited Box & Disk?

Email: editor@callapple.org
Subject: Game Boxes

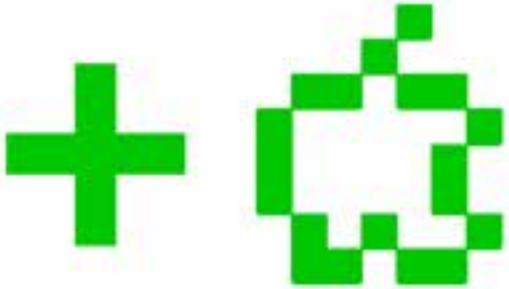


Lordlings of Yore
The Game of Knights, Knaves and Necromancers

Call–A.P.P.L.E.™

Apple Pugetsound Program Library Exchange

Ringoban



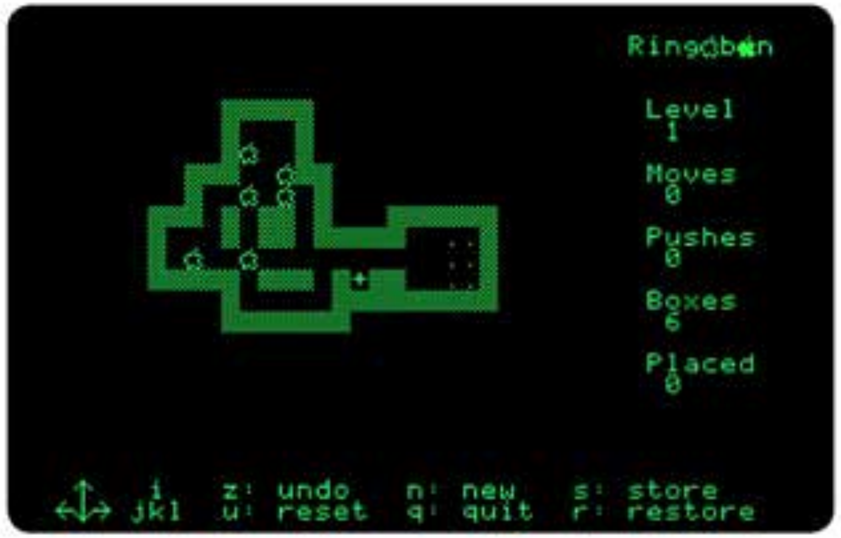
Mike Beaumont

Produced by
Brian Wiser & Bill Martens

Ringoban is an Applesoft BASIC version of the game *Sokoban* for the Apple II. This version by Mike Beaumont contains 90 levels of brain-warping fun. Work through the levels one-by-one and become the master of the Ringoban. It provides a great challenge for even the best puzzlers as well as the information necessary to create your own levels.

Features Include:

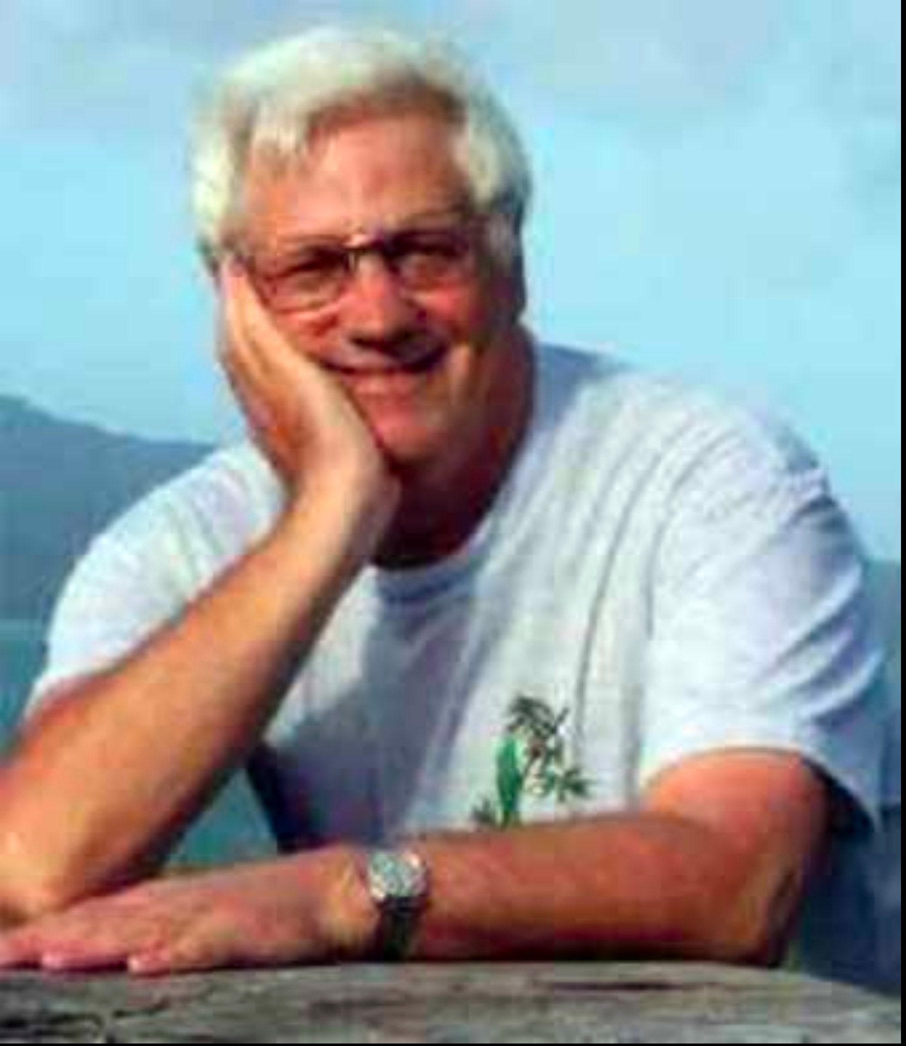
- Uses MouseText and requires ProDOS.
- 90 levels of fun, complete with maps for the perpetually confused.
- Complete source code.
- Design your own levels using any word processor that can save a standard text file.

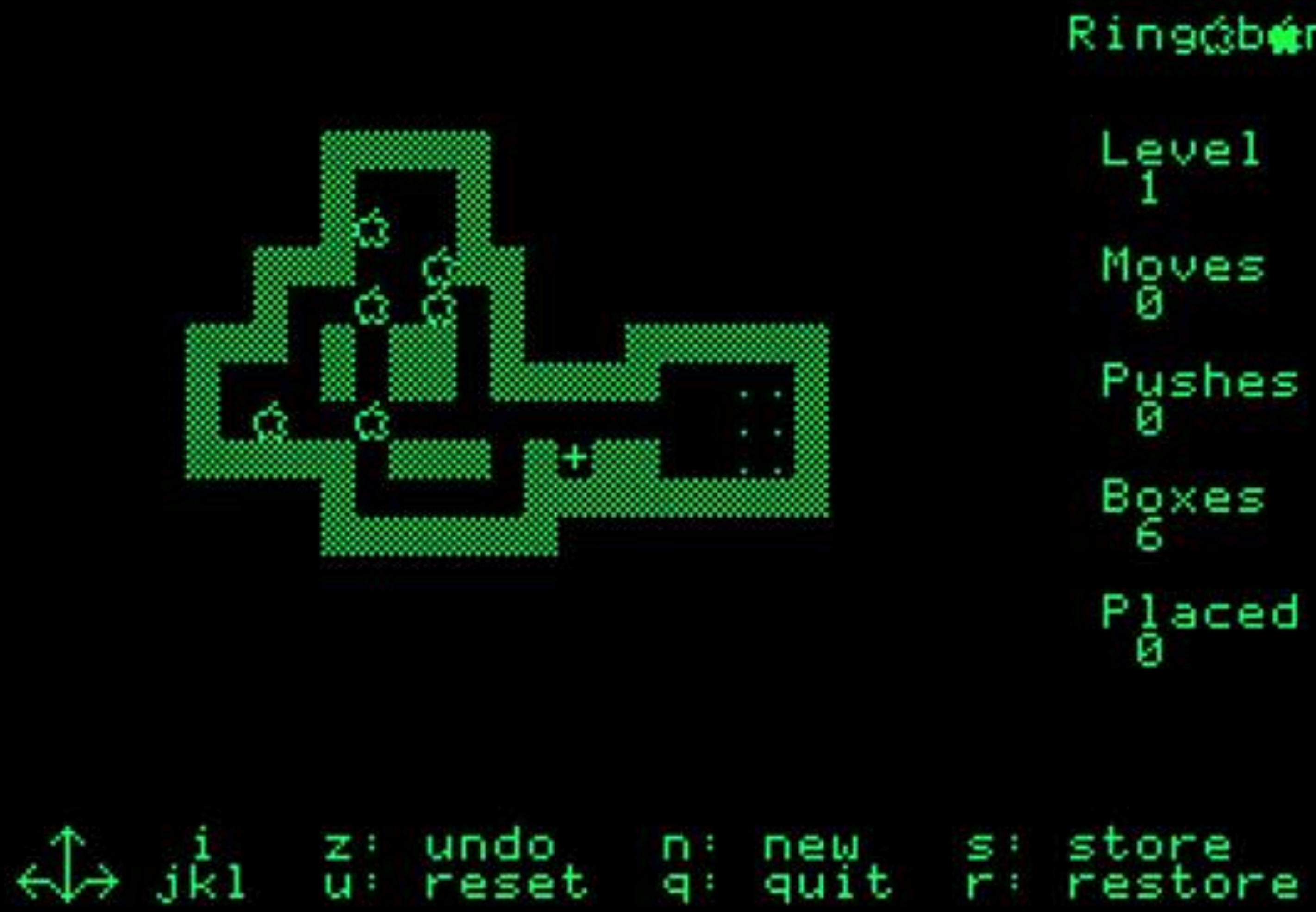


```
                                   Ringoban

                                    Level
                                    1
                                    Moves
                                    0
                                    Pushes
                                    0
                                    Boxes
                                    6
                                    Placed
                                    0

    i   z: undo   n: new   s: store
   jkl  u: reset  q: quit  r: restore
```

Apple PugetSound Program Library Exchange
www.callapple.org

9 781794 777613




Ringóbón
Level
1
Moves
0
Pushes
0
Boxes
6
Placed
0
i
jkl
z: undo
u: reset
n: new
q: quit
s: store
r: restore

Call–A.P.P.L.E.™

Apple Pugetsound Program Library Exchange

Terminal Boredom

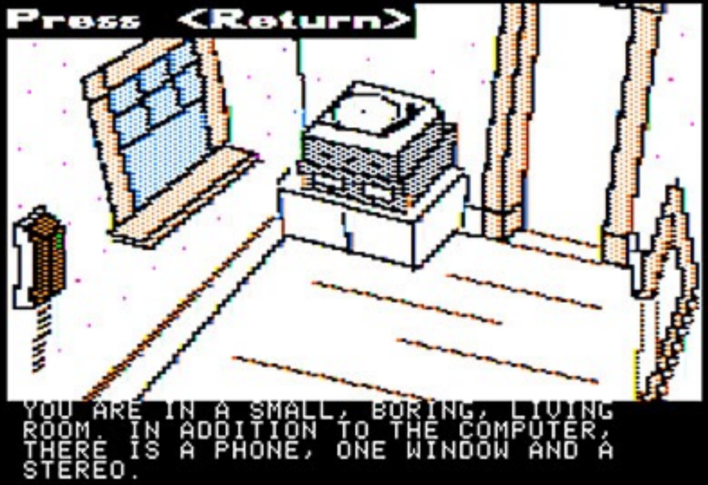
Karl Bunker

Produced by
Brian Wiser & Bill Martens

Terminal Boredom is an illustrated Apple II game of the "adventure" type. However, rather than the usual slay-the-dragon / destroy-the-giant-robot / find-the-hidden-treasure stuff typically found in such games, the primary object is simply to keep your on-screen alter ego from falling asleep.

In addition to giving it something to occupy its sleepy little mind, you'll have to make sure that your alter ego begins and completes a study of Applesoft BASIC.

Because your character seems to be perpetually on the verge of being bored to unconsciousness, neither of these is too easy. Winning requires planning, careful observation and the experience that comes from losing the game a few (or many) times over. You'll have to find ways to keep your character awake, and you'll have to avoid various sleep-inducing obstacles that are thrown in your path.

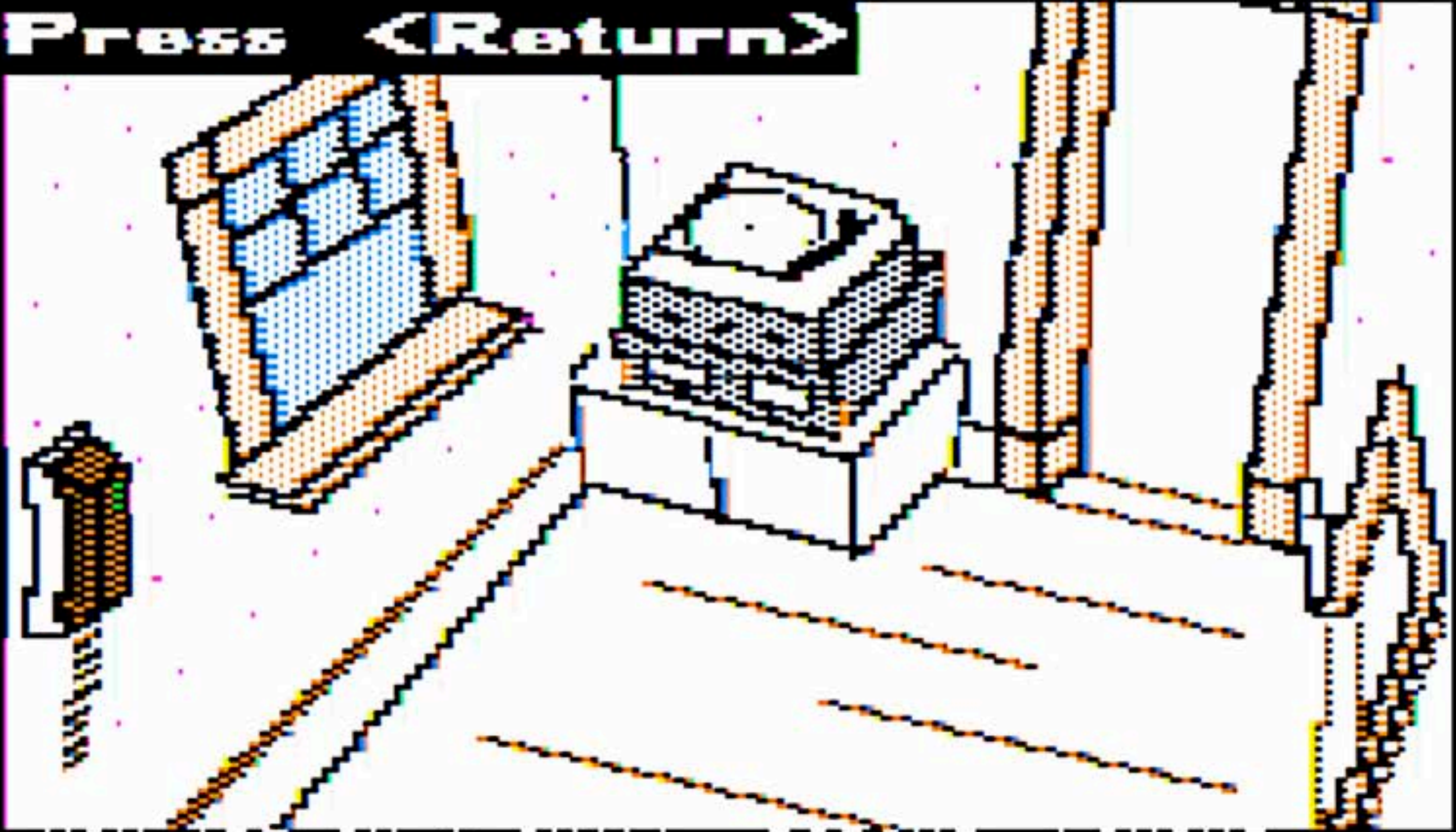


```
Press  <Return>
YOU ARE IN A SMALL, BORING, LIVING
ROOM. IN ADDITION TO THE COMPUTER,
THERE IS A PHONE, ONE WINDOW AND A
STEREO.
```

Apple PugetSound Program Library Exchange

www.callapple.org

9 781387 893157



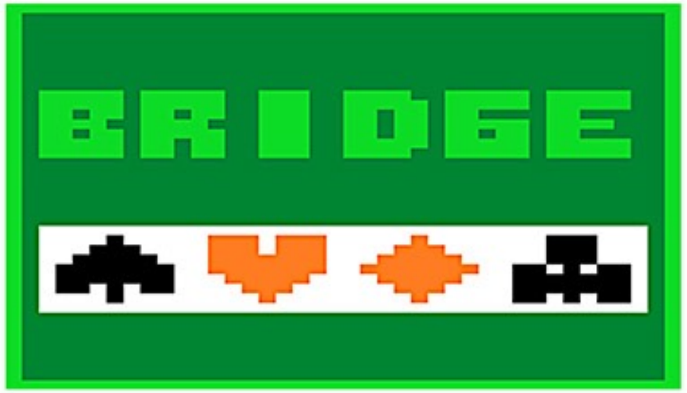
YOU ARE IN A SMALL, BORING, LIVING
ROOM. IN ADDITION TO THE COMPUTER,
THERE IS A PHONE, ONE WINDOW AND A
STEREO.

Call–A.P.P.L.E.™

Apple Pugetsound Program Library Exchange

Trik

Contract Bridge for the Apple II

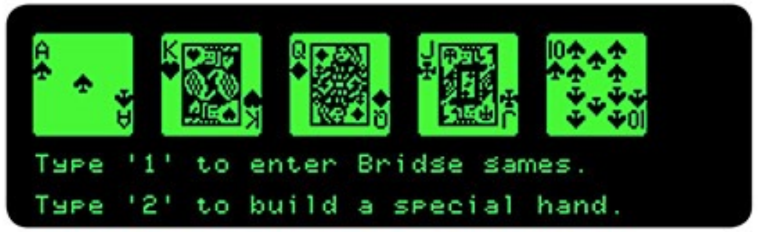


Jim Hilger

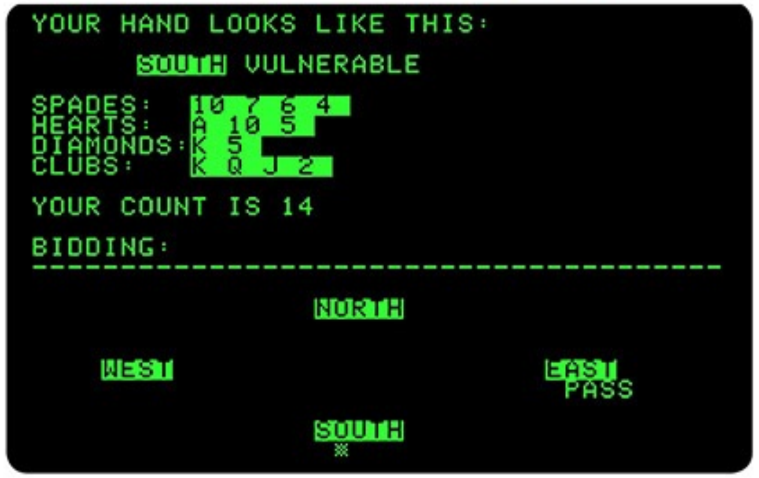
Produced by
Brian Wiser & Bill Martens

Trik is a full-featured single player Contact Bridge game for the Apple II computer. Features include playing a straight up game, building a hand, duplicating other games, sample hands for learning the game, and many others. *Trik* is written in Apple II Integer BASIC.

The game uses standard codes during play, allowing the user to display the hands, play out the current hands, re-bid with the same hands, or restart with a new deal. This manual includes basics such as How to Play, Play Codes, and Savings Hands. Background information on Contract Bridge, History of the Game, Rules, Strategy, Bidding, and Play Techniques are also included.



