hardcore

UPDATE 1.1 SEP '81

update to DISKLOCKS

TEXT INVADERS

HARDCORE ADVENTURE

]9[ ]8[

x

!

1
=7=

HARDCORE computing

:::::::::::::::::TABLE OF CONTENTS:::::::::::::::::::

Publisher.....Charles R. Haight
Editor..... Bev R. Haight
Subscriptions.....Karen Fitzpatrick

Contributing Editors:
.....Bobby
.....B (Leo) Bryte
Graphic presentations.....Ryuji

HARDCORE Updates are published monthly except on those months that HARDCORE Computing is published.

SoftKey Publishing
P.O. Box 44549
Tacoma, WA 98444
(206) 531-5690

Subscription (1 year):
United States - $20.00
Canada - $28.50
Mexico - $32.50
Central America - $32.50
S. America - $37.50
Europe, Africa, others - $42.00

Sample copies of most current issue:
$3.50 (US and Canada)
$6.50 (Elsewhere)

UPDATE 1.1 SEP '81

What is it ?

Well, here it is: a HardCore Update. What is an Update ? It's a monthly mini-mag of early - release information that will be appearing in the upcoming HardCore Computing issue. In addition to the updates of important columns, it will contain news items and various notes and notices of interest to apple computerists.

The early release of an article in an Update gives readers a chance to question, correct, modify or add to that article before its final appearance in the magazine proper. Our writers do not claim to be "know-it-alls" and therefore are willing to be assisted or corrected by our readers. In fact, your assistance is necessary for the very existence of this type of magazine.

Our writers are no different from most of our readers except that they have submitted (and had accepted by us) a testimonial of their interest in computering: an article or column. Some of the articles began as letters to the editor. HardCore articles are very much like the programs we offer: they are subject to the scrutiny of our readers so that they may be improved. That means that when it is reprinted in the following issue, that same article will be somewhat different, perhaps with more illustrations, or footnotes, or whatever was lacking according to our readers response.

This particular Update contains several items awaiting your examination:

1... The promised update to Bobby's column on Copy-Protection begins with an explanation of the disk itself. We would like to know if it is too elementary or if it needs more basic explanations. Please let us know.

2... The creation of the Ultimate Adventure Program begins, naturally, with a request for ideas, assistance, complaints, and other manner of input from our readers. This will be a reader-designed adventure game that should top all adventure games and Leo is going to show you how it is written. In fact, He's writing it right now...

3... You have all probably heard of (and played) Invaders.... Well, to show that you can play graphic games on the text page, We present the premier listing of Text Invaders 1.1 (by none other than yours truly...) Written in Applesoft, this particular version of that old standby arcade favorite, has all the thump-thumps of the arcade version, and the disintegrating bunkers, exploding tanks and dying invaders so familiar to video-game enthusiasts.... but with a twist. Your gun has a value of 1 to 9, and so do the marching invaders. If you shoot one with a number higher than your gun's value, all you do it lower the invader's number. This program will be presented in Issue #2 as a lesson in Applesoft string manipulation. It is listed here for those who are familiar with strings and would like to get a jump on the next issue. We would like you to make improvements on it if you can...

Finally, we have a host of tidbits:

A...We're still are looking for programs, even ones as simple as Text Invaders! Most programs can be used to teach some aspect of computering.

B...Remember that first long and complete program you actually created by yourself? We'd like to know what it was (tic-tac-toe, or black- jack, or a recipe data file, or maybe just a telephone "list-and-search" program, or a simple arcade game) and whether you'd like to have it published in our upcoming series on "My First Program". Emphasis will be on your learning experience and how you would write that same program today. Many of our readers are just learning the various languages and would profit by discovering some of your learning experiences, problems and their resolutions. Some of the programs will be used in other columns such as: Apple Softies, Into Integer, Pascal Primer, and others.

C...We might have made a mistake in shipping out the Alerts via bulk mail. We discovered that Bulk takes around 3 weeks, and that sort-of defeats the whole purpose of having an Alert (They also may not have arrived at all, so we've reprinted the Alert notice in this update.) We are going to examine our mailing practices and try to get a second class mailing permit (which can take over half a year to get). Meanwhile, no more bulk mail shipping of Alerts.

D...It seems that we are not only disliked by many of the software vendors (software houses and retail outlets like computer stores), but some of the clubs also do not care for HardCore's Free Press Editorial Policy of "publishing all legal useful information on apple computering" (which includes, ofcourse, information on "copy-protection"). So it seems that information suppression will go on and on and on...