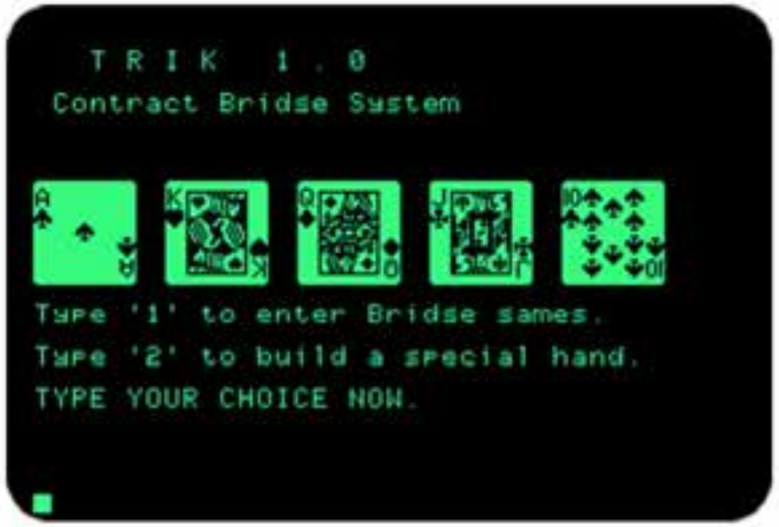
```
Type '1' to enter Bridge games.

Type '2' to build a special hand.
```



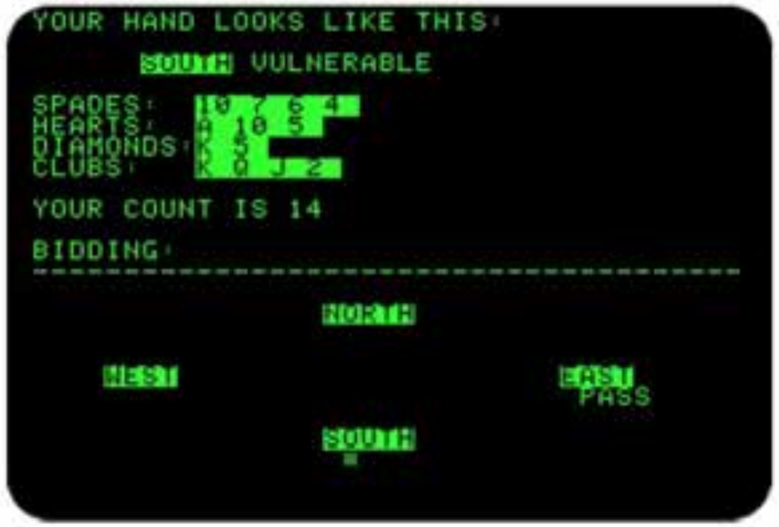
```
YOUR HAND LOOKS LIKE THIS:
      SOUTH VULNERABLE

SPADES:   10 7 6 4
HEARTS:   A 10 5
DIAMONDS: K 5
CLUBS:    K Q J 2

YOUR COUNT IS 14

BIDDING:
---------------------------------------

              NORTH

     WEST                    EAST
                              PASS

              SOUTH
```

Apple PugetSound Program Library Exchange
www.callapple.org

9 781387 800209



```
  T R I K   1 . 0
 Contract Bridge System

Type '1' to enter Bridge games.
Type '2' to build a special hand.
TYPE YOUR CHOICE NOW.
```



```
YOUR HAND LOOKS LIKE THIS:
      SOUTH VULNERABLE
SPADES:    10 7 6 4
HEARTS:    A 10 5
DIAMONDS:  K 5
CLUBS:     K Q J 2
YOUR COUNT IS 14
BIDDING:
----------------------------------------

              NORTH

    WEST                        EAST
                                 PASS

              SOUTH
```

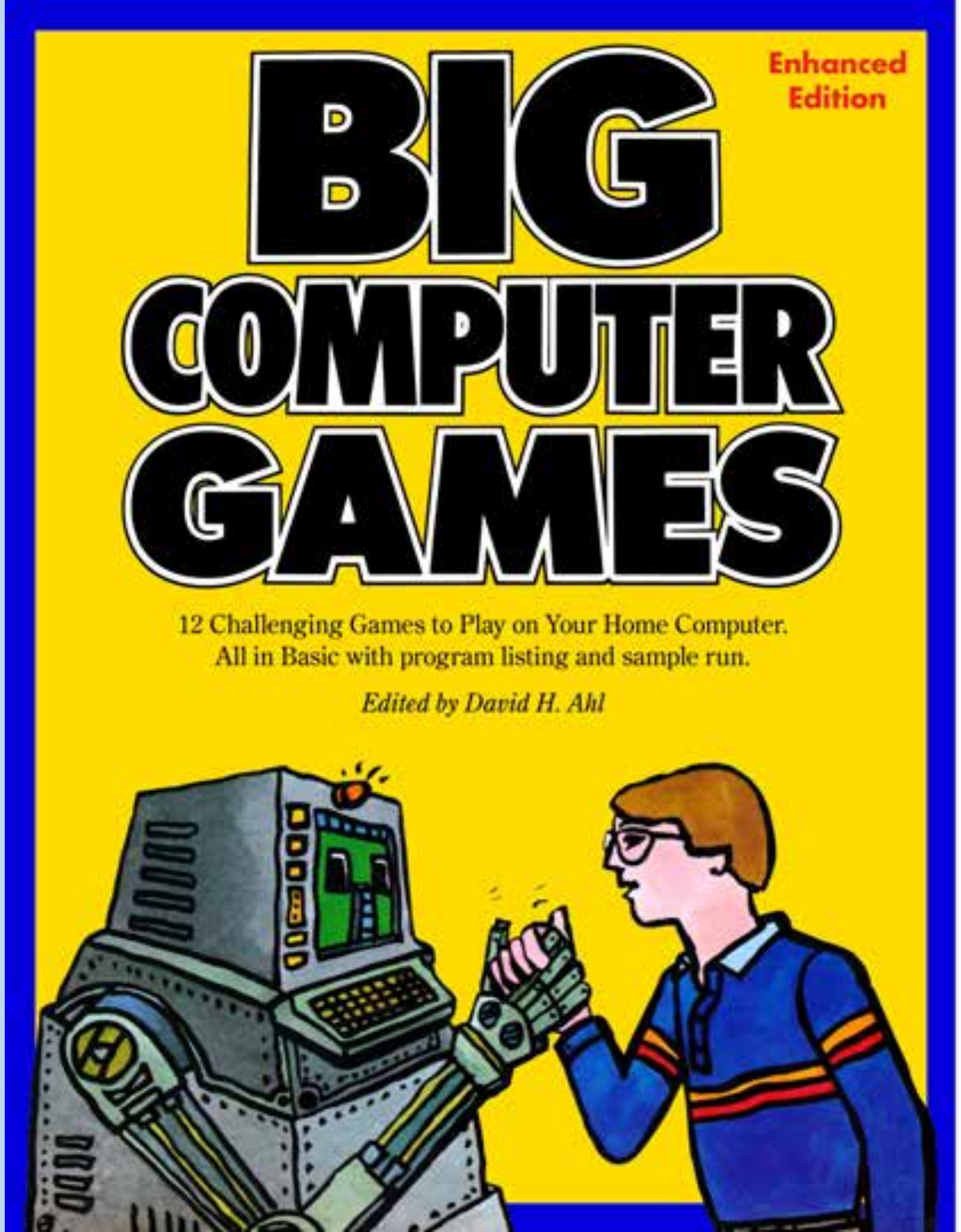
How to Play

There will be a pause as TRIK 1.0 BID executes the CHAIN to TRIK1.0 PLAY. If EAST/WEST are the declarer, you will be asked to swap hands with EAST or WEST, since TRIK 1.0 PLAY only plays defense. If NORTH/SOUTH are the declarer then no hand switching is necessary.

Next NORTH becomes the dummy and WEST begins thinking about what to lead. Since WEST is thinking in Integer BASIC, it takes a number of seconds to actually play. During play, you will always play for the dummy (NORTH) and SOUTH, while the computer will always play for EAST and WEST. A flashing question mark will appear whenever it is your turn to play a card.

Once again, you are provided with a number of short-hand codes for entering the cards you wish to play. They are show in the Play Codes section.

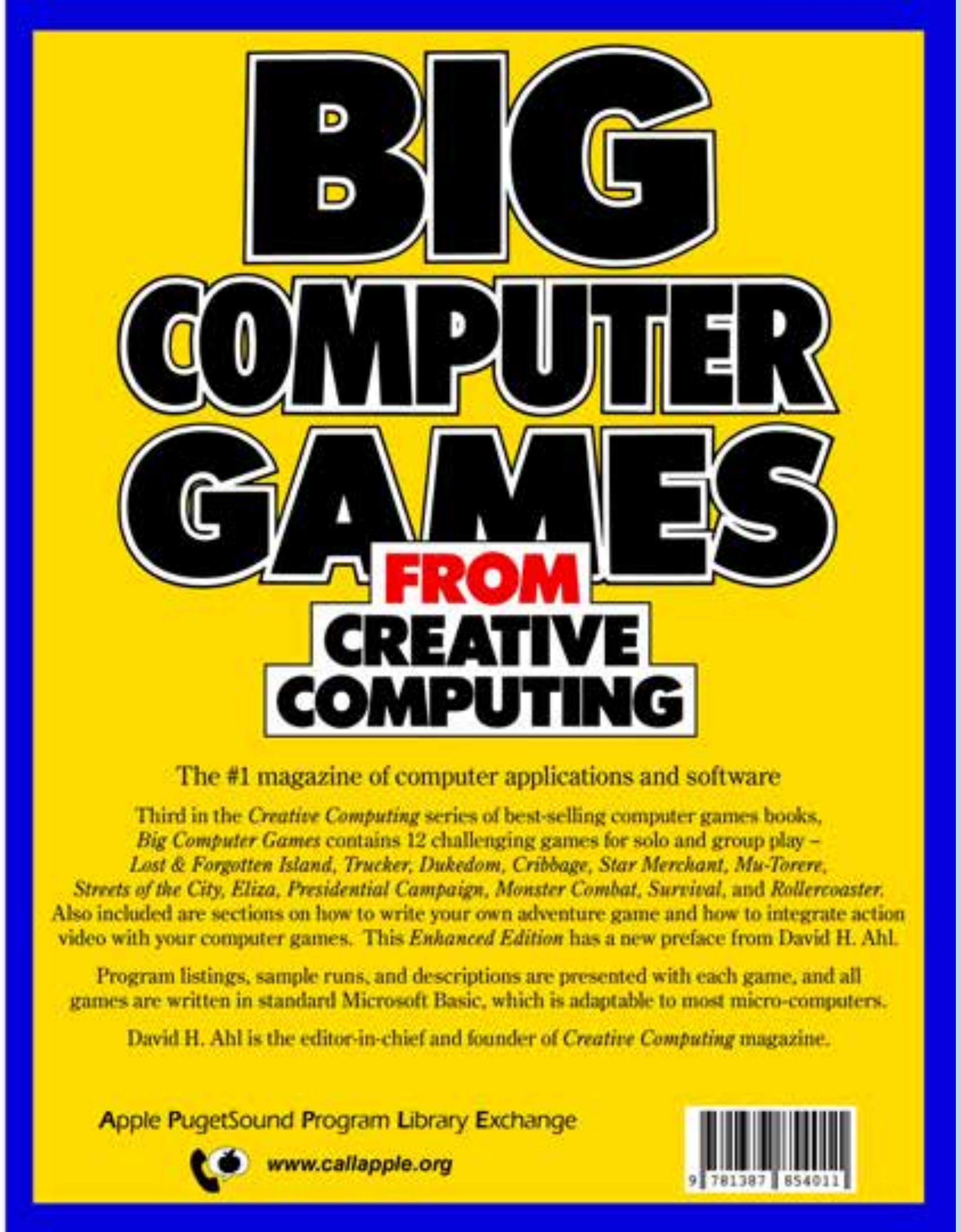
As with the bidding, once you enter the code you wish to play, you need to press the Return key. If the code is not a valid code, the computer will buzz disgustingly at you, and await a proper play.


Enhanced Edition

BIG COMPUTER GAMES

12 Challenging Games to Play on Your Home Computer.
All in Basic with program listing and sample run.

Edited by David H. Ahl


BIG COMPUTER GAMES FROM CREATIVE COMPUTING

The #1 magazine of computer applications and software

Third in the *Creative Computing* series of best-selling computer games books, *Big Computer Games* contains 12 challenging games for solo and group play – *Lost & Forgotten Island, Trucker, Dukedom, Cribbage, Star Merchant, Mu-Torere, Streets of the City, Eliza, Presidential Campaign, Monster Combat, Survival,* and *Rollercoaster.* Also included are sections on how to write your own adventure game and how to integrate action video with your computer games. This *Enhanced Edition* has a new preface from David H. Ahl.

Program listings, sample runs, and descriptions are presented with each game, and all games are written in standard Microsoft Basic, which is adaptable to most micro-computers.

David H. Ahl is the editor-in-chief and founder of *Creative Computing* magazine.

Apple PugetSound Program Library Exchange

www.callapple.org

9 781387 854011

Dukedom

You are one of several Dukes chosen by the High King to help run the Kingdom. Your Duchy is not in the best of shape, and your job is to build up its population, land holdings, and grain reserves. Your secret ambition is to become powerful enough to overthrow the High King.

The game cycles on an annual basis, and it is now fall and the harvest has just been completed. Each year at this time the computer will display the current population, land and grain totals, followed by a detailed report of the previous year's events. Note that land and grain are measured in metric units hectares (HA.) and hectoliters (HL.), respectively.

Each year you will have to make the following decisions:

Grain for Food

You must decide how much grain to feed the peasants. 14 HL. of grain will just adequately feed one peasant; 13 will cause some hunger and decrease the peasants' fighting ability, and 12 or fewer will cause some starvation. The peasants will complain if you try to starve then excessively and they know that you are holding back grain. If you feed the peasants more than 14 HL. each (up to a maximum of 18) they will appreciate the boom and fight better in any war the following summer. A long term memory keeps track of the peasants' cumulative attitude (it fades slowly with time) and if you create sufficient bad will (by underfeeding them, for instance) they will depose you. You may enter the quantity of grain for the peasants in two ways: Numbers less than 100 are interpreted as hectoliters-per-peasant, while an entry of 100 or more represents the total amount for the entire population.

Land to Buy

Enter the number of hectares of land you want to buy. The prices offered vary from about 4 hectoliters/hectare to about 30, depending primarily on last year's crop yield. If you don't want to buy any land, enter 0. You will then be given the option of selling your land at a price one unit lower than the buying price. Enter the number of hectares you want to sell, or enter 0 if you don't want to sell any.

Land to Plant

Enter the number of hectares you wish to plant. Each hectare planted will require 2 hectoliters of grain to seed it. Also, remember that each peasant can plant

```
Grain at start             5138
Used for food             -2364
Seeding                   -1100
Rat losses                -181
Crop yield                 3707
Castle expense            -120
Royal tax                 -475
Grain at end of year       4605


Grain for food =12
Some peasants have starved
Land to buy at 12 HL./HA. =
Land to sell at 11 HL./HA. =
Land to be planted = 450
Yield = 7.82 HL./HA.
Rats infest the gainery
The king requires 3 peasants for
his estate and mines.Will you supply
them (Yles or pay 300 HL.of
grain instead (N)o ? y
```

```
Year 8 Peasants 220 Land 950 Grain 4019

Peasants at start            205
Starvations                  -15
King's levy                  -3
Natural deaths               -9
Births                        26
Peasants at end              220

Land at start                950
Land at end of year          950

100%  80%  60%  40%  20%  Depl
  500  400   50    0    0    0

Grain at start               4605
Used for food               -2460
Seeding                     -900
Rat losses                  -150
Crop yield                   3519
Castle expense              -120
Royal tax                   -475
Grain at end of year         4019


The peasants tire of war and stevation
You are deposed

Do you wish to play again ? n
```

Eliza

Eliza was originally written by Joseph Weizenbaum in LISP at MIT. The first version in Basic was written by Jeff Shrager in 1973 and converted to MITS 8K Basic (later to become Microsoft Basic) by Steve North in 1977. It originally appeared in *Creative Computing*, July/August 1977.

Introduction

Eliza is a program which accepts natural English as input and carries on a reasonably coherent conversation based on the non-directive psychoanalytic techniques of Carl Rogers. You will have to forgive Eliza for her awkward English. You will find it is best not to use punctuation (especially commas and contractions) in your input and keep each line of input to one main idea. Since Eliza is a non-directive therapist, you will have to carry the conversation; nevertheless, that can lead some mighty interesting results. You may end your conversation by typing in "SHUT UP" (or just "SHUT").

How It Works

In order to do what it does, Eliza must: (1) get a string from the user and prepare it for further processing; (2) find the keywords in the input string; (3) if a

```
2010 DATA "ARE YOU SAYING THAT JUST TO BE NEGATIVE?"
2020 DATA "YOU ARE BEING A BIT NEGATIVE."
2030 DATA "WHY NOT?"
2040 DATA "ARE YOU SURE?"
2050 DATA "WHY NO?"
2060 DATA "WHY ARE YOU CONCERNED ABOUT MY*"
2070 DATA "WHAT ABOUT YOUR OWN*"
2080 DATA "CAN YOU THINK OF A SECIFIC EXAMPLE?"
2090 DATA "WHEN?"
2100 DATA "WHAT ARE YOU THINKING OF?"
2110 DATA "REALLY, ALWAYS?"
2120 DATA "DO YOU REALLY THINK SO"
2130 DATA "BUT YOU ARE NOT SURE YOU*"
2140 DATA "DO YOU DOUBT YOU*"
2150 DATA "IN WHAT WAY?"
2160 DATA "WHAT RESEMBLENCE DO YOU SEE?"
2170 DATA "WHAT DOES THE SIMILARITY SUGGEST TO YOU?"
2180 DATA "WHAT OTHER CONNECTIONS DO YOU SEE?"
2190 DATA "COULD THERE REALLY BE SOME CONNECTION?"
2200 DATA "HOW?"
2210 DATA "YOU SEEM QUITE POSITIVE."
2220 DATA "ARE YOU SURE?"
2230 DATA "I SEE."
2240 DATA "I UNDERSTAND."
2250 DATA "WHY DO YOU BRING UP THE TOPIC OF FRIENDS?"
2260 DATA "DO YOUR FRIENDS WORRU YOU?"
2270 DATA "DO YOUR FRIENDS PICK ON YOU?"
2280 DATA "ARE YOU SURE YOU HAVE ANY FRIENDS?"
2290 DATA "DO YOU IMPOSE ON YOUR FRIENDS?"
2300 DATA "PERHAPS YOUR LOVE FOR FRIENDS WORRIES YOU?"
2310 DATA "DO COMPUTERS WORRY YOU?"
2320 DATA "ARE YOU TALKING ABOUT ME IN PARTICULAR?"
2330 DATA "ARE YOU FRIGHTENED BY MACHINES?"
2340 DATA "WHY DO YOU MENTION COMPUTERS?"
2350 DATA "WHAT DO YOU THINK COMPUTER HAVE TO DO WITH YOUR PROBLEM?"
2360 DATA "DON'T YOU THINK COMPUTERS CAN HELP PEOPLE?"
2370 DATA "WHAT IS IT ABOUT MACHINES THAT WORRIES YOU?"
2380 DATA "SAY, DO YOU HAVE ANY PSYCOLOGICAL PROBLEMS?"
2390 DATA "WHAT DOES THAT SUGGEST TO YOU?"
2400 DATA "I SEE."
2410 DATA "I'M NOT SURE I UNDERSTAND YOU FULLY."
2420 DATA "COME COME ELUCIDATE YOUR THOUGHTS."
2430 DATA "CAN YOU ELABORATE ON THAT?"
2440 DATA "THAT IS QUITE INTERESTING."
2500 REM
2510 REM --- REPLY FINDING DATA ---
2520 REM
2530 DATA 1,3,4,2,6,4,6,4,10,4,14,3,17,3,20,2,22,3,25,3
2540 DATA 28,4,28,4,32,3,35,5,40,9,40,9,40,9,40,9,40,9,40,9
2550 DATA 49,2,51,4,55,4,59,4,63,1,63,1,64,5,69,5,74,2,76,4
2560 DATA 80,3,83,7,90,3,93,6,99,7,106,6
```

```
Hi! I'm Eliza. What is your problem?
? I'M AN IBM 370
DID YOU COME TO ME BECAUSE YOU ARE AN IBM 370
? YES I WANT A NEW LINE PRINTER
WHAT WOULD IT MEAN IF YOU GOT  A NEW LINE PRINTER
? ABOUT 1200 LPM
SAY, DO YOU HAVE ANY PSYCOLOGICAL PROBLEMS?
? HOW WOULD YOU KNOW YOU'RE A SPECTRAVIDEO
WHAT MAKES YOU THINK I AM  A SPECTRAVIDEO
? IT SAYS SO ABOVE YOUR KEYBOARD
WHY ARE YOU CONCERNED ABOUT MY  KEYBOARD
? CAUSE THAT IS WHAT I'M TYPING ON
HOW LONG HAVE YOU BEEN TYPING ON
? SINCE I TYPED RUN
WHAT DOES THAT SUGGEST TO YOU?
? YOU'RE THE SHRINK
DOES IT PLEASE YOU TO BELIEVE THAT I AM  THE SHRINK
? DON'T YOU KNOW
WE WERE DISCUSSING YOU-- NOT ME.
? I AM BEGINNING TO WONDER ABOUT YOU
DID YOU COME TO ME BECAUSE YOU ARE  BEGINNING TO WONDER ABOUT I
? I DIDN'T START TO WONDER ABOUT YOU UNTIL AFTER I CAME TO YOU
```

Monster Combat

Monster Combat was written by Lee J. Chapel and originally appeared in *Creative Computing*, February 1981.

Monster Combat is a game in which you wander around a forest and encounter various monsters. Your objective is to win as much treasure from each encounter as possible and, of course, not get killed in the process.

Play of the Game

When you play the game you will be randomly placed in a forest ten by ten squares in size. Only one of these squares (the one you are in) is displayed, thus allowing you to see only a small part of the forest at a time. The sector you are in is again divided into ten by ten squares. Each of these, too, is divided up to ten by ten; but you can see these hundred smallest squares. Each of these little squares is shown by a single character. It covers an area of forest ten by ten yards, making the faller square that is displayed a hundred by a hundred yards and the entire forest a thousand by a thousand yards. T's are trees, -'s are paths, I's are walls, ^'s are inns, and M's are enchanted castles. The "U" is you.

Also displayed with the portion of forest you are in is your combat strength, treasure total, and the various magic spells you have. Your combat strength is used to fight the various monsters you meet, each monster having a combat strength of his own; these range from five (for a goblin) to a hundred (for a basilisk). Your combat strength is also used in movement, the amount used depending upon how far you go, how much treasure you're lugging around, and the type of terrain you end up on after you move.

At the inns you are allowed to regain the strength you began with and all the magic you had at the start. Don't worry when you find yourself displayed in the square below the inn when you stop there; that is the way the program is set up. Of course, the innkeeper takes some of your treasure for providing you with his services. However, sometimes he has information which he passes on to you at no additional cost—like where the forest edge is, or where an enchanted castle might be found.

There may be up to fifteen enchanted castles in the forest. These usually contain items of great value

Star Merchant

```
1560 IF A(I,1)=0 THEN 1630
1570 FOR J=1 TO 19*A(I,1)
1580 READ A$
1590 NEXT J
1600 READ A$
1610 PRINT USING"##";I;:PRINT TAB(3);A$;TAB(20);:PRINT USING"###";A(I,3);
1620 PRINT TAB(25);:PRINT USING"########";A(I,2);:PRINT TAB(35);:PRINT USING"###
";A(I,4)
1630 NEXT I
1640 RETURN
1650 REM Buy Cargo Sub
1660 PRINT "Enter the lot number of cargo that you":INPUT "want to purchase";K
1670 IF K=ABS(INT(K)) AND K>0 AND K<6 THEN 1700
1680 PRINT "Invalid lot number."
1690 RETURN
1700 IF A(K,1)>0 THEN 1730
1710 PRINT "Lot";K;"has already been purchased."
1720 RETURN
1730 IF A(K,2)<=B2 THEN 1760
1740 PRINT "You can not buy cargo on credit."
1750 RETURN
1760 IF A(K,3)<=W2 THEN 1790
1770 PRINT "You do not have sufficient cargo space.")
1780 RETURN
1790 FOR I=1 TO 20
1800 IF H(I,1)=0 THEN 1840
1810 NEXT I
1820 PRINT "All 20 cargo partitions are occupied."
1830 RETURN
1840 FOR J=1 TO 4
1850 H(I,J)=A(K,J)
1860 NEXT J
1870 A(K,1)=0
1880 W2=W2-H(I,3)
1890 B2=B2-H(I,2)
1900 T6=T6+3E-03
1910 N1=N1-1
1920 PRINT:PRINT "Transaction completed"
1930 PRINT "Cargo stored in partition";I
1940 RETURN
1950 REM Sell Cargo Sub
1960 INPUT"Enter partition of cargo to be sold";K
1970 IF K=ABS(INT(K)) AND K>0 AND K<21 THEN 2000
1980 PRINT "Invalid partition number"
1990 RETURN
2000 IF H(K,1)>0 THEN 2030
2010 PRINT "Cargo partition is empty."
2020 RETURN
2030 B2=B2+H(K,2)
2040 W2=W2+H(K,3)
2050 T6=T6+3E-03
2060 H(K,1)=0
2070 PRINT "Transaction completed"
2080 RETURN
2090 REM List Starports Sub
2100 PRINT "No.  Name              Trade Cl  Dist Dir"
2110 RESTORE
2120 FOR I=1 TO 10
2130 READ A$,B$
2140 PRINT USING"##";I;:PRINT TAB(5);A$;TAB(21);B$;
2150 PRINT TAB(29);:PRINT USING"##.##";D(I);:PRINT TAB(35);:PRINT USING"###";T(I)
2160 NEXT I
2170 RETURN
2180 REM Travel Sub
2190 IF B2>0 THEN 2220
2200 PRINT "You can not leave starport until all":PRINT "debts are cleared."
2210 RETURN
2220 PRINT "Enter destination star number";
2230 INPUT I
2240 IF I<>S1 THEN 2270
2250 PRINT "You are already at";I
2260 RETURN
2270 IF I=ABS(INT(I)) AND I>0 AND I<N3+1 THEN 2300
2280 PRINT "Invalid star number"
2290 RETURN
2300 S1=I
2310 REM Get Star Trade & Location Data
2320 GOSUB 3510
```

Survival

Survival was written by Stewart F. Rush and originally appeared in *Creative Computing*, January 1982.

It is the year 1991. You have crash landed on the moon and have only 180 minutes of oxygen and 230 units of power remaining. You are at Mare Serenitatis and observe the long, eerie shadows being cast by the distant mountains across the barren landscape. The realization sinks in that you are in big trouble.

Game Description

Survival is an "adventure" type of game. With logic, skill, persistence, and a little bit of luck, it is possible to survive. The action takes place on the surface of the moon where you must assess the situation, explore the surroundings, avoid potential hazards, and gather needed resources.

It is a race against time. Many explorations are required before the total situation is revealed, and the resources and life-threatening hazards are discovered. Only then, can the process of determining an optimum course of action begin.

Once you succeed in surviving, there is then the challenge to plan new survival sequences to minimize the total elapsed time.

The commands to move are NORTH, SOUTH, EAST, WEST, UP, and DOWN. These commands may be spelled out or entered as a single letter—N, S, E, W, U, and D.

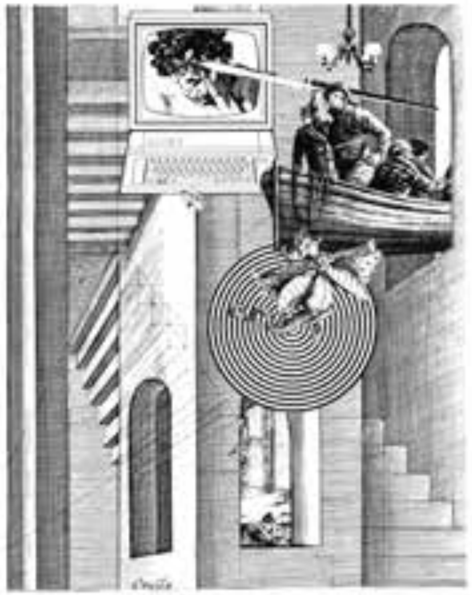
Other commands consist of an action verb followed by a noun. Examples of these commands are:

```
GET ILLUMINATOR
DROP KNIFE
INVENTORY
```

The set of commands is relatively small, hence you may have to try several alternatives to find the one that works. All commands may be abbreviated to the first three letters. To exit the program, you may enter END or QUIT. There is no provision for saving a partially completed game.

How To Write An Adventure Game

by Greg Hassett



As I gazed back at the crystal bridge that I had just crossed, I could hear water rushing nearby. My brass lantern was getting dim, and I knew that I would have to rest soon. The wisps of white mist danced before my eyes as if alive, and a sudden cold chill ran up my spine. I had with me a diamond necklace which I was determined to keep. A nasty dwarf emerged from the gloom. He threw a sharp knife at me! I grabbed my axe and heaved it at him. His body vanished in a cloud of greasy black smoke. My lamp was now out; I would have to search for batteries tomorrow in the dark. So I put my necklace in my small leather sack and called it a day.

I did not lie down on the cavern floor and go to sleep. I merely turned off my home computer. I had been play a game called "Adventure." In this game, you explore a network of caves and pits in search of priceless treasure. This game is not the type of game which is mastered in an hour. It may take days, weeks, or even months to complete an Adventure.

This "original Adventure," developed at Stanford University a few years back by Willie Crowther and Don Woods, required large amounts of disk storage space. This made it very difficult to convert to run on a personal computer. However, other versions of Adventure have sprung up in the past year that are specifically designed to fit in the smaller machines.

To play Adventure, you enter commands to the computer in one- or two-word sentences in what seems to be English. A typical command might be "INSERT COIN" or "GET NECKLACE." To move about, you use commands such as "GO NORTH" or enter a new "location," and a new room description will be displayed. An example of such a description might be:

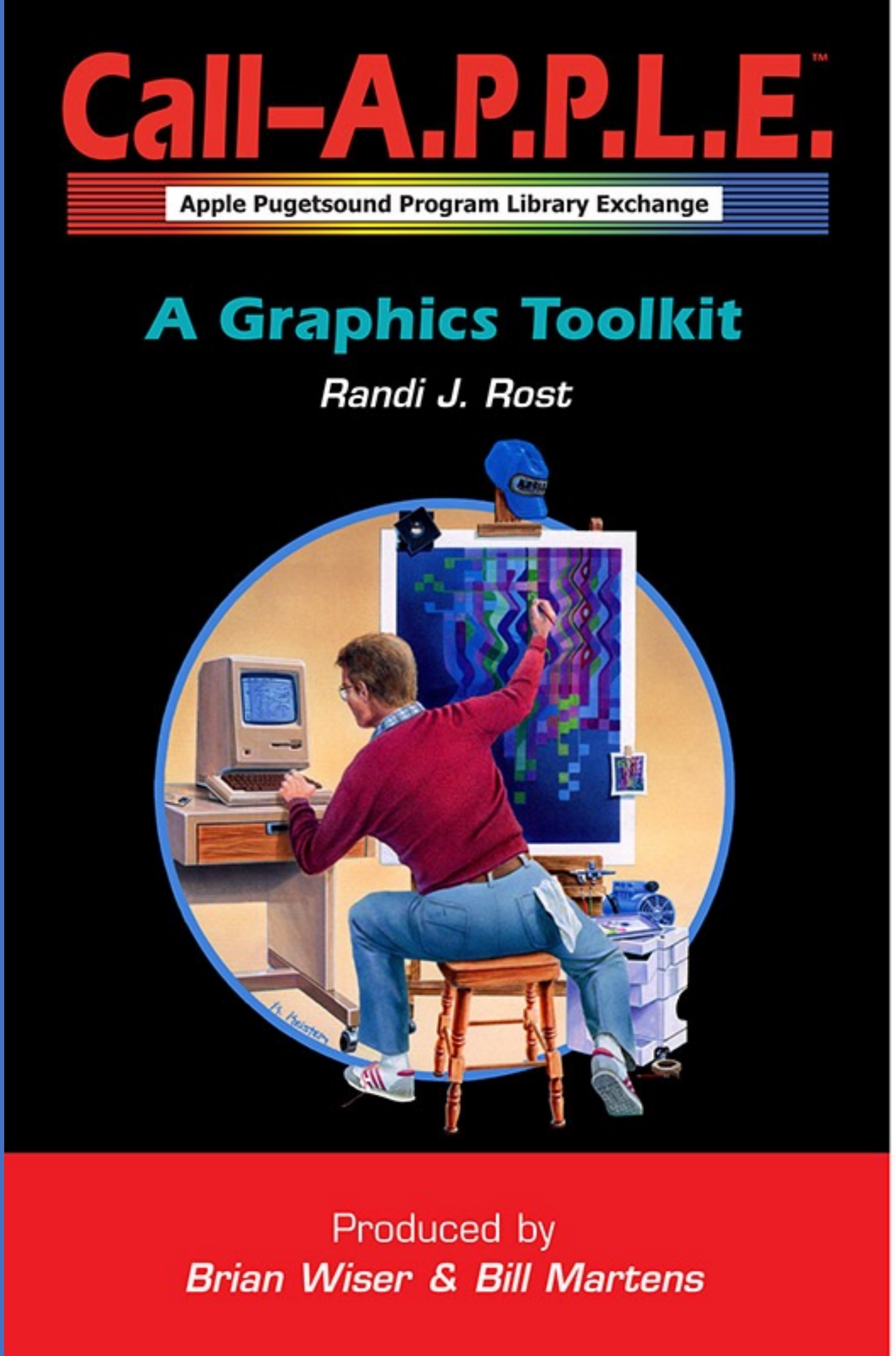
I AM IN A RADIANT CAVERN FORTY FEET HIGH. THE WALLS AND FLOORS ARE MADE OF SMOOTH MARBLE. THE POOLS OF CLEAR WATER ON THE FLOOR INDICATE AN OPENING HIGH ABOVE ME. UP ON THE CEILING GLOWS AN EERIE RED LIGHT.

AROUND ME I SEE: POOLS OF WATER. SMALL PLASTIC VIAL....

Later on in the game, the vial might come in handy for holding some liquid, etc., so in this situation it might be wise to "GET VIAL."

The one thing that I feel makes Adventuring so interesting is the clues that are given as you explore.

Knowing that clues exist is one thing; isolating them and figuring out what they mean is quite another. In Adventure, clues exist *everywhere*. They are in the room descriptions, the object descriptions. Let's say you enter a room where there are many stalactites, but no stalagmites on the floor. This in itself is clue. If you think about it, stalagmites could be worn off if creatures lived there and walked through the cavern.



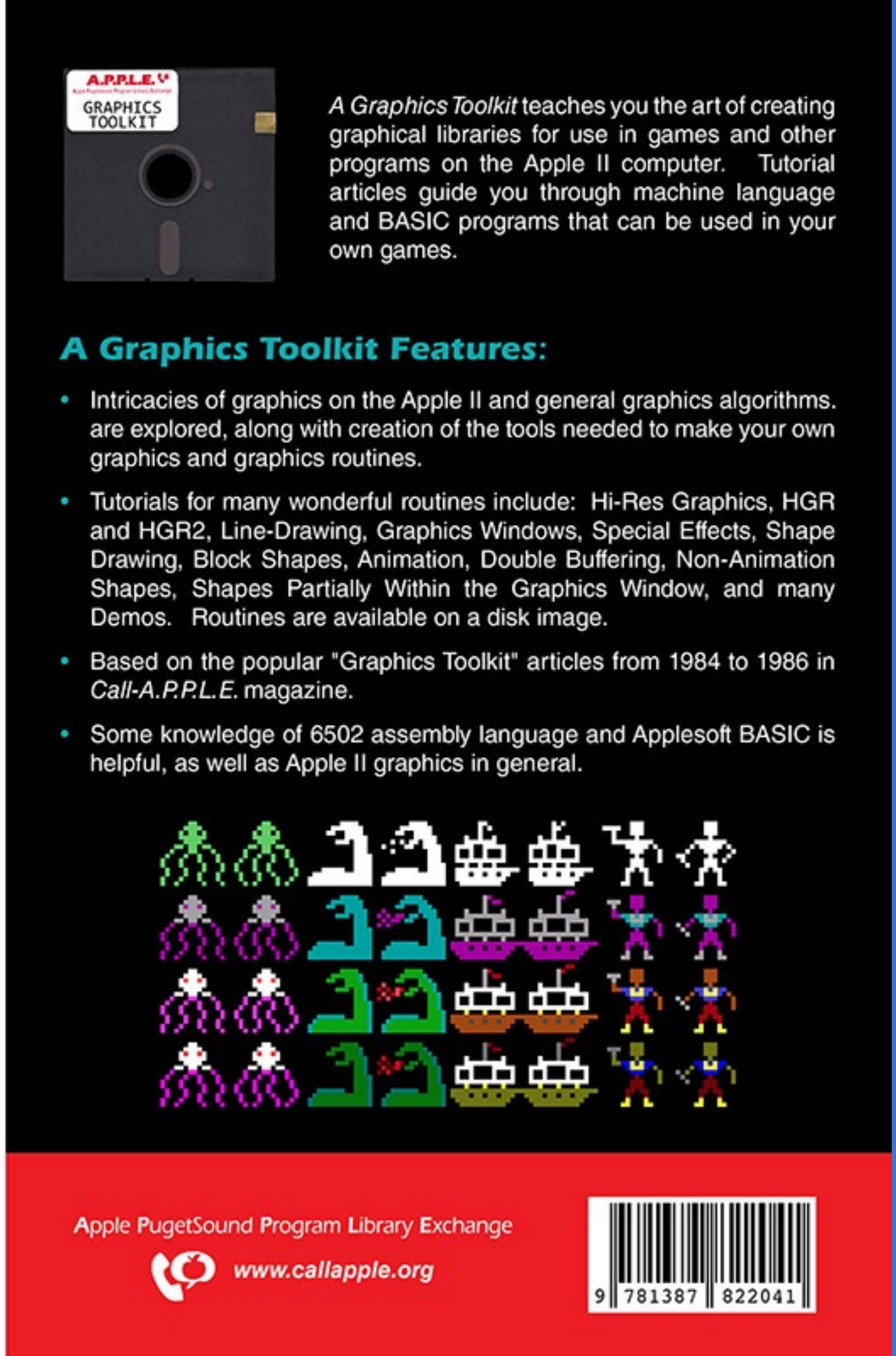
Call-A.P.P.L.E.™

Apple Pugetsound Program Library Exchange

A Graphics Toolkit

Randi J. Rost

Produced by
Brian Wiser & Bill Martens



A Graphics Toolkit teaches you the art of creating graphical libraries for use in games and other programs on the Apple II computer. Tutorial articles guide you through machine language and BASIC programs that can be used in your own games.

A Graphics Toolkit Features:

- Intricacies of graphics on the Apple II and general graphics algorithms. are explored, along with creation of the tools needed to make your own graphics and graphics routines.
- Tutorials for many wonderful routines include: Hi-Res Graphics, HGR and HGR2, Line-Drawing, Graphics Windows, Special Effects, Shape Drawing, Block Shapes, Animation, Double Buffering, Non-Animation Shapes, Shapes Partially Within the Graphics Window, and many Demos. Routines are available on a disk image.
- Based on the popular "Graphics Toolkit" articles from 1984 to 1986 in *Call-A.P.P.L.E.* magazine.
- Some knowledge of 6502 assembly language and Applesoft BASIC is helpful, as well as Apple II graphics in general.

Apple PugetSound Program Library Exchange

www.callapple.org

VI. A Tool or Three

Now that we have the groundwork laid out let's get to work. The HGR and HGR2 commands perform very similar functions in Applesoft, HGR displays and clears High-Res page one and sets the current plotting color to black ($00), HGR2 does the same thing but on High-Res page two, The only other difference between the two is that HGR sets mixed text and graphics mode, and HGR2 sets full-screen graphics mode. Another potentially useful function missing from Applesoft is one that would clear the current High-Res page to any desired color. Is it possible that we could kill all three birds with one stone?

Listing 1.3 shows that the answer is yes. By carefully coding the routine and providing three separate entry points, we can accomplish all three tasks with the same routine. Again for consistency, we'll choose to make both the HGR and HGR2 routines set full-screen graphics mode. Let's take a close look at the routines contained in Listing 3. The instructions at the HGR2 entry point cause page two of High-Res graphics to be displayed and the A-register to be loaded with a value of $40 before branching around the code for the HGR entry point. For HGR, a value of $20 is loaded into A and High-Res page one is displayed.

We have now reached the label HCLEAR with either a $20 or a $40 in the A register. Recall that High-Res page one starts at $2000 and page two starts at $4000. The value in A then is simply the high byte of the start address of the graphics page to be manipulated. This value is stored immediately in the location HGRPG. HGRPG is a very special page zero location that will always point to the High-Res page currently being read or written (but not necessarily displayed). It will be used extensively by other routines. Next we tweak all the switches necessary to set up full page, High-Res graphics, and then load A with the color to be used in filling the screen black.

Now we're at the third of our function entry points. If we jump directly to BKGND with the color (0-7) to be used in filling the screen in the A register, we can skip all the set-up work performed by HGR and HGR2 and take advantage of the same code they use to fill the screen with zeros (black0). At this point we transfer the contents of A into Y and do an indirect indexed load of the A-register using COLTABL. COLTABL is another page zero location, set up by the INIT routine to point to the start of the color mask table. This loads the appropriate color mask byte for the specified color from our

lookup table. The color mask value becomes our current plotting color and is stored in two locations, COLOR and COLI where it can be found later. It's also stored in the location SHFTCHK where it can be found and used by the SHFTCOL routine.

Remember when we were discussing the on-off-on bit patterns earlier? We discovered that there are seven pixels per byte, right? This implies that we need to be careful in order to maintain a pattern of on-off-on through several bytes. If we have 1010101 for the first byte and for the second byte, the concatenated result is 10101011010101, clearly not what we want. What we need is a 1010101 followed by a 0101010 and so on. This one-bit shift is necessary to maintain the proper color masking pattern for all the bytes in a row and is precisely the function performed by the SHFTCOL routine.

Upon returning from SHFTCOL, the shifted color mask is loaded from SHFTCHK and stored in COL2. We now have everything we need and the rest of the routine is straightforward. A big loop is set up and all we need to do is copy COL1 into all the even bytes and COL2 into all the odd bytes of the High-Res screen indicated by HGRPG.

Listing 1.3 - HGR and HGR2 Subroutines

```
*
* The HGR and HGR2 subroutines function almost
*  identically to their Applesoft counterparts.
* No arguments need be passed to HGR and HGR2.
* To use BKGND, store the color (0-7) to clear the
*  screen to in A, then JSR BKGND.
*
 ORG $85B

SHFTCHK = $1C
TEMPPTR = $26
COLOR = $E4
HGRPG = $E6
COLTABL = $FE
DISPGR = $C050
NOMIX = $C052
DISPG1 = $C054
DISPG2 = $C055
```

Bresenham's Algorithm

As everyone who's tinkered at the level of bits and bytes knows, integer arithmetic is much faster than floating point arithmetic on machines with no floating point hardware. In addition, adds and subtracts are significantly faster than multiplies and divides on machines with no multiply or divide instructions. Many microcomputers, including the Apple II, are based on microprocessors that contain neither multiply and divide instructions nor floating point capability. J.E. Bresenham came up with an algorithm that runs significantly faster than the simple DDA algorithm on such machines since it requires only additions, subtractions, and multiplication by two (which can be performed with a rotate instruction).

Bresenham's algorithm is like the simple DDA in that each iteration changes one of the coordinates by 1 or -1. For each iteration, the other coordinate mayor may not change, depending on the error term maintained by the algorithm. For cases where dx < dy, this error term is initialized to abs(dx)*2 -dy, and for dx >=dy it is initialized to abs(dy)*2 dx. This forces the error term to be negative. Each time a point is plotted a value of dx*2 (for the first case) or dy*2 (for the second case) is added to the error term. When the error term becomes positive, it is time to increment the other coordinate as well. Listing 3.2 is an Applesoft implementation of Bresenham's algorithm.

Listing 3.2 - Bresenham's Algorithm in Applesoft

```
100  TEXT : HOME : VTAB 10
110  PRINT "INPUT X1,Y1";: INPUT X1,Y1
120  PRINT "INPUT X2,Y2";: INPUT X2,Y2
130  HGR : HCOLOR= 3
140 DX = X2 - X1:XINC = 1
150  IF DX >  = 0 THEN 170
160 DX =  - DX:XINC =  - 1
170 DY = Y2 - Y1:YINC = 1
180  IF DY >  = 0 THEN 200
190 DY =  - DY:YINC =  - 1
200  IF DX < DY THEN 290
205  REM  ** CASE WHERE DX >= DY **
210 ERR = DY * 2 - DX
220  HPLOT X1,Y1
```

Special Effects

The routine in Listing 4.2 is of questionable value, but add it to your library of routines anyway and someday you may find a use for it. The routine is based on the peculiarities of the Apple's graphics architecture we just mentioned. There are three ways we can modify a byte of graphics memory:

1. Invert the color flag
2. Invert the seven pixel values in the byte
3. Invert the color flag and the seven pixel values in the byte

The routine will perform one of the three "complements" on the current graphics window depending on the value passed in the A-register. The beauty is that a second complement of the same type will restore the screen to its original state. You may find this useful for explosions or "glitter" effects in title pages.

The CMPLMNT routine utilizes the sometimes useful, but always debatable technique of self-modifying cost. Since the three types of complements differ in only two EOR instructions, we can save a little space using this technique. The first part of the CMPLMNT routine stores the proper EOR instructions in the main loop of the routine depending on the value passed in the A-register. Self-modifying can be a dangerous thing (hard to find bugs!) but it can also save you a headache or some space once in a while.

Listing 4.2 - Questionable Routine

```
*  CMPLMNT WILL COMPLEMENT ALL THE COLORS
*  IN THE CURRENT GRAPHICS WINDOW. CALLING CMPLMNT
*  AGAIN WILL REVERT EVERYTHING BACK TO
*  ITS ORIGINAL VALUE.  PASS A 0, 1, OR 2 IN THE A
*  REGISTER DEPENDING ON WHICH TYPE OF COMPLEMENT IS
*  DESIRED.
*
 ORG $AA2
YTABL EQU $EC
YTABH EQU $EE
TEMPPTR EQU $26
HGRPG EQU $E6
LFTXBYTE EQU $A68
```

Call–A.P.P.L.E.™

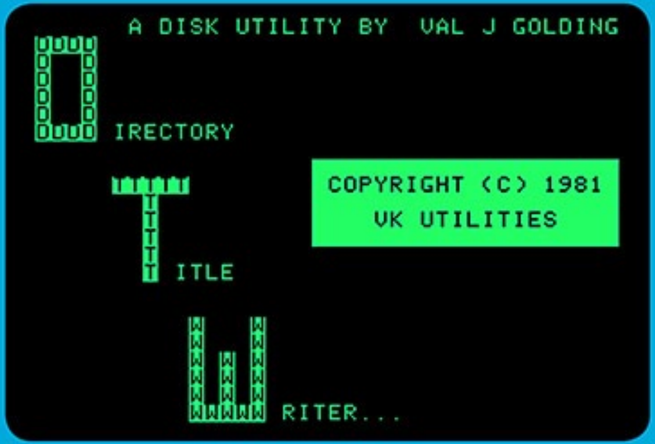
Apple Pugetsound Program Library Exchange

DIRECTORY TITLE WRITER...

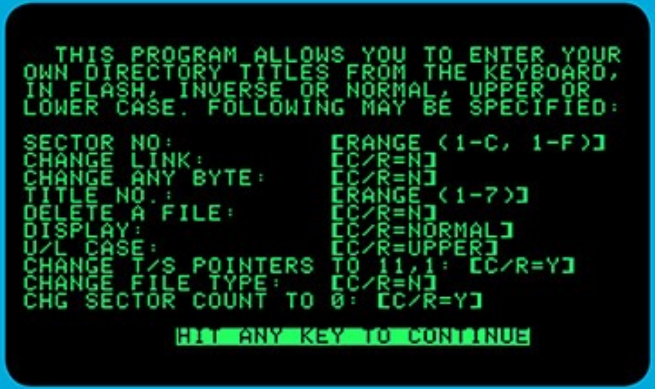
Val J. Golding

Produced by
Brian Wiser & Bill Martens

Directory Title Writer is a DOS 3.3 utility that lets you create elaborate and fancy file names on your Apple II disk catalog. Type inverse, flash, and lower case directly from the keyboard.



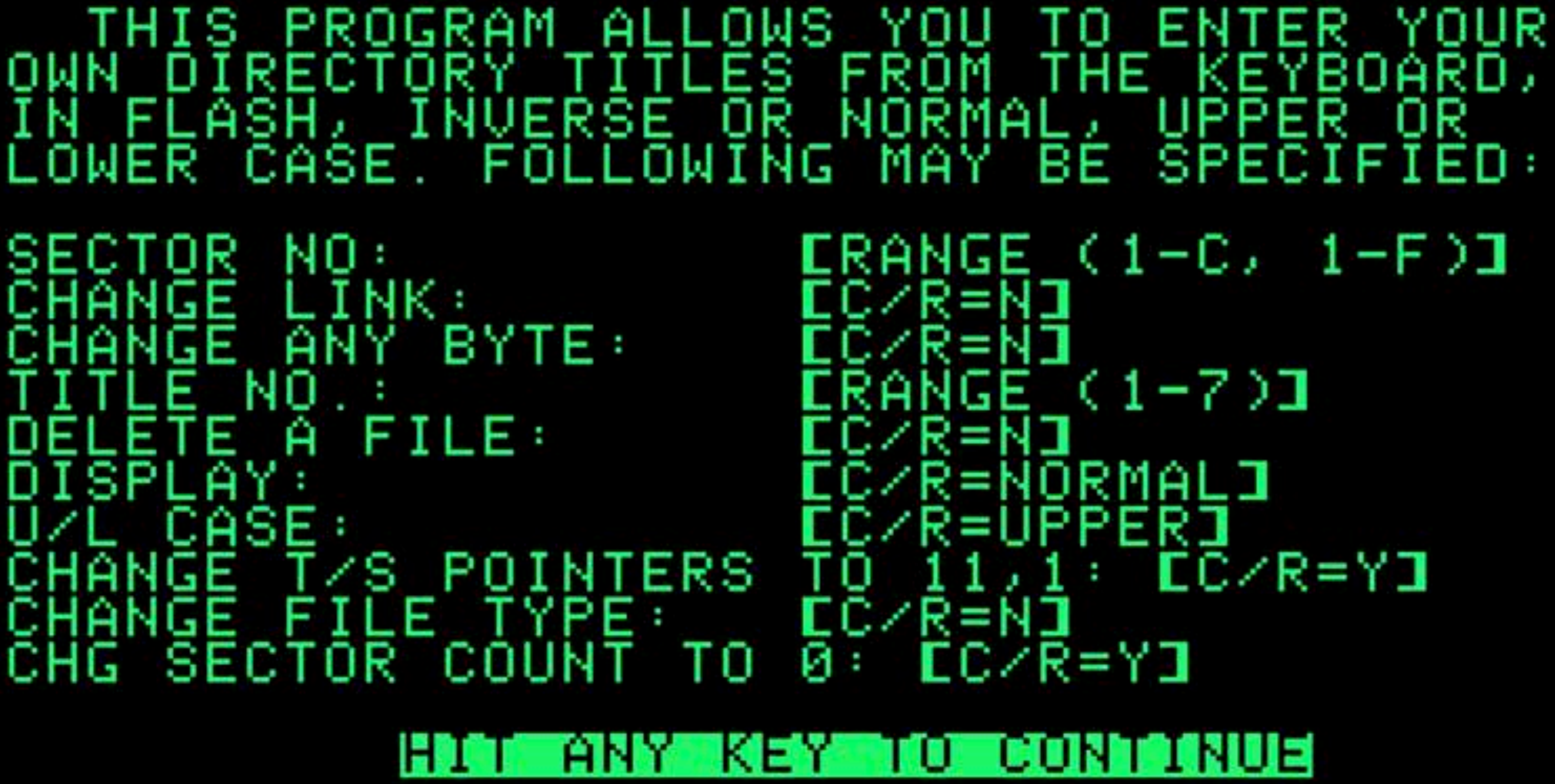
```
          A DISK UTILITY BY  VAL J GOLDING
D IRECTORY
                         COPYRIGHT (C) 1981
                           VK UTILITIES
   T ITLE

      W RITER...
```



```
  THIS PROGRAM ALLOWS YOU TO ENTER YOUR
OWN DIRECTORY TITLES FROM THE KEYBOARD,
IN FLASH, INVERSE OR NORMAL, UPPER OR
LOWER CASE. FOLLOWING MAY BE SPECIFIED:

SECTOR NO:             [RANGE (1-C, 1-F)]
CHANGE LINK:           [C/R=N]
CHANGE ANY BYTE:       [C/R=N]
TITLE NO.:             [RANGE (1-7)]
DELETE A FILE:         [C/R=N]
DISPLAY:               [C/R=NORMAL]
U/L CASE:              [C/R=UPPER]
CHANGE T/S POINTERS TO 11,1: [C/R=Y]
CHANGE FILE TYPE:      [C/R=N]
CHG SECTOR COUNT TO 0: [C/R=Y]

       HIT ANY KEY TO CONTINUE
```

Apple PugetSound Program Library Exchange
www.callapple.org

9 781387 893126



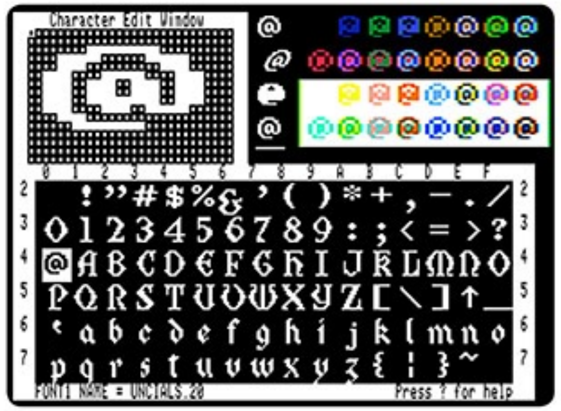
```
  THIS PROGRAM ALLOWS YOU TO ENTER YOUR
OWN DIRECTORY TITLES FROM THE KEYBOARD,
IN FLASH, INVERSE OR NORMAL, UPPER OR
LOWER CASE. FOLLOWING MAY BE SPECIFIED:

SECTOR NO:                [RANGE (1-C, 1-F)]
CHANGE LINK:              [C/R=N]
CHANGE ANY BYTE:          [C/R=N]
TITLE NO.:                [RANGE (1-7)]
DELETE A FILE:            [C/R=N]
DISPLAY:                  [C/R=NORMAL]
U/L CASE:                 [C/R=UPPER]
CHANGE T/S POINTERS TO 11,1: [C/R=Y]
CHANGE FILE TYPE:         [C/R=N]
CHG SECTOR COUNT TO 0: [C/R=Y]

          HIT ANY KEY TO CONTINUE
```

Call-A.P.P.L.E.™

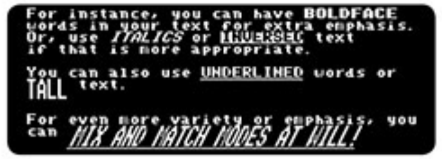
Apple Pugetsound Program Library Exchange

Robert C. Clardy & Alan B. Clark

Produced by
Brian Wiser & Bill Martens



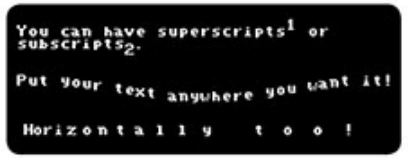
Generate text in different typefaces, sizes, colors, and angles from your Apple II programs on the Double Hi-Resolution screen. Features an easy-to-use editor for creation of your own unique character sets, with 46 fonts included. Assumes a working knowledge of Applesoft BASIC.



For instance, you can have BOLDFACE words in your text for extra emphasis. Or, use *ITALICS* or INVERSE text if that is more appropriate.

You can also use UNDERLINED words or TALL text.

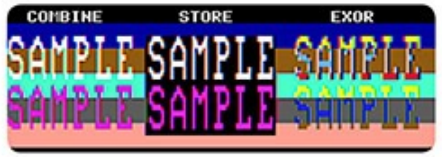
For even more variety or emphasis, you can *MIX AND MATCH MODES AT WILL!*

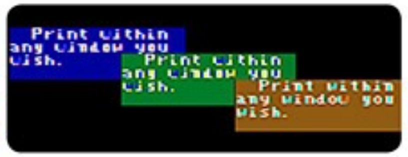


You can have superscripts^1 or subscripts$_2$.

Put your text anywhere you want it!

Horizontally too!

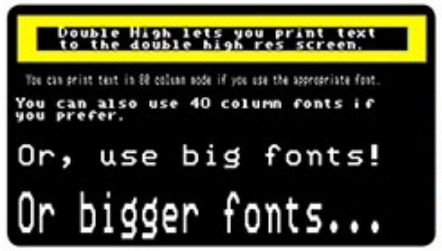




Print within any window you wish.

Print within any window you wish.

Print within any window you wish.



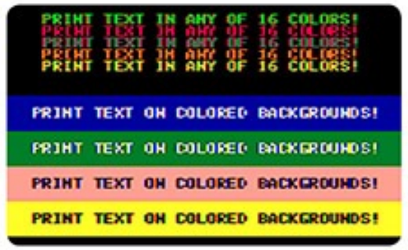
Double High lets you print text to the double high res screen.

You can print text in 80 column mode if you use the appropriate font.

You can also use 40 column fonts if you prefer.

Or, use big fonts!

Or bigger fonts...

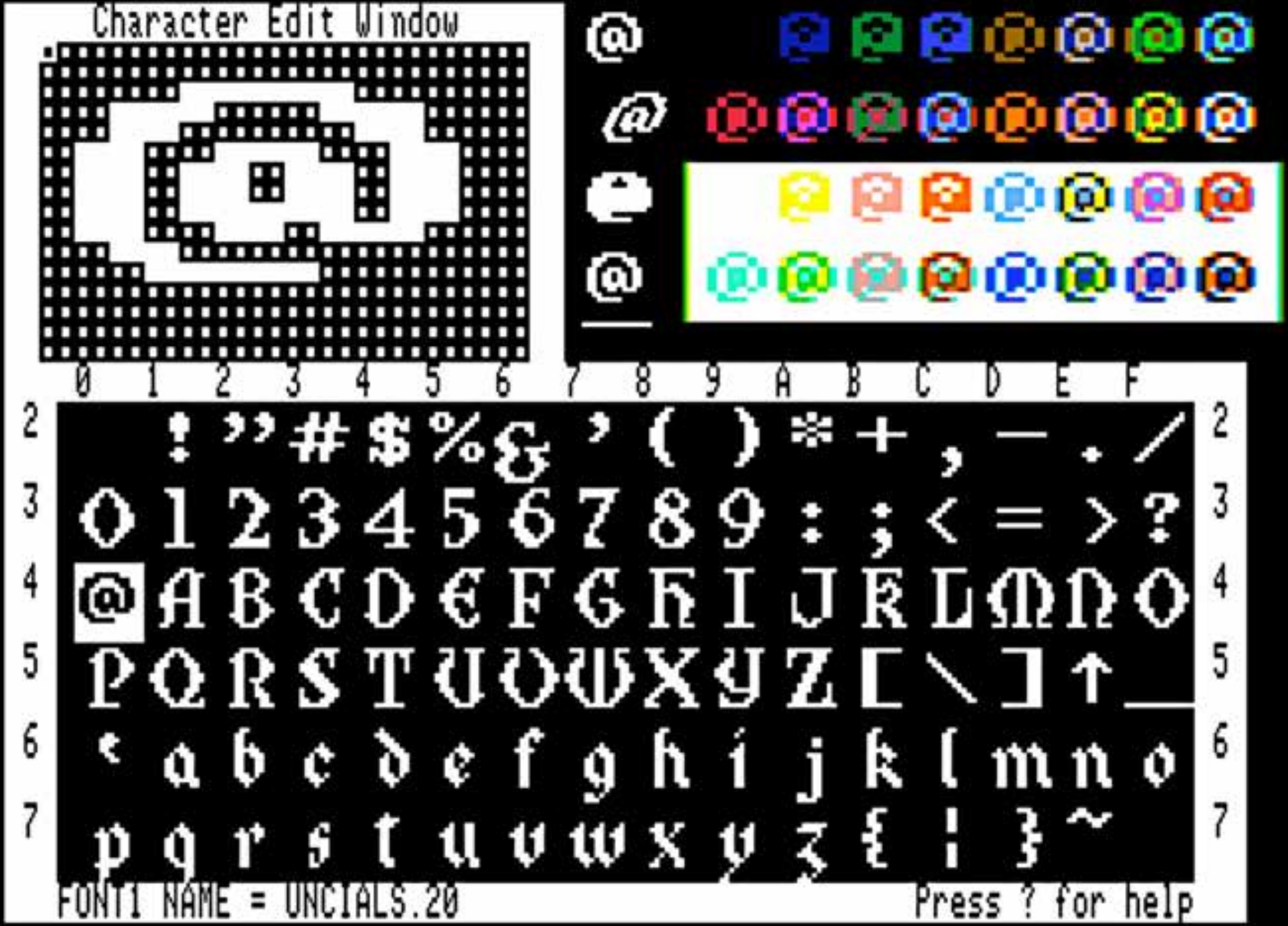


PRINT TEXT IN ANY OF 16 COLORS!
PRINT TEXT IN ANY OF 16 COLORS!
PRINT TEXT IN ANY OF 16 COLORS!
PRINT TEXT IN ANY OF 16 COLORS!
PRINT TEXT IN ANY OF 16 COLORS!

PRINT TEXT ON COLORED BACKGROUNDS!
PRINT TEXT ON COLORED BACKGROUNDS!
PRINT TEXT ON COLORED BACKGROUNDS!
PRINT TEXT ON COLORED BACKGROUNDS!

Apple PugetSound Program Library Exchange
www.callapple.org

9 781387 886975


Character Edit Window

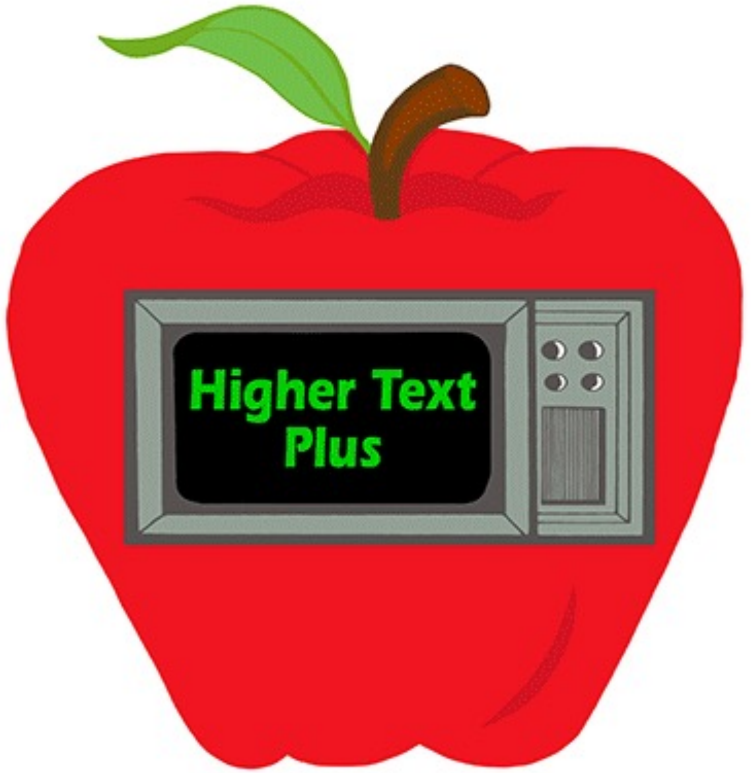
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	2
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	3
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	4
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	_	5
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	6
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		7

FONT1 NAME = UNCIALS.20

Press ? for help

Call–A.P.P.L.E.™

Apple Pugetsound Program Library Exchange



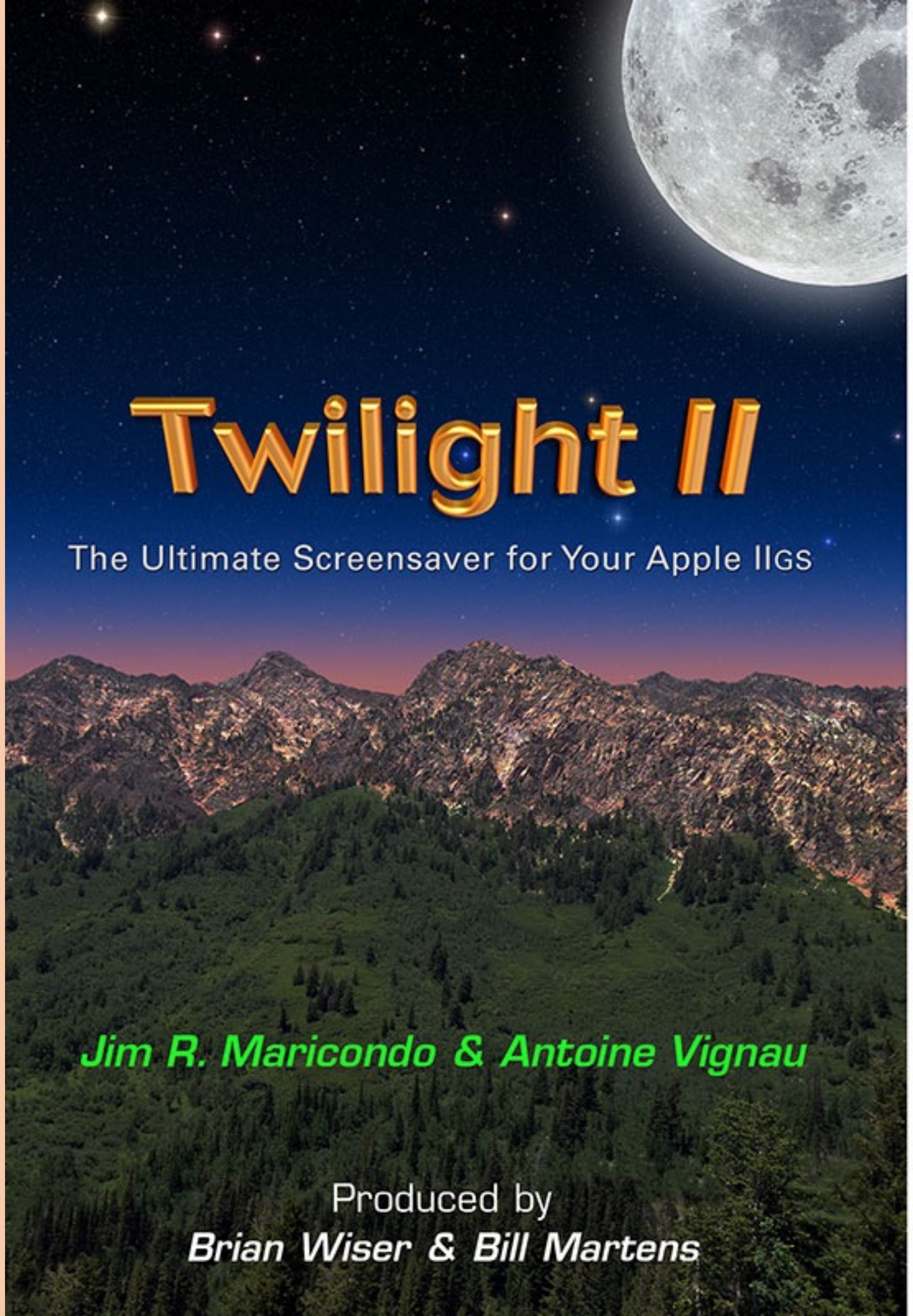
Ron Aldrich & Darrell Aldrich

Produced by
Brian Wiser & Bill Martens

Generate upper and lower case text in a variety of typefaces, sizes, and colors from your programs for display on the Apple II hi-resolution screen. Features an easy-to-use editor for creation of your own unique character sets, with over 45 fonts included. Several additional utilities enhance your creation of hi-res displays. This program assumes a working knowledge of Applesoft BASIC.

Apple PugetSound Program Library Exchange
www.callapple.org

9781716727245

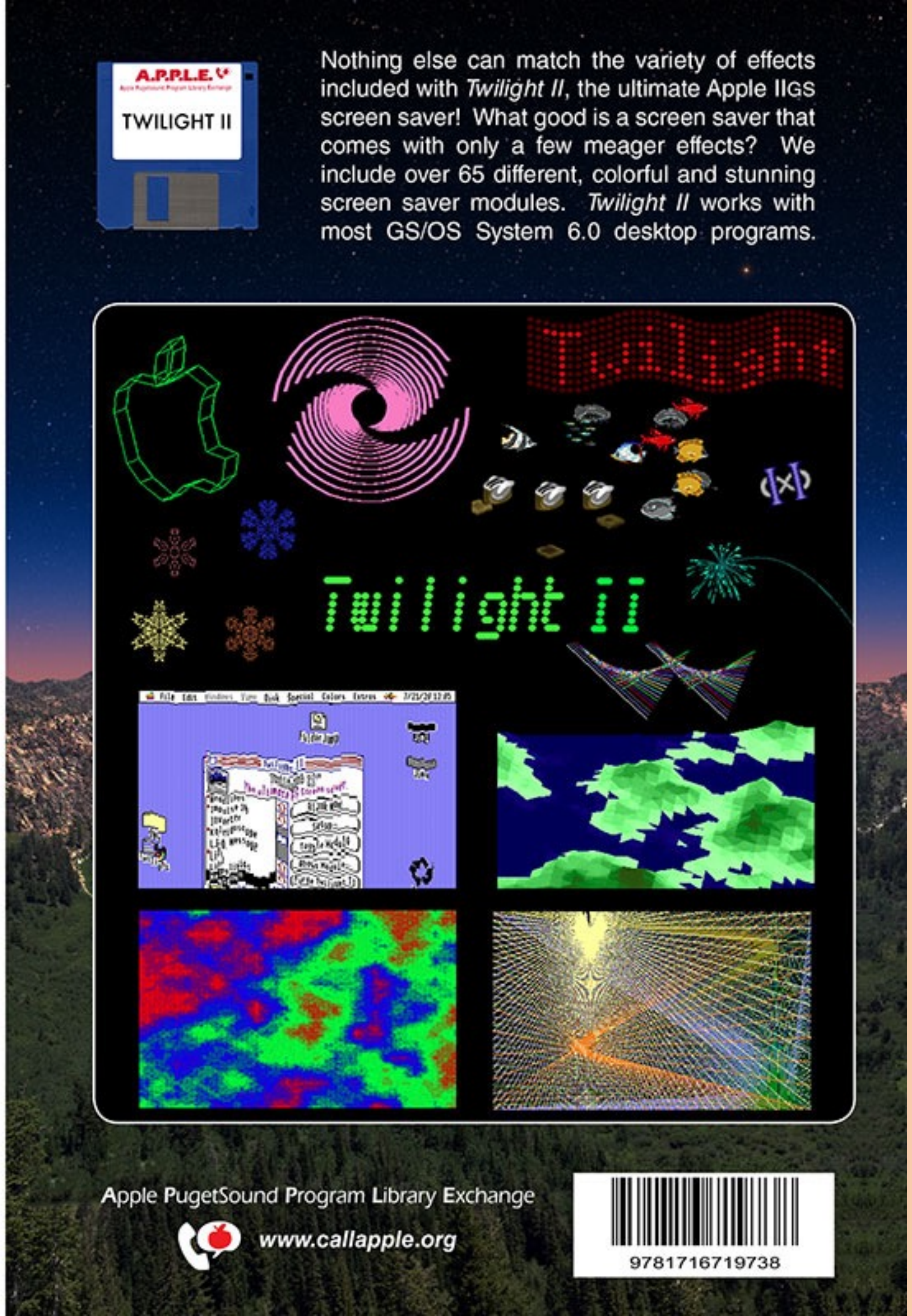


Twilight II

The Ultimate Screensaver for Your Apple IIGS

Jim R. Maricondo & Antoine Vignau

Produced by
Brian Wiser & Bill Martens



Nothing else can match the variety of effects included with *Twilight II*, the ultimate Apple IIGS screen saver! What good is a screen saver that comes with only a few meager effects? We include over 65 different, colorful and stunning screen saver modules. *Twilight II* works with most GS/OS System 6.0 desktop programs.

Apple PugetSound Program Library Exchange

www.callapple.org

9781716719738

SIGNATURE GS

by Duilio Proni

Quality Computers

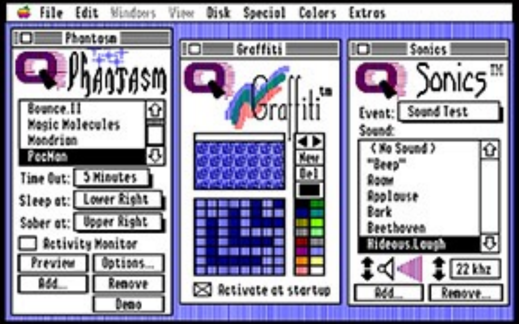
4 MODULES THAT LET YOU CREATE A COMPUTING ENVIRONMENT UNIQUELY YOUR OWN

Produced by
Brian Wiser & Bill Martens

Signature GS is a collection of Control Panels (CDEVs) that make your Apple IIGS easier and more fun to use by adding a Screen Blanker, a Desktop Pattern Editor, a Sound Selector, and a Boot Setup Utility to GS/OS version 5.0 or later.

Signature GS Features:

- ***Phantasm*** – A utility that prolongs screen life and looks good doing it. When your screen has been still too long, it can become damaged. *Phantasm* gives your screen something to do through a variety of fun, attractive screen activities, including "Magic Molecules" and "Slimy Slugs." Plus, you control when it comes on and goes off.
- ***Graffiti*** – There's nothing wrong with the IIGS desktop, except it's boring. *Graffiti* gives you a variety of colorful desktop patterns to choose from, or you can create your own with the built-in desktop painter.
- ***Sonics*** – The "S" in IIGS stands for sound. *Sonics* lets you customize your IIGS sounds, and assign special sounds for specific functions. And these aren't just beeps – they're fun and entertaining. Imagine your Apple actually "belching" out a floppy disk.
- ***Boot Master*** – Changing the configuration of all your IIGS drivers, and system components can be a pain. *Boot Master* makes it easy by putting all the controls for your IIGS drivers, CDAs, NDAs, FSTs, INITs, and CDEVs on one master control panel.



Bounce.II
Magic Molecules
Mondrian
PacMan

Time Out: 5 Minutes
Sleep at: Lower Right
Sober at: Upper Right
☐ Activity Monitor

☒ Activate at startup

Event: Sound Test
Sound:
‹ No Sound ›
"Beep"
Boom
Applause
Bark
Beethoven
Hideous.Laugh
22 khz

Apple PugetSound Program Library Exchange

www.callapple.org

ISBN 978-0-359-72017-0 90000
9 780359 720170

SIX PACK

BILL TUDOR

A REFRESHING COLLECTION OF UTILITIES FOR APPLE IIGS SYSTEM 6

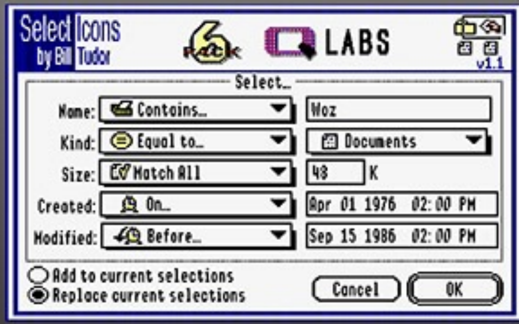
Quality Computers

Produced by
Brian Wiser & Bill Martens

Six Pack, the first collection of Apple IIGS Finder Extensions, adds 13 new features to System 6. Just click the icons you want to work with, then select the *Six Pack* functions you want to perform from the Extras menu.

Six Pack Features:

- ***Alarm Clock*** – See time in a window or menu bar and provide alerts.
- ***CDEV Alias*** – Add any Control Panel directly to the Apple menu.
- ***CPU & Memory Use*** – See how your computer's resources are used.
- ***Crypt*** – Save your data from prying eyes!
- ***File Compare*** – Visually compare files for differences.
- ***File Peeker*** – Preview the contents of graphics, text, and sound files.
- ***HotKeys*** – Add dozens of functions to your function keys.
- ***IR & OpenSesame*** – Save memory using CDAs/NDAs, Inits, Drivers.
- ***MoreInfo*** – See and change file data like actual / auxiliary file types.
- ***Print Catalog*** – Print a directory listing with filename, filetype...
- ***Select Icons*** – Based on partial filename, filetype, modification date...
- ***Super DataPath*** – Remembers a default data directory for programs.
- ***Workset*** – Remembers a group of files and opens them all at once.



Select...

Name:	Contains...	Woz
Kind:	Equal to...	Documents
Size:	Match All	48 K
Created:	On...	Apr 01 1976 02:00 PM
Modified:	Before...	Sep 15 1986 02:00 PM

○ Add to current selections
● Replace current selections

Apple PugetSound Program Library Exchange
www.callapple.org

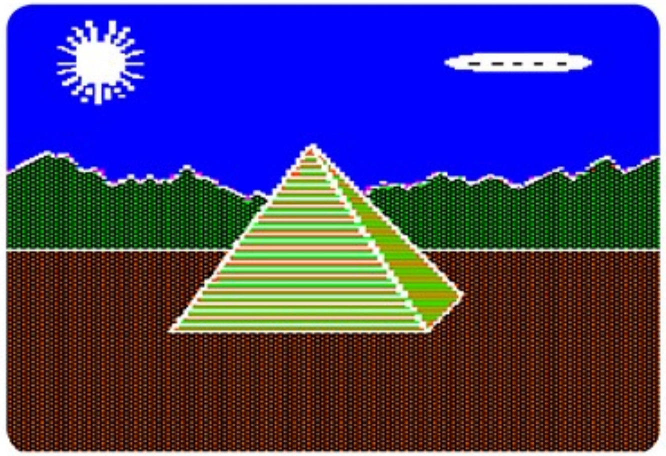
ISBN 978-0-359-74615-6 90000
9 780359 746156

Call–A.P.P.L.E.™

Apple Pugetsound Program Library Exchange

The Etch-a-Sketch

and Other Fun Programs



Brian Wiser

Produced by
Brian Wiser & Bill Martens

The Etch-a-Sketch and Other Fun Programs is a collection of Apple II software programmed by a student in the 1980s. BASIC and machine language programming were once taught in schools, and here you'll find a variety of useful graphics, education, utility, and game software. The author also shares stories about his programming experiences in school.

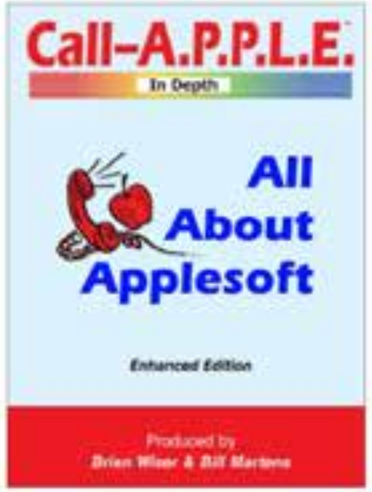
Features 13 Programs Including:

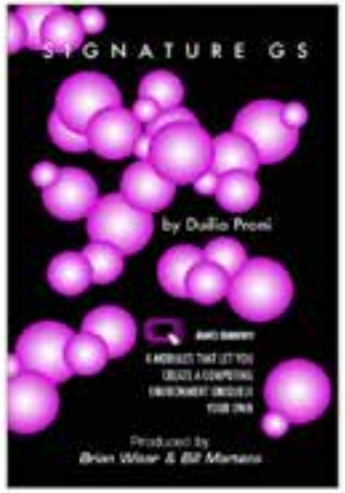
- ***The Etch-a-Sketch*** – fun drawing with keyboard, joystick, and sound.
- ***The Apple*** – the six color logo in beautiful lo-res.
- ***Annual Graph Matrix*** – graph monthly amounts for one year.
- ***Compound Interest*** – calculate investment interest over time.
- ***States & Capitals*** – learn about the U.S. through quizzes.
- ***Access Code*** – an alarmed gatekeeper for your disks.
- ***H*** – a powerful HELLO program for launching files in DOS 3.3.
- ***Random Access Filer*** – a simple text database for contacts.
- ***Tunnel Race*** – dodge obstacles through a text-based cavern.

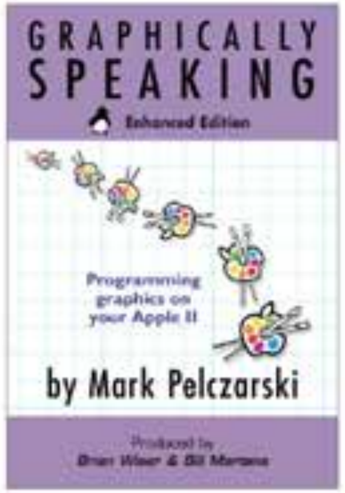
Apple PugetSound Program Library Exchange
www.callapple.org

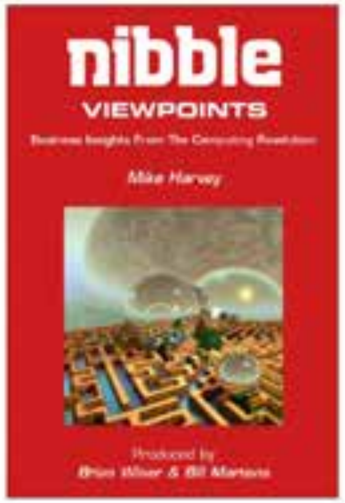
ISBN 978-0-359-74406-0 90000

CallAPPLE.org/books

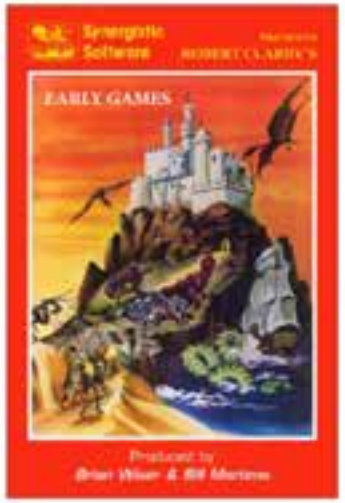


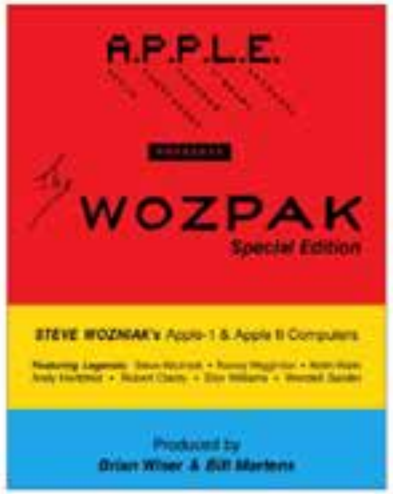


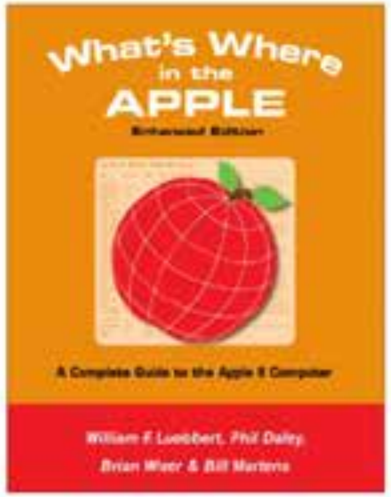



nibble
VIEWPOINTS
Business Insights From The Computing Revolution
Mike Harvey
Produced by
Brian Wiser & Bill Martens

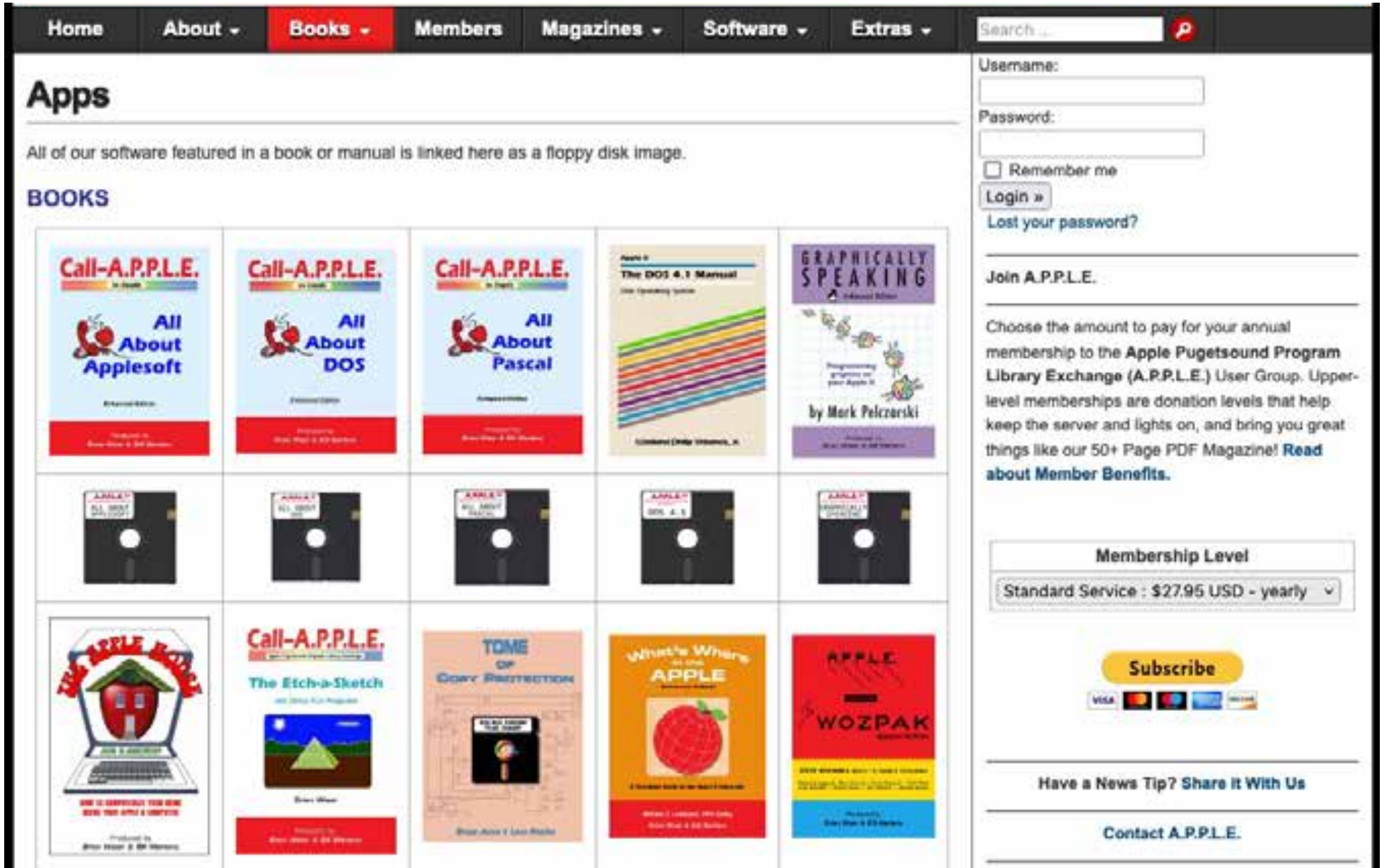
CYBER JACK
The Adventures of
Robert Clardy and Synergistic Software
Produced by
Brian Wiser & Bill Martens




A.P.P.L.E.
WOZPAK
Special Edition
STEVE WOZNIAK's Apple-1 & Apple II Computers
Produced by
Brian Wiser & Bill Martens


What's Where in the APPLE
Enhanced Edition
A Complete Guide to the Apple II Computer
William F. Luebbert, Phil Daley
Brian Wiser & Bill Martens

CallAPPLE.org/apps



Apps

All of our software featured in a book or manual is linked here as a floppy disk image.

BOOKS

Username:

Password:

Remember me

Login »

Lost your password?

Join A.P.P.L.E.

Choose the amount to pay for your annual membership to the **Apple Pugetsound Program Library Exchange (A.P.P.L.E.)** User Group. Upper-level memberships are donation levels that help keep the server and lights on, and bring you great things like our 50+ Page PDF Magazine! **Read about Member Benefits.**

Membership Level

Standard Service : $27.95 USD - yearly

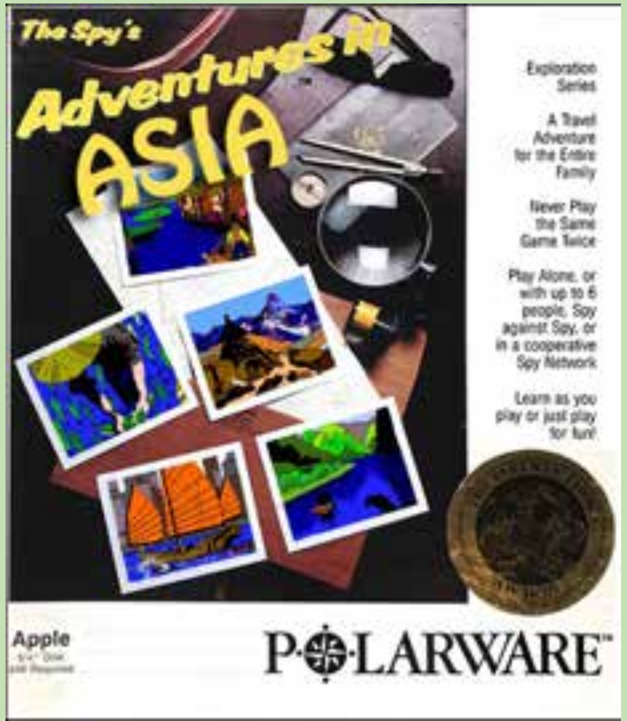
Subscribe

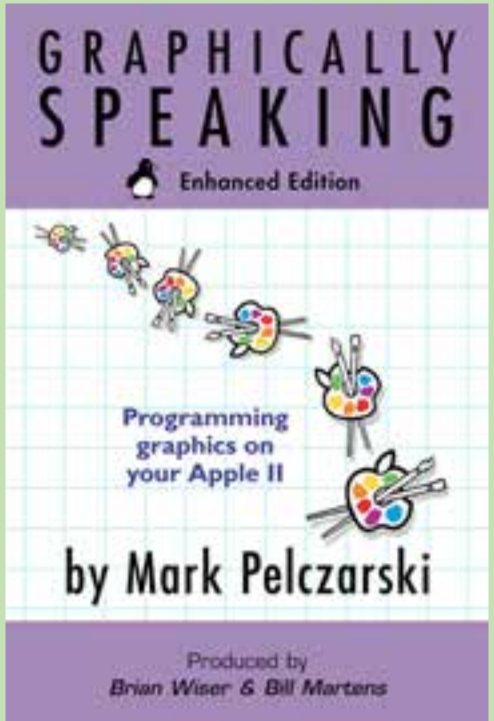
Have a News Tip? Share It With Us

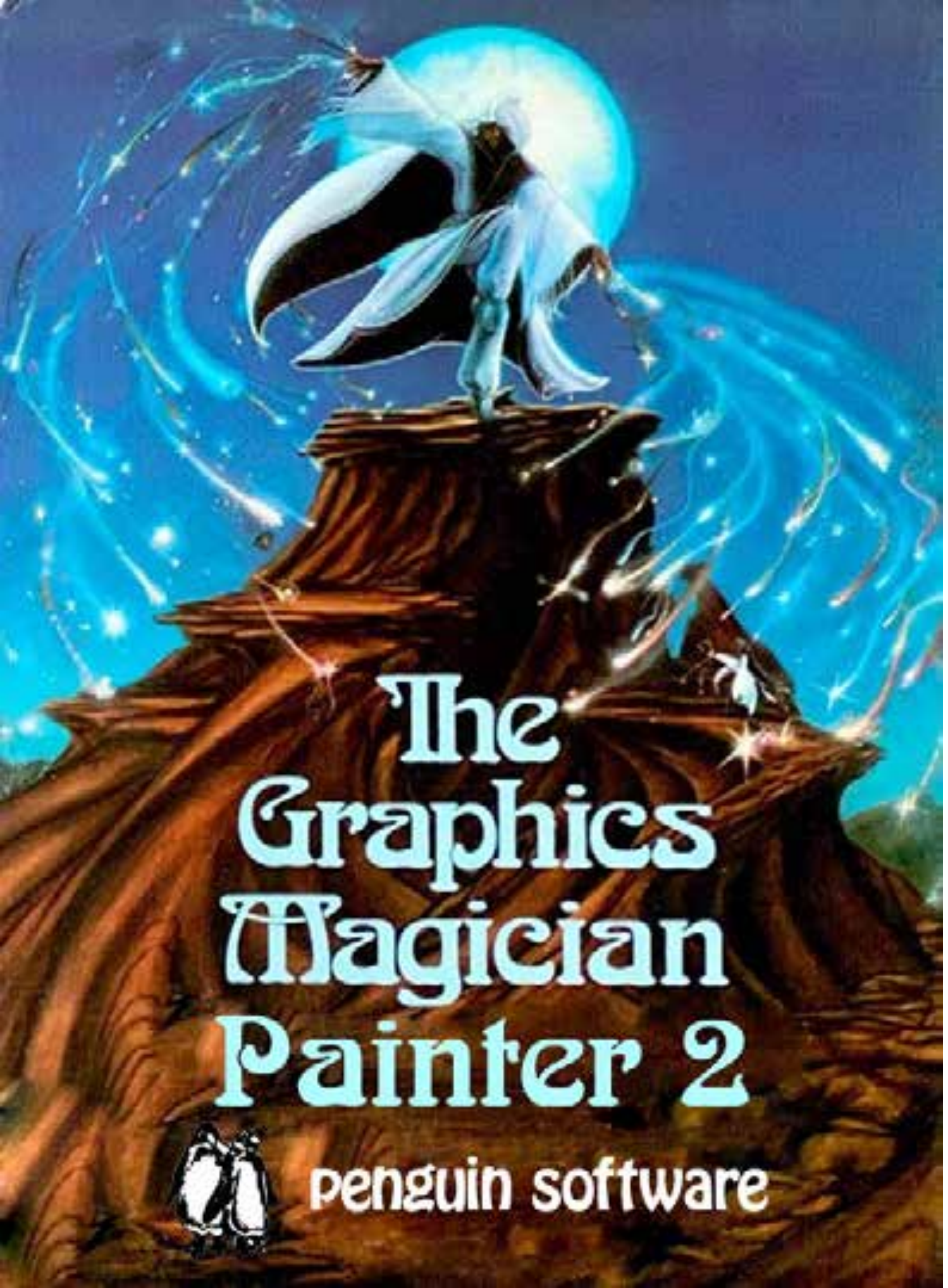
Contact A.P.P.L.E.

penguin software

CallAPPLE.org/penguin


The Spy's
Adventures in
ASIA
Exploration Series
A Travel Adventure for the Entire Family
Never Play the Same Game Twice
Play Alone, or with up to 6 people. Spy against Spy, or in a cooperative Spy Network
Learn as you play or just play for fun!
Apple
POLARWARE™





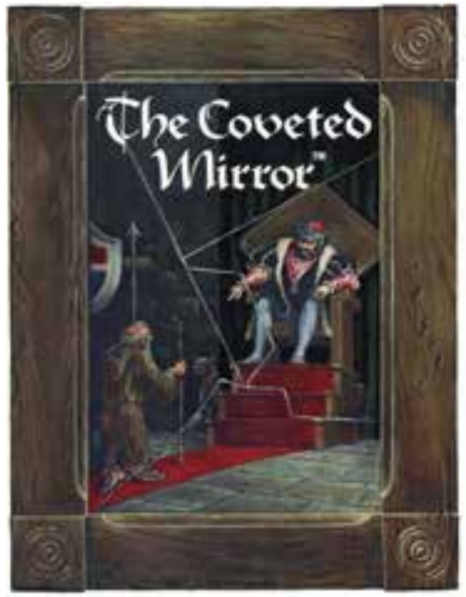
Graphics Magician Painter 2.02

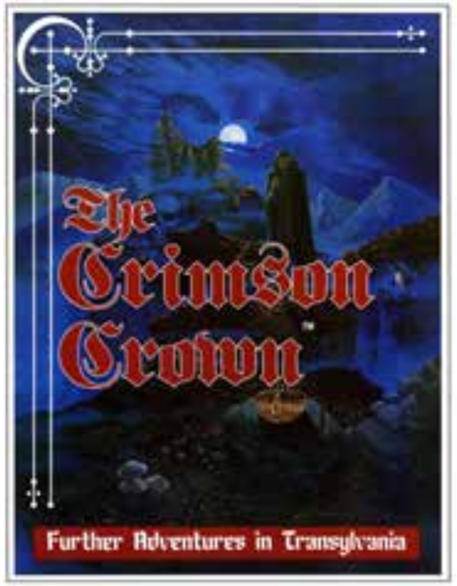
Unreleased Program & Source

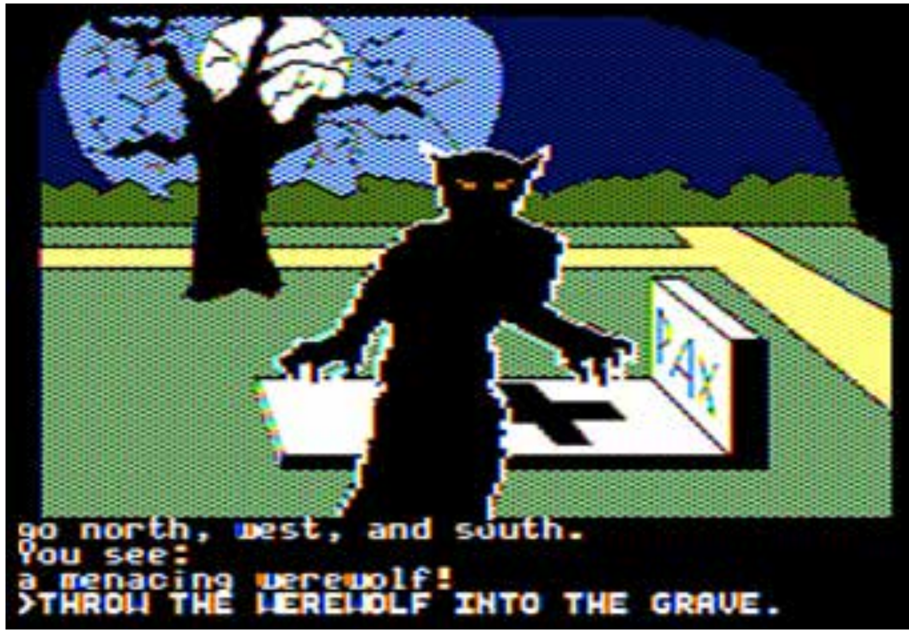
Double Hi-Res & ProDOS

Available in 2022 under a
Creative Commons CC BY-SA license

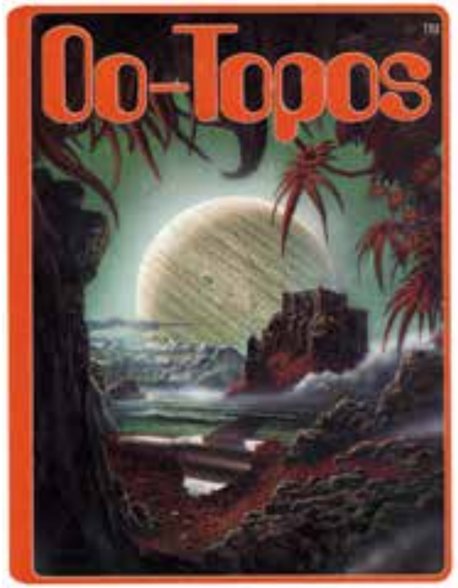
Comprehend 2: Adventure Development System





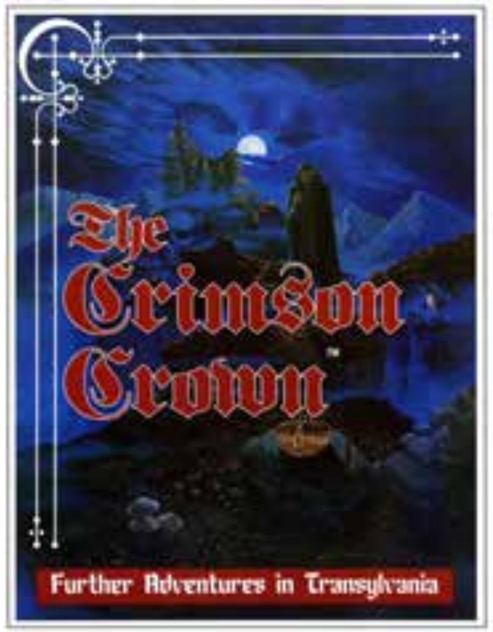


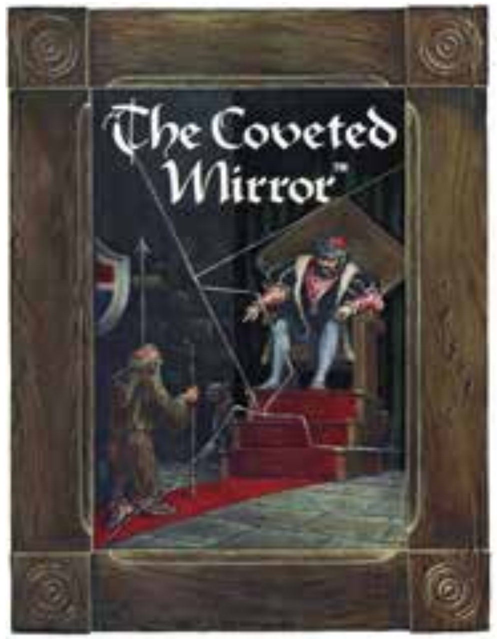
```
go north, west, and south.
You see:
a menacing werewolf!
>THROW THE WEREWOLF INTO THE GRAVE.
```

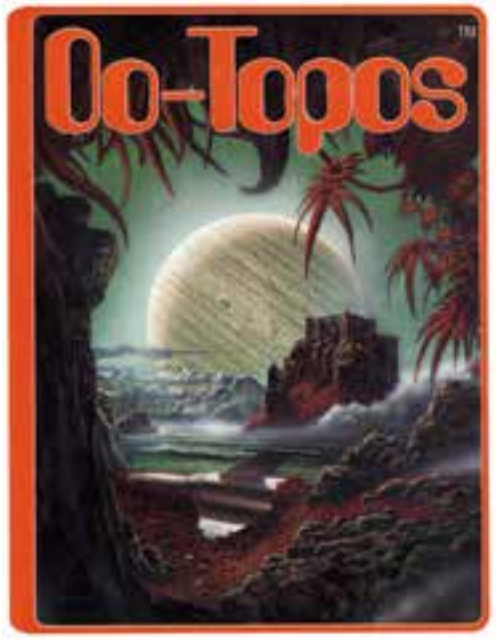


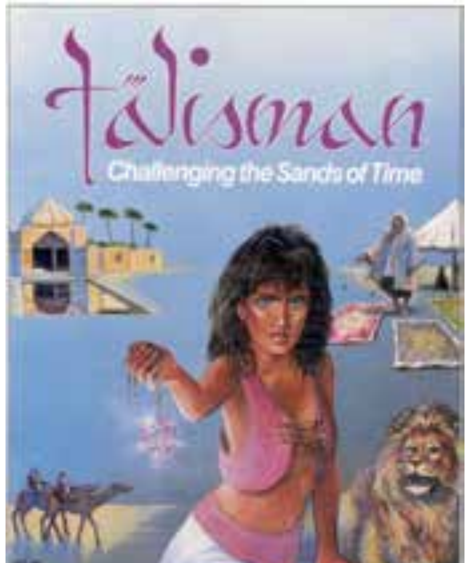
Available in 2022 under a
Creative Commons CC BY-SA license

5 Games: *Source Code & Resources + Surprises*





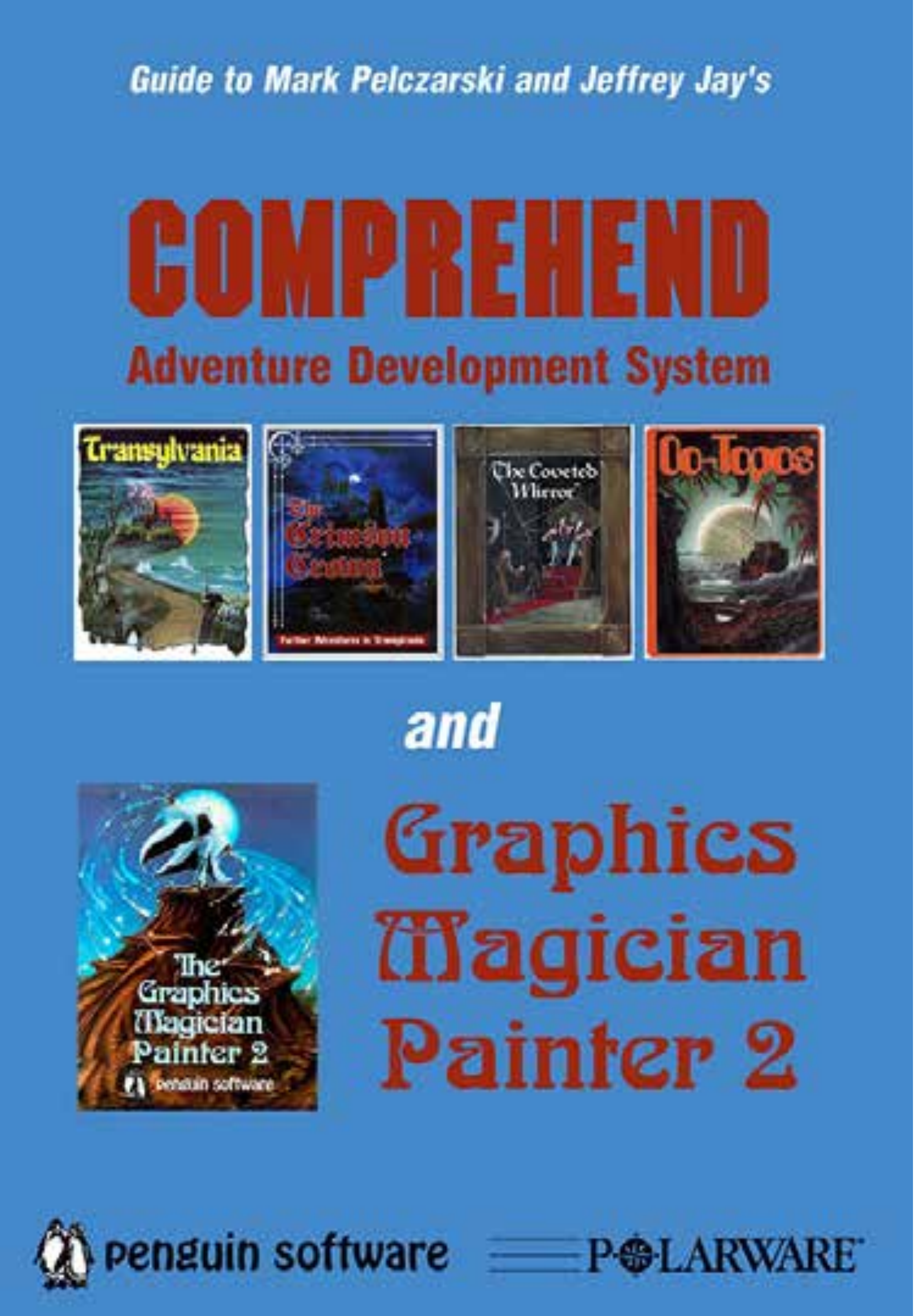




Available in 2022
for Educational and Historical
purposes – *no License*

CallAPPLE.org/penguin

Books in Progress



Guide to Mark Pelczarski and Jeffrey Jay's

COMPREHEND

Adventure Development System

and

Graphics Magician Painter 2

penguin software POLARWARE

Printed Manual

All available documentation for

Comprehend 2 and
Graphics Magician Painter 2

Available in 2022

Call–A.P.P.L.E.™

In Depth

All About Pilot

Programming Language for Computer-Aided Instruction

Produced by
Brian Wiser & Bill Martens

Net-works II BBS

by Nick Naimo

NET-WORKS II V2.52 SER# 44348
COPYRIGHT (C) 1982 BY A.O.S.
DISTRIBUTED BY H.T.S.P.I.
High Technology Software Products, Inc.
Oklahoma City OK / All rights reserved worldwide
High Technology Software

Available in 2022

Understanding the Apple IIe

by Jim Sather

Foreword by Steve Wozniak

ENHANCED EDITION

A Learning Guide and Hardware Manual
for the Apple IIe Personal Computer.

Understanding the Apple IIe

A Learning Guide and Hardware Manual for the Apple IIe Computer

*"**Understanding the Apple IIe** leaves no stone unturned in the search into the inner workings of the Apple IIe computer."*
—Steve Wozniak

Quality Software is pleased to present the definitive source of information about how the Apple IIe works. Jim Sather has followed up his exhaustive analysis of the inner workings of the Apple II computer with an even more detailed analysis of the Apple IIe. Now he has documented his findings in a way that will benefit everyone interested in microcomputer technology.

Understanding the Apple IIe Enhanced Edition —

- This new "Enhanced Edition" is completely remastered with updated text, layout and over 100 recreated illustrations.
- Documents all Apple IIe mainboard circuits, including diagrams and descriptions of the inner workings of the MMU, IOU, and timing HAL.
- Describes Apple II disk controller operation, including previously undocumented details of the logic state sequencer.
- Reveals previously unnoticed features of Apple IIe graphics. Explains Double High-Resolution graphics.
- Explains the differences between the Apple II and the Apple IIe Personal Computer.
- Contains 15 hardware and software Application Notes including EPROM mods, disk write protect mod, split screen programming, and the DOS HOSS firmware card modification.
- Includes a chapter on basic maintenance that provides simple troubleshooting steps.
- Describes the PAL IIe motherboard video circuits.
- Valuable programming reference material included.
- Documents the Enhanced IIe upgrade.

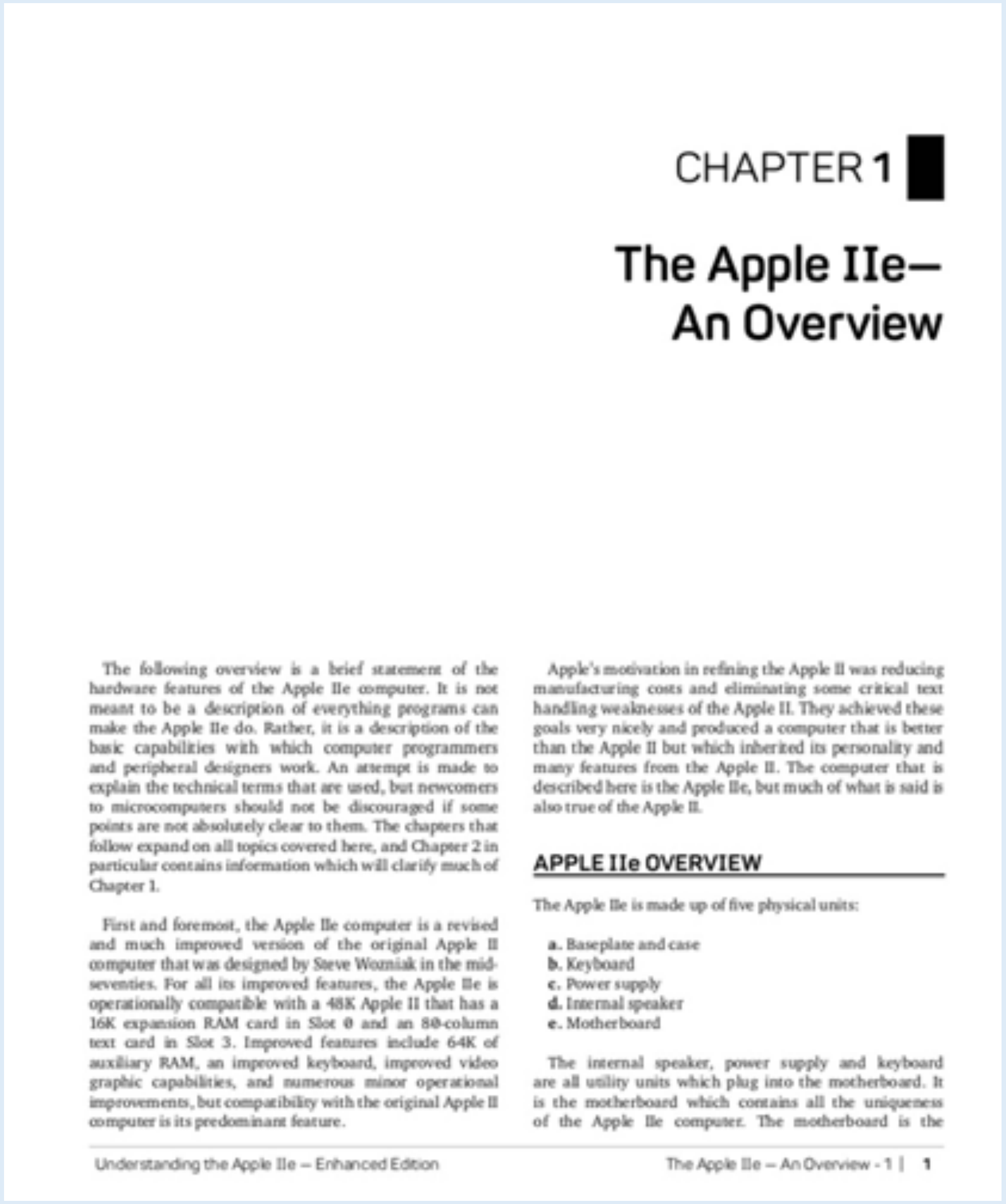
About the Author
James Fielding Sather, a former electronics field technical representative for ITT Gilfillian, was an independent author, programmer, and designer of circuits for microcomputers, specializing in the Apple II and Apple IIe.

If you are at all curious about how the Apple IIe works, you are sure to find ***Understanding the Apple IIe*** very valuable.

Its companion volume Understanding the Apple II, describes the operation of the early models, the Apple II and II plus.

callapple.org

Apple
PugetSound
Program
Library
Exchange



CHAPTER 1

The Apple IIe— An Overview

The following overview is a brief statement of the hardware features of the Apple IIe computer. It is not meant to be a description of everything programs can make the Apple IIe do. Rather, it is a description of the basic capabilities with which computer programmers and peripheral designers work. An attempt is made to explain the technical terms that are used, but newcomers to microcomputers should not be discouraged if some points are not absolutely clear to them. The chapters that follow expand on all topics covered here, and Chapter 2 in particular contains information which will clarify much of Chapter 1.

First and foremost, the Apple IIe computer is a revised and much improved version of the original Apple II computer that was designed by Steve Wozniak in the mid-seventies. For all its improved features, the Apple IIe is operationally compatible with a 48K Apple II that has a 16K expansion RAM card in Slot 0 and an 80-column text card in Slot 3. Improved features include 64K of auxiliary RAM, an improved keyboard, improved video graphic capabilities, and numerous minor operational improvements, but compatibility with the original Apple II computer is its predominant feature.

Apple's motivation in refining the Apple II was reducing manufacturing costs and eliminating some critical text handling weaknesses of the Apple II. They achieved these goals very nicely and produced a computer that is better than the Apple II but which inherited its personality and many features from the Apple II. The computer that is described here is the Apple IIe, but much of what is said is also true of the Apple II.

APPLE IIe OVERVIEW

The Apple IIe is made up of five physical units:

a. Baseplate and case
b. Keyboard
c. Power supply
d. Internal speaker
e. Motherboard

The internal speaker, power supply and keyboard are all utility units which plug into the motherboard. It is the motherboard which contains all the uniqueness of the Apple IIe computer. The motherboard is the

Understanding the Apple IIe – Enhanced Edition

The Apple IIe – An Overview - 1 | 1

- Produced and Redesigned by Gary Graham
- Co-produced by Brian & Bill
- Fully re-typeset, 20 of 100+ Illustrations recreated
- Publishing in 2022

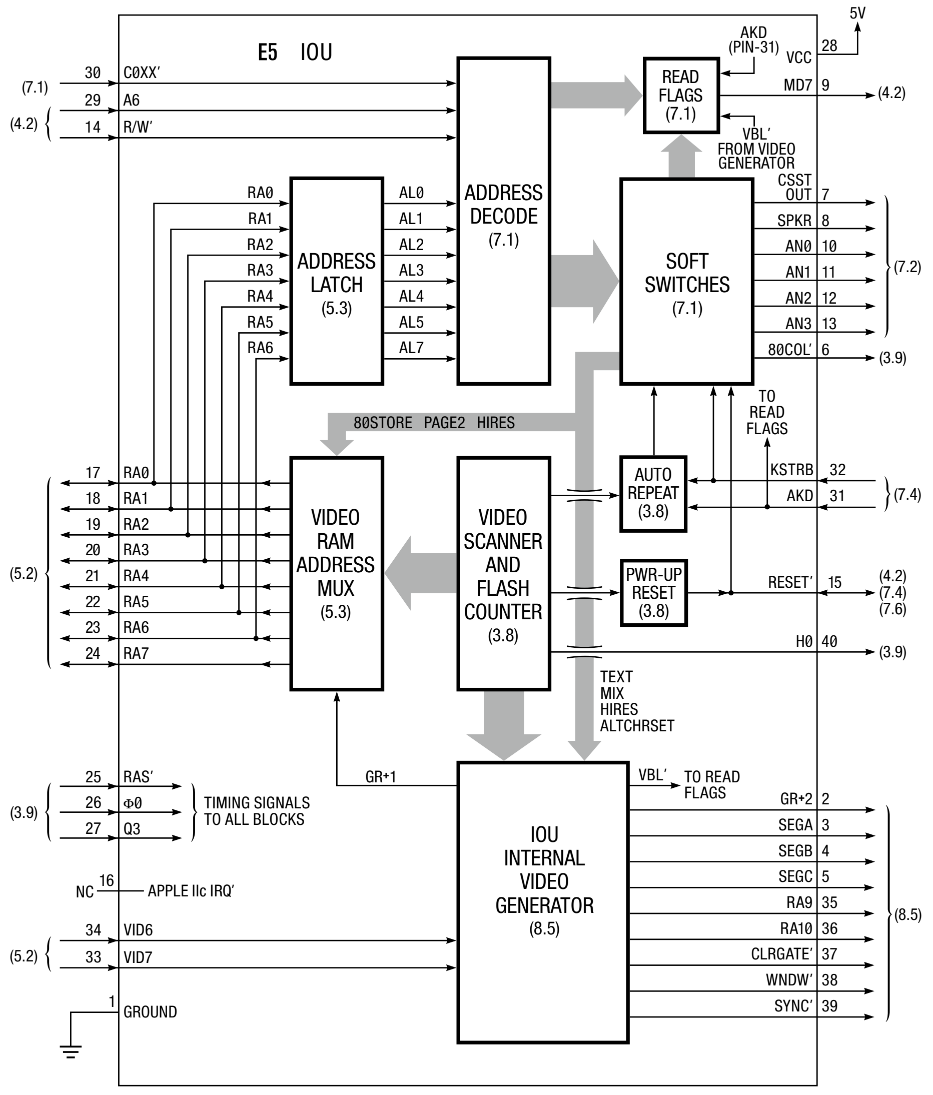


Figure 1.1 IOU Functions and Pin Assignments.

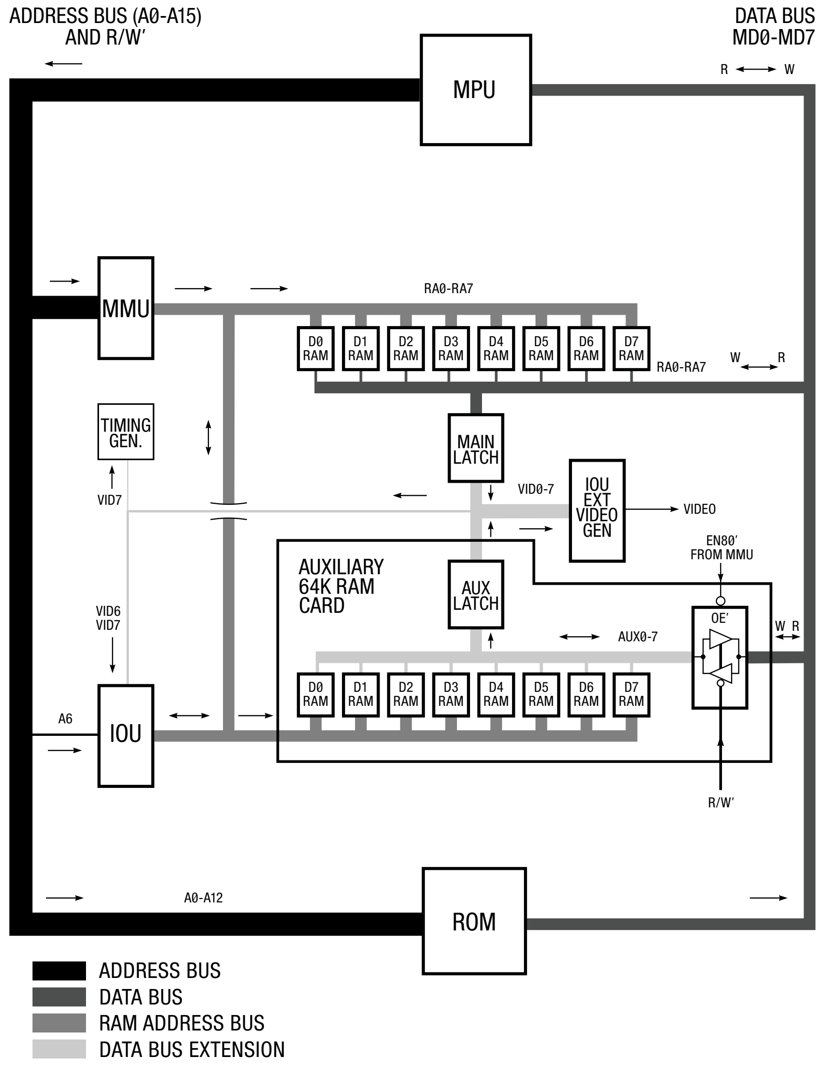
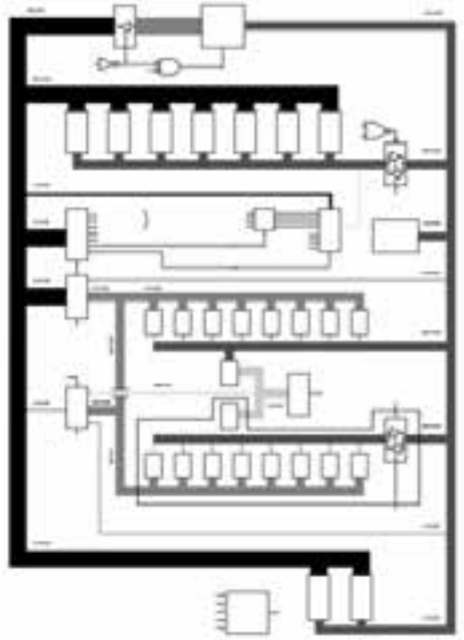


Figure 2.4 Bus Diagram: RAM Addressing and DATA Distribution in the Apple IIe.



CHAPTER **2**

The Bus Structure of the Apple IIe

There are many signals distributed throughout the Apple IIe, but the most fundamental data transfer takes place on the **data bus**, and the most basic control information is distributed via the **address bus**. To understand how the Apple IIe and other microcomputers really work, it is very important to understand the bus structure. Fortunately, it's not that hard to understand. The basic concepts of the bus structure are within the grasp of nearly everyone who uses a microcomputer.

The bus structure is a natural starting point for learning what really goes on inside the Apple computer. Discussing the bus structure will lead naturally to the discussion of the other microcomputer elements that the bus is connected to. First, though, we need to find out what a bus is and how it is used.

COMPUTER BUSES AND THREE STATE LOGIC

Logic signals in the Apple are distributed electrically via conductive paths on the motherboard. When a number of signals are grouped functionally and distributed throughout a microcomputer, they are collectively referred to as a bus. Physically, then, a bus is an electrical distribution of multiline information. In the Apple, the **address bus** is a **sixteen**-line electrically distributed information group, and the **data bus** is an **eight**-line electrically distributed information group.

Some devices connected to a bus are strictly receivers of information. ROM is like this in its connection to the **address bus**. Receivers respond to the high/low information on the lines of the bus without appreciably affecting the bus information. Electrically speaking, the receiver input presents a high impedance to the bus which enables other devices to bring the bus lines high or low. If impedance is a new word to you, it may help to think of high impedance as high isolation.

Some devices on a bus must be information transmitters capable of bringing the bus lines high or low. If more than one information transmitter is connected to a bus, each transmitter must be able to disconnect itself from control of the bus by presenting a high impedance to the bus. Only one device can control the bus at a time. Instead of two state, the outputs of these devices are said to be three state

or tri-state. The three states are high voltage, low voltage, and high impedance. All information transmitters to the data bus of the Apple are capable of presenting these states to their bus connections. The ROM output to the **data bus** is a typical three state output.

A third type of device, capable of transmitting to or receiving from a bus, is called a **transceiver** (**transmitter/receiver**). The MPU, for instance, receives (reads) data from and transmits (writes) data to the data bus, so as far as the data bus is concerned, the MPU is a transceiver. While the MPU is reading, it presents a high impedance to the data bus so the addressed device can place data on the data bus. While the MPU is writing, it controls the data bus.

Figure 2.1 shows a hypothetical 4-line bus. The symbols shown are schematic representations of a tri-state line driver, a line receiver, and a line transceiver. A triangle represents a single line driver. Triangles with a control line coming in from the side are tri-state line drivers. A little circle at a control input to a triangle means that the input is active when its voltage is low.

Here is a truth table for the tri-state line driver shown in Figure 2.1:

INPUT	OUTPUT ENABLE	OUTPUT
Any	Low	High Impedance
High	High	High
Low	High	Low

The control line either enables the high/low output or forces the output to high impedance. The high/low output, when enabled, follows the input.

It can be seen that the **output enable** controls of the various information transmitters are the key to cohesive control of the bus. For a bus with many possible information transmitters, like the data bus of the Apple, there has to be some intelligent management of the various tri-state output enables. We will see shortly how this is accomplished. In the following discussions, remember that when a device like a ROM chip responds to an address prompt by placing data on the data bus, this is accomplished via an output enable to the tri-state outputs of the ROM chip.

Figure 2.2 shows a highly simplified diagram of the bus structure of the Apple IIe. There are two distinct multiline signal paths: the **address bus** and the **data bus**. The R/W line (Read/Write control) is shown separate and can be thought of as an extension of the address bus controlling the direction of data flow on the data bus. Communication takes place on every 6502 cycle between the MPU and an addressed device. Data flows between the MPU and the device in a direction determined by the R/W line. The MPU controls the R/W line and the address bus.

Figure 2.3 shows the two types of bus access which occur in the Apple IIe. In a read access, the MPU places an address on the address bus and reads the data bus. In a write access, the MPU places an address on the address bus and places data on the data bus. This establishes a system of data bus control that had to be implemented in the design of the Apple.

The control system works like this:

1. When the R/W line is low (write access), all inputs to the data bus are disabled except the MPU.
2. When the R/W line is high (read access), all inputs to the data bus are disabled except the device which is addressed.

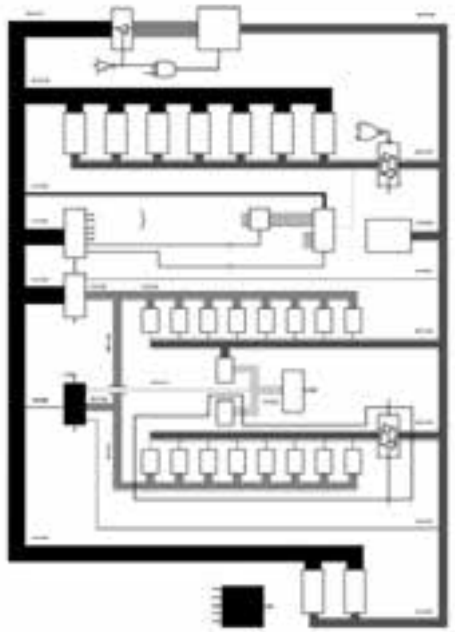
This system concept keeps traffic flow orderly and is a basic feature of microcomputer design.

The only remaining points to be made about buses involve semantics. The **peripheral slots** are sometimes referred to as the **peripheral bus** or the **Apple bus**. In fact, the wiring of the slots fits our description of a bus as a functional group of distributed signals. The slots are a bus whose distributed signals include the address bus, the data bus, and other signals. Up to this point, the discussions have avoided calling the slots a bus only to avoid confusion between the card cage bus and the more basic address bus and data bus. The connections to the RAM and ROM chips form two more distributed signal groups that can be referred to accurately as the RAM bus and the ROM bus. This book will continue to use the word "bus" to refer to the address bus, the data bus, and the extensions of these two basic communications paths. The peripheral bus, RAM bus, ROM bus, and other distributed signals will be referred to using other terminology.

The lines of the various buses in the Apple are referred to by one or more letters followed by a number. For example, the lines of the Apple address bus are referred to as *A0* through A15. The largest number, A15 in this example, refers to the line which carries the most significant bit of information.

A list of bus terminology used in this book follows here:

NAME OF BUS	LINE TERMINOLOGY
Address	A0—A15
Data	MD0—MD71
Multiplexed RAM address	RA0—RA7
Auxiliary RAM data	AUXD0—AUXD7
Video data	VID0—VID7
Peripheral slot data	D0—D7



CHAPTER 3

Timing Generation and the Video Scanner

Most operational aspects of the Apple IIe have now been discussed within the context of the bus structure. However, this discussion has left out one of the Apple's most important operational aspects–timing. Timing synchronizes everything that goes on in the Apple. To discuss it, we must get into real nuts and bolts detail about computer operation.

Up to this point, the subject matter of *Understanding the Apple IIe: Enhanced Edition* has been of a general nature. No attempt was made in Chapters 1 and 2 to explain the finer points of Apple IIe operation. Having gained understanding of the Apple's bus structure, you are largely aware of the methods of communication and control that take place in this computer. The following chapters will build on this foundation of understanding, examining and discussing the detailed features of all functional areas of the Apple IIe.

The perceptive reader is probably getting the message that the going is about to become stickier. This book attempts to explain as much as possible about the operation of the Apple in understandable English. There comes a point, however, beyond which clear illustration is achieved only with such technical tools as timing diagrams, truth tables, logic diagrams and schematic diagrams. One of the goals of *Understanding the Apple IIe: Enhanced Edition* is to assist those readers who desire to do so to analyze the operation of the Apple IIe in depth. For this reason, some technically oriented analysis aids are presented in this chapter and succeeding chapters. These technical aids will be accompanied by technical language. Every person reading these words is capable of understanding the technical sections, but some readers may not wish to, and others will find it a struggle. Every effort has been made to assist all readers in achieving fullest possible understanding from the least possible effort.

By way of warning, the details of some functional areas are just plain difficult, but most of the areas are pretty painless.[1] In particular, much of the complexity of the Apple is concentrated in RAM and its associated circuitry. Some other complicated circuitry, like the internal workings of the MPU, will not be discussed at all. Besides the RAM

1. Even though it is not part of the motherboard circuitry, disk I/O is the subject of a chapter of Understanding the Apple IIe: Enhanced Edition. Readers intrepid enough to tackle this chapter will find disk I/O to be a complex but interesting area of study.

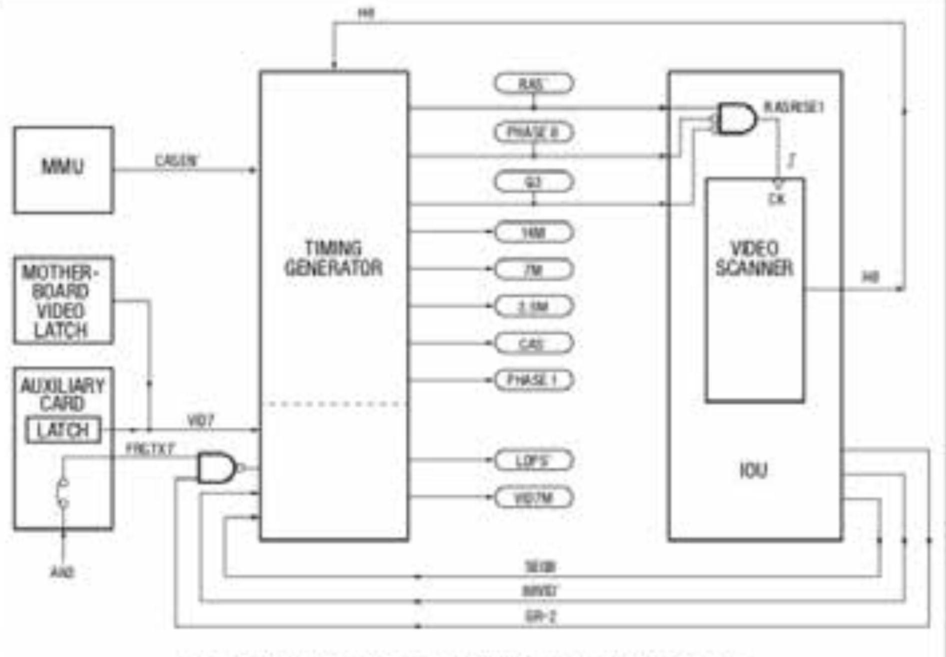


Figure 3.1 Functional Diagram: The Timing Generator and Video Scanner

circuitry, the most difficult topics probably are the details of timing and video generation. Timing comes next, so put on your overshoes - we're going wading.

TIMING OVERVIEW

The important timing signals in the Apple IIe all originate at a small group of circuits called the timing generator. You should appreciate this when studying the Apple, because it makes a difficult job easier. Interrelated digital timing originating from multiple sources can scramble your brains. With a single timing source we can assimilate the timing sequences and then apply them to the various functional areas in the following chapters.

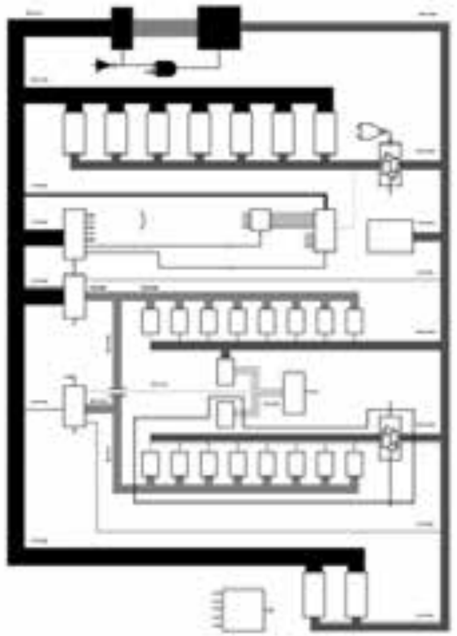
Timing signals are distributed to all areas of the Apple, but the Apple's timing requirements are determined primarily by RAM usage. RAM is accessed alternately by the 6502 processor and the video scanner. Executing a stored sequential program and generating a color television video signal are two entirely different tasks, but the two tasks are synchronized in the Apple. As we shall see, execution of this double task dictates certain facts of life about Apple timing.

The timing generator controls the timing and affects all areas of the Apple IIe. Some of these are as also affect timing generation (see Figure 3.1).

The external influences are as follows:

1. One of the timing signals, CAS', is enabled or disabled by CASEN' from the MMU.
2. VID7 of the video data bus and the display mode affect the generation of the LDPS' and VID7M video timing signals.
3. An auxiliary card working in coordination with a Slot 1 peripheral card can disable all of the timing signals and substitute alternate signals. This is not normally done in operational Apples, but it is a capability.
4. Feedback from the video scanner elongates one system clock period toward the end of each horizontal television scan.

The elongation referred to in item 4 above is necessary to keep colors consistent from scan to scan. It also means the clock period of the 6502 is not constant but is elongated on every 65th cycle. This book will refer to this elongated machine cycle as the long cycle. Because of the feedback from the video scanner to the timing generator, the two



CHAPTER 4 The 6502 Microprocessor

The 6502 MPU (Micro Processing Unit) is the device in the Apple IIe which executes stored sequential programs. It is a single 40-pin integrated circuit that executes 6502 machine language programs as it reads them from the data bus. It can be thought of as the brains of the Apple.

The 6502 was designed by MOS Technology in the mid 1970s as part of their MCS6500 series microprocessor family. It has been a popular choice as a microprocessor for personal computers, being used in computers produced by Apple, Atari, Commodore, Ohio Scientific, Rockwell International, and other manufacturers. The 6502 gives adequate computing speed and versatility at a very low cost. Its programming language is very simple, making it an ideal MPU for the occasional computer programmer.

The most important 6502 related knowledge for an Apple owner to attain is programming knowledge. The ability to read and write 6502 assembly language programs greatly expands the horizons of an Apple computerist. 6502 assembly language is not, however, a major topic of this book. These pages are concerned primarily with the hardware implementation of the 6502 in the Apple IIe computer. Volumes have been written about various aspects of the 6502, especially programming. The choice of 6502 topics in this chapter was governed by the unique features of 6502 use in the Apple, and by the goal of this book to fill information gaps in Apple literature available to the public.

The Western Design Center, Inc., founded by Bill Mensch of Motorola and MOS fame, sells modern versions of the 6502 in both the classic 40 pin DIP package (appears to be pin-compatible with the original), as well as more modern surface mount packages. These are being used in all sorts of commercial devices, including external sound interfaces from companies like MOTU, medical devices and scanners, hand-held games and more. Bill Mensch is one of the holders of the 6502 patent, so who better to work on it today than him. The W65C02S is closest to the original 8 bit 6502. There are also additional versions that are 8/16 bit hybrids and can therefore address more memory, and should be compatible with the Nintendo and Apple IIGS versions of the chip. For pricing and availability you can visit:

www.westerndesigncenter.com/wdc/

6502 SIGNALS

There are 40 pins on a 6502, three of which serve no function. In addition to the address output and data input/output, there are four outputs (R/W, PHASE 1, PHASE 2, and SYNC) and six inputs (READY, IRQ', NMI', PHASE 0, SET OVERFLOW', and RESET'). There are three power supply connections. One pin requires +5 volts and two pins require ground. Figure 4.1 shows the 6502 pin assignments. Figure 4.2 shows the 6502 hardware implementation in the Apple. A brief discussion of the 6502 signals with Apple implementation notes follows.

Clockpulses - PHASE Ø, PHASE 1, PHASE 2

The 6502 has most of its required clockpulse generation circuitry built-in. It requires only an externally generated time base which can be implemented in several ways. In the Apple, the PHASE 0 time base is developed independent of 6502 internal circuits and fed as the clockpulse input to the 6502.

The 6502 generates its required PHASE 1 and PHASE 2 clocks from the PHASE 0 input. PHASE 1 is high during the first half of a machine cycle, and PHASE 2 is high during the second half of a machine cycle. PHASE 1 is not the simple inversion of PHASE 2. There is a slight delay between the PHASE 1 transitions and the PHASE 2 transitions. The rising edge of one always follows the falling edge of the other. The PHASE 1 and PHASE 2 clocks are available at pins 3 and 39 of the 6502. PHASE 1 and PHASE 2 are not connected in the Apple IIe but are used only inside the 6502.

Address and R/W'

During every machine cycle, the 6502 places an address on its address output. In association with the address it outputs, it brings its R/W line high or low, thereby telling the world whether it wants to read or write data. With 16 address lines, the 6502 is capable of producing 65536 different values at its address output.

The 6502 address and R/W outputs are not tristate in the Apple, but these signals are connected to the address bus through external tri-state bus drivers. This enables peripheral cards to gain access to the address bus via the DMA' line.

Data Bus

The data input/output of the 6502 is eight lines. This gives the 6502 its overall classification as an 8-bit microprocessor. Data direction is inward except during PHASE 2 of write cycles. In the Apple IIe, the 6502 data lines are connected directly to the data bus.

RESET'

The RESET' input to the 6502 causes the 6502 to start or restart. A RESET' causes the 6502 to disable interrupts and begin program execution at an address stored in locations $FFFC and $FFFD of ROM. 6502 operation is inhibited while RESET' is held low. The RESET' sequence begins when RESET' transits from low to high.

In the Apple IIe, the RESET' line is connected to pin 31 of

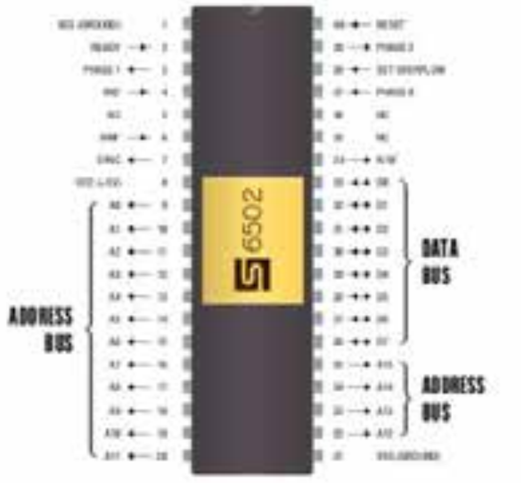
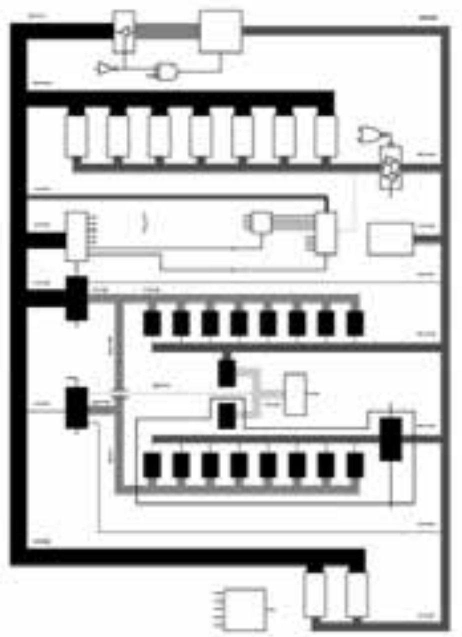


Figure 4.1 6502 Pin Assignments

```
SOURCE FILE: SPLIT SCREEN
0000:          1 ***********************************************************
0000:          2 *                                                         *
0000:          3 *                     HIRES/LORES SPLIT                   *
0000:          4 *                        8515/8515                        *
0000:          5 *                      BY JIM SATHER                      *
0000:          6 *                        2/15/1983                        *
0000:          7 *                                                         *
0000:          8 ***********************************************************
C000:          9 KBD     EQU  $C000
C010:         10 KBDSTRB EQU  $C010
C054:         11 PAGE1   EQU  $C054
C056:         12 LORES   EQU  $C056
0000:         13 *
0000:         14 * THIS PROGRAM TOGGLES THE HIRES/LORES SWITCH
0000:         15 * EVERY 8515 CYCLES.
0000:         16 *
----- NEXT OBJECT FILE NAME IS SPLIT SCREEN.OBJ0
1F00:         17         ORG  $1F00
1F00:AC 54 C0 18 SPLIT   LDY  PAGE1
1F03:A0 27    19 SLEW    LDY  #39       ;SLEW SCREEN IF KEY PRESSED.
1F05:20 27 1F 20         JSR  WAITX10
1F08:AC 10 C0 21         LDY  KBDSTRB
1F0B:AC 00 C0 22 KEYCHK  LDY  KBD
1F0E:30 F3    23         BMI  SLEW
1F10:69 01    24         ADC  #1        ;TOGGLE HIRES/LORES SWITCH
1F12:29 01    25         AND  #$01
1F14:AA       26         TAX
1F15:BC 56 C0 27         LDY  LORES,X
1F18:A2 08    28         LDX  #8
1F1A:20 31 1F 29         JSR  WAITX1K   ;WAIT 8000 CYCLES
1F1D:A0 31    30         LDY  #49
1F1F:20 27 1F 31         JSR  WAITX10   ;WAIT 490 CYCLES
1F22:18       32         CLC
1F23:90 E6    33         BCC  KEYCHK
1F25:         34 *
1F25:         35 * TIMING ROUTINES:
1F25:         36 * WAITX10 WAITS Y-REG TIMES 10 CYCLES.
1F25:         37 * (MINIMUM WAIT 20 CYCLES)
1F25:         38 * WAITX1K WAITS X-REG TIMES 1000 CYCLES.
1F25:         39 *
1F25:D0 01    40 LOOP10  BNE  SKIP
1F27:88       41 WAITX10 DEY                ;WAIT Y-REG TIMES 10
1F28:88       42 SKIP    DEY
1F29:EA       43         NOP
1F2A:D0 F9    44         BNE  LOOP10
1F2C:60       45         RTS
1F2D:48       46 LOOP1K  PHA
1F2E:68       47         PLA
1F2F:EA       48         NOP
1F30:EA       49         NOP
1F31:A0 62    50 WAITX1K LDY  #98       ;WAIT X-REG TIMES 1000
1F33:20 27 1F 51         JSR  WAITX10
```

Figure 3.11a Assembler Listing: Timed Execution Screen Splitting (1 of 2)



CHAPTER 5 RAM and Memory Management

One might think that RAM and its associated circuitry should be a relatively easy subject. You write to it and read from it. What else is there?

Well, the **MPU** does write to and read from RAM, but the video scanner reads from RAM too. Additionally, both the MPU and **video scanner** access **auxiliary card RAM**.[1] And then there is the 64K dynamic RAM chip with its ROW address, COLUMN address, and refresh requirement. Add all of this to the most involved bank switching scheme ever imagined and you have a lot of functional and operational complexity.

When RAM is accessed by the MPU, motherboard circuitry must activate signals which tell the RAM chips to pass data to or receive data from the data bus. Control of Apple IIe data bus communication is the task of the MMU. An MPU program configures the Apple memory by setting up MMU softswitches, and when the MPU accesses an Apple IIe device, an MMU data bus management signal either directly or indirectly activates the device. Apple refers to this broad control function as **memory management**, and as will be seen, managing RAM in the Apple IIe is the major part of the task of managing memory.

In this chapter, we will examine the requirements of the 64K dynamic RAM chip, and the ways in which they are met in the Apple IIe. We will also see how the MMU monitors the address bus to manage the overall configuration of the Apple IIe memory map.

THE 64K DYNAMIC RAM CHIP

64K RAM chips are 65,536 **bit** read/write memories. As is indicated in the bus structure diagram in Figure 2.7, it takes eight chips to make up the 65,536 bytes of read/write memory on the Apple IIe motherboard. This Standard chip is available from a number of manufacturers in a variety of speeds. With a 2 MHz access rate, the Apple does not put a particularly stringent speed requirement on its RAM.

The RAM chip is a 16-pin device requiring two power supply inputs, +5V and ground. Figure 5.2 shows the pin assignments of the 64K RAM chips (see RAM chip F6).

1. Most discussions of RAM in Understanding the Apple IIe: Enhanced Edition assume that a 64K RAM card is installed in the auxiliary slot.

CHAPTER **6**

ROM in the Apple IIe

Non-erasable, random access, read only memories have taken many forms in the history of digital computers. From vacuum tubes to diodes to small scale integration to large scale integration, there has always been a need for the general purpose computer to have a resident program ready to tell it what to do at turn on. With the coming of large read only memories on single chips, the scope of programs contained in ROM in general purpose computers has expanded greatly.

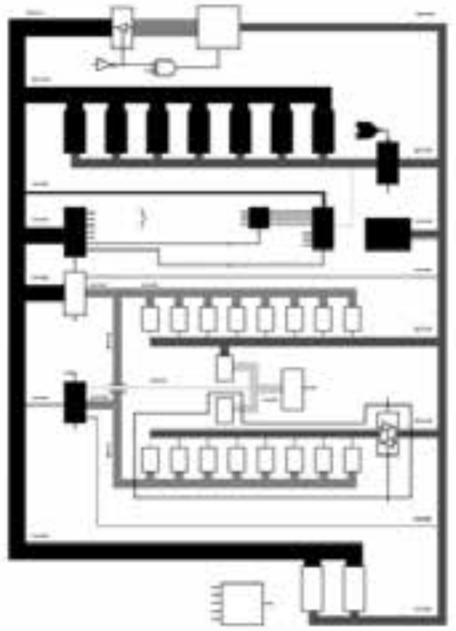
The Apple IIe design supports 16,128 bytes of motherboard ROM, addressed from $C100 through $FFFF. Additionally, several provisions exist for controlling the Apple via programs in ROM on peripheral cards. These include addressing peripheral ROM using a slot's assigned address area (its $C0nX DEVICE SELECT' range and its $CnXX I/O SELECT' range), addressing peripheral ROM using the I/O STROBE' ROM addresses ($C800—$CFFF), and inhibiting motherboard memory and stealing addresses $0000—$BFFF and $C100—$FFFF. All told, the Apple IIe computer has a very versatile capability for operating under control of programs stored in ROM.

The sophistication of the ROM chips makes their connections in the Apple simple and easy to understand. Nevertheless, the topic of ROM is involved enough to merit its own chapter. For the most part, discussion of ROM in this chapter is limited to the motherboard C1—DF and E0—FF ROMs, and peripheral slot ROMs containing 6502 machine code and related data. The keyboard ROM and video ROM are covered in Chapters 7 and 8 respectively.

ROM HARDWARE

The Apple IIe firmware is programmed into two 2365A type ROMs. The 2365A is a 28-pin, 8192-byte, NMOS ROM with two programmable chip select inputs, an OE' input, an CE' input, and 200-, 300-, or 450-nanosecond access time. NMOS stands for Negative channel, Metal Oxide Semiconductor construction. These technical terms will help you if you look for the equivalent chip in a manufacturer's data book. Look under NMOS, 8192 x 8 ROMs.

A number of manufacturered 28-pin, 8192-byte ROMs will work in the Apple IIe. 2365A is the Synertek part number, and it is used here because it is used in Apple



CHAPTER **7**

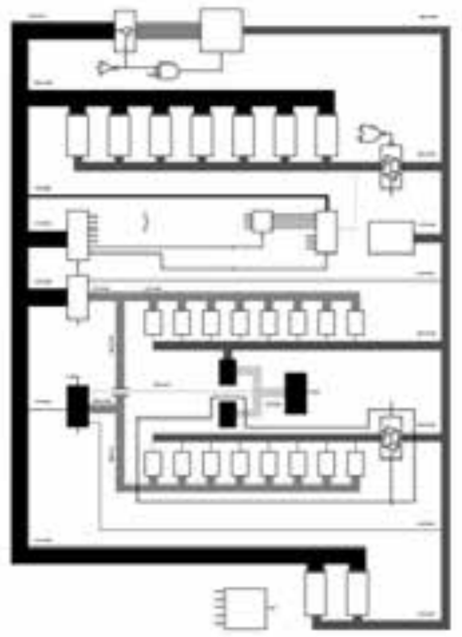
Input/Output in the Apple IIe

It bears repeating. MPU command of all devices in the Apple IIe computer is via signals decoded from the address bus. All persons who program the Apple become aware of this sooner or later, and all users of the Apple can save themselves problems if they understand command by addressing. The concept of data transfer between the MPU and a memory location is very easily grasped, but it must be understood that the MPU also controls parts of the computer via the address bus, often with no related transfer of data on the data bus.

For the most part, I/O in the Apple is performed under direct control of the MPU. This includes all built-in I/O features except video output, and it also includes most conventional I/O performed via peripheral cards. The MPU controls these devices by addressing them, just as if it were addressing memory. As every memory location has a specific address, every I/O device which is directly controlled by the MPU has a specific address or range of addresses.

Now there is no "Control Address Bus" command in the 6502's repertoire. The 6502 reads from or writes to the data bus on every cycle. So what does the programmer do when he wants to toggle the speaker? He utilizes a "LDA $C030" or a "CPX $C030" or a "WHO GIVES A DARN $C030" and ignores the meaningless data bus. This is why you can program the speaker with a statement like "SOUND = PEEK(-16336)". The object is not to "PEEK" into memory. The object is to get $C030 onto the address bus, commanding the speaker to toggle. Beneath the lid of the Apple, on every cycle, whether memory or a control function is being addressed, the state of the address bus is decoded to tell the rest of the Apple what the MPU is doing.

The purpose of this chapter is to discuss the various I/O devices that are built into the Apple IIe, and to show how addresses are decoded to give the MPU control over I/O processes. Related subjects such as peripheral slot capabilities and I/O firmware are discussed. Video output generation is not discussed, but is the sole topic of the following chapter.



CHAPTER **8**

Video Generation

The marriage of data processing and video display technology has been one of the most important developments in the advance of computers. Combined with the keyboard, the video display provides a direct communication link between people and computers that makes the old, expensive, physically large computers seem to be just machines. Imagine communication with your Apple using a teletype terminal with no video display. How would that affect important applications like word processing, spread sheet accounting, data base management, graphics display, and Donkey Kong? Without video, the Apple wouldn't be worth owning.

Of course, the Apple does have a video display capability, and a large portion of the motherboard hardware is related to the generation of video. We have seen in other chapters that many features of bus structure, timing, and RAM addressing in the Apple IIe are dictated by the fact that the dissimilar tasks of stored program execution and video display generation are performed simultaneously in this computer. Additionally, the **video scanner** (internal to the IOU) and **video generator** (partly internal and partly external to the IOU) are functional areas that exist for the sole purpose of making up the video display. All of these functional areas are interconnected in a scheme which allows Apple programs to control the video output.

Figure 8.1 is a simplified diagram of the video display control processes of the Apple IIe. As Figure 8.1 shows, the MPU controls the output of video in a very indirect way. Under direction of the controlling program, the MPU sets the screen mode, computes a correct address in memory, and stores selected code at that memory address. In so doing, the MPU is setting up a small area of the **screen map**. The extent of MPU involvement and, by extension, programmer involvement in outputting video consists entirely of setting up this screen map. The actual output of video is controlled by the video scanner which scans memory and drives out the map, and by the video generator which processes the map to produce the **VIDEO signal**. You can actually stop the MPU by pulling READY or DMA' low, and the Apple will continue to output to the screen the map which was set up by the MPU before you stopped it.

Compare this indirect MPU involvement to a printer output port where the MPU-under program control as always—actually stores coded data at a special address

CHAPTER **9**

The Disk Controller

The long term success of the Apple II line of computers could not have come about without the development of the Disk II 5¼ inch floppy disk drive. For simple storage of data and programs, disk I/O is merely faster and more convenient than cassette I/O. But for important computer uses such as word processing, data base management, business accounting, and file handling, the disk drive or its equivalent is mandatory. There can be no doubt that for most owners the disk drive is the most important peripheral in the Apple IIe computer. Since the original Apple II was strictly cassette based, the interface to the Disk II had to be built on a peripheral card . The extent of motherboard support of the Disk II is an empty card slot and the motherboard firmware. Motherboard firmware includes no disk handling routines but only looks for disk handling routines in the peripheral slots and jumps to them at power up. The **bootstrap** program and the circuits that interface the computer with the drive are on the **disk controller**, which is a peripheral card usually installed in Slot 6. The disk controller is connected by a 20-wire ribbon cable to the disk drive, which contains more electronic circuits as well as the drive mechanisms.

It is primarily the controller circuits and the available disk operating systems which determine the features of disk operation that are unique to the Apple, and it is the controller circuits which are the main topic of this chapter. General features of the disk drive are also discussed, but no attempt is made here to document the disk operating systems beyond the **RWTS** (Read or Write a Track and Sector) subroutine of DOS 3.3, the **DIIDD** (Disk II Device Driver) subroutine of ProDOS, and the disk data formats of DOS 3.2, DOS 3.3, and ProDOS.[1]

DISK II OVERVIEW

Near the end of 1977, Apple Computer's decision making group was still very small. Mike Markkula, the chairman of the board, presented the group with a list of products needed to be developed for the Apple. At the top of the list was a floppy disk drive for the Apple II. Within a very short period of time, Apple developed the Disk II and released it along with its operating system, DOS 3.

1 Operation of DOS 3.3 and ProDOS are well documented in the books *Beneath Apple DOS* (Quality Software, 1982) and *Beneath Apple ProDOS* (Quality Software, 1984) by Don Worth and Pieter Lechner.

CHAPTER 10

Maintenance and Care of your Apple IIe

The modern microcomputer is such a marvelous thing. Just think of the accumulated knowledge and industrial capability of the human race represented by such a machine. Invented by man, feared by man, exploited by man, and hated by man. Especially hated by man when it doesn't work right. After years of having computer systems subjecting us to impersonal and illogical errors, we have advanced to the point where we have computer systems in our own homes subjecting us to personal and illogical errors.

The vast majority of computer mistakes are caused by imperfect programs. The more involved a program, the greater the chance of an oversight by the programmer. We used to curse the computers. Now, when a husband writes a program that allows his wife to enter her kitchen recipes, then destroys them in milliseconds, it's not the computer that gets cursed. Yes, today's computers are very personal.

Occasionally, computer malfunctions will actually be caused by a hardware failure rather than a program in disarray. This should be a fairly rare occurrence, because digital electronic circuitry is so reliable. Yet, hardware failures do occur, and most of us encounter them eventually in our home or business system. In this chapter, attention will be given to the maintenance philosophy of the Apple IIe computer. There will be some discussion of what options you have when your system fails, and of some simple fault isolation steps which can be taken by you in your home. We also will discuss ways of reducing the probability of hardware failure in your system.

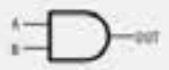
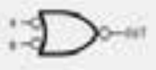
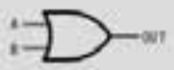
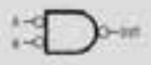
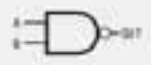
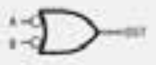
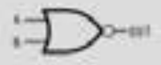
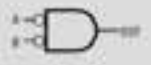
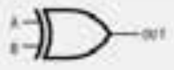
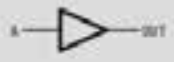
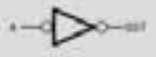
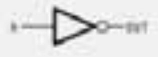
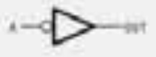
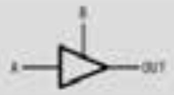
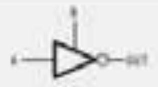
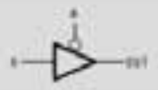
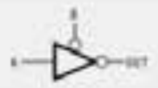
APPLE HARDWARE RELIABILITY

There is no electronic circuitry more reliable than modern digital electronic circuitry. Digital ICs routinely operate for thousands of hours without failure, and they are easy to replace if they do fail. The Apple II line of personal computers where built to be very reliable. There are, however, less reliable facets of a computer than the ICs that populate it. Some weak links in the reliability chain are discussed here.

Tinkering Users

If you are very involved with your hardware, trying new and different things all the time, you are bound to make some mistakes which cause hardware casualties. People

Table E.1 Basic Logic Gates (No relation to Bill Gates)

NAME	REPRESENTATION	SECONDARY REPRESENTATION	TRUTH TABLE			6502 EQUIVALENT
			B	A	OUT	
AND			L	L	L	AND XXXX
			L	H	L	
			H	L	L	
			H	H	H	
OR			L	L	L	ORA XXXX
			L	H	H	
			H	L	H	
			H	H	H	
NAND			L	L	H	AND XXXX
			L	H	H	EOR #$FF
			H	L	H	
			H	H	L	
NOR			L	L	H	ORA XXXX
			L	H	L	EOR #$FF
			H	L	L	
			H	H	L	
EXCLUSIVE OR			L	L	L	EOR XXXX
			L	H	H	
			H	L	H	
			H	H	L	
AMPLIFIER				L	L	NOP
				H	H	
INVERTER				L	H	EOR #$FF
				H	L	
TRI-STATE AMPLIFIER HIGH ENABLE			L	L	Z	—
			L	H	Z	
			H	L	L	
			H	H	H	
TRI-STATE INVERTER HIGH ENABLE			L	L	Z	—
			L	H	Z	
			H	L	H	
			H	H	L	
TRI-STATE AMPLIFIER LOW ENABLE			L	L	L	—
			L	H	H	
			H	L	Z	
			H	H	Z	
TRI-STATE INVERTER LOW ENABLE			L	L	H	—
			L	H	L	
			H	L	Z	
			H	H	Z	

A.P.P.L.E. KansasFest Giveaway

In honor of the return to in-person KansasFest, A.P.P.L.E. will award copies of an upcoming, unreleased A.P.P.L.E. book to three lucky winners.

Contest is open to all KansasFest attendees, both physical and remote. Details will be posted in our KansasFest Discord channel – “A.P.P.L.E.’s Blast from the Past 2022.”

Winners will be announced after KansasFest at:
www.callapple.org

Surprise in A.P.P.L.E. Catalog

A.P.P.L.E. Catalog
Summer 2022

KansasFest Edition

Produced by:
A.P.P.L.E. Staff

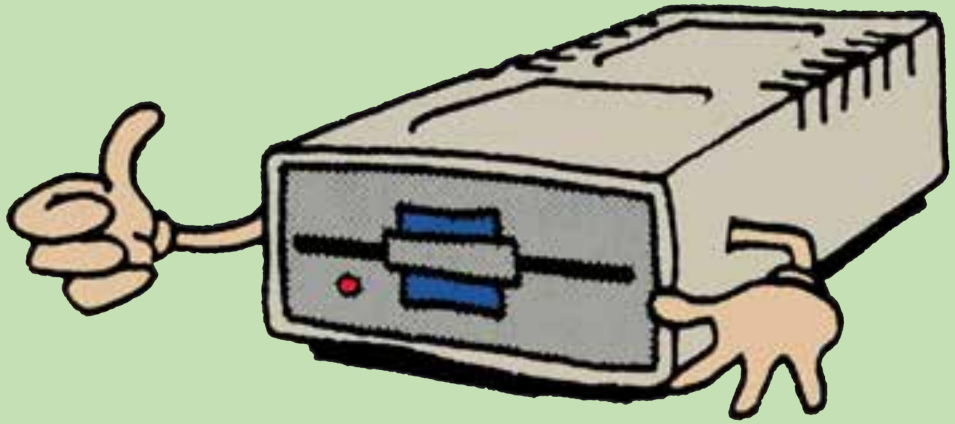
Apple PugetSound Program Library Exchange

Download our PDF Catalog on Discord or www.callapple.org

A new ‘mystery’ book may be of interest: www.callapple.org/books

[redacted]

Stay Tuned…



CallAPPLE.org