DISKLOCKS update: how to

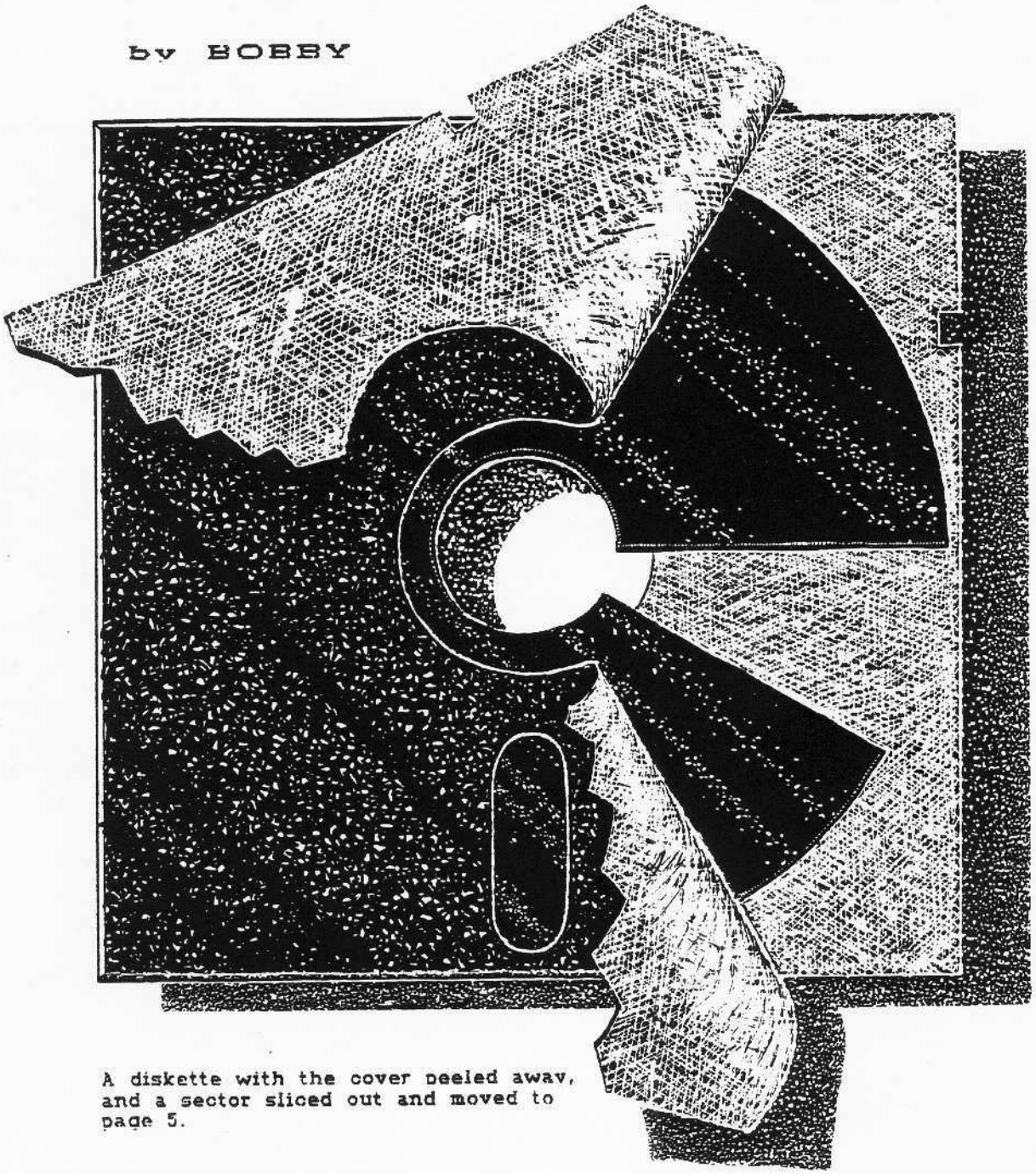
This column is the Update to the DISKLOCKS that appeared in the Premier Issue. The tables and illustrations for 16-sectors (3.3 DOS) 48 K. We plan to have tables for the 3.2 DOS when this Update appears in Issue 2, as well as any suggested changes and additions that readers send us. Copy-protection should be employed by the individual user for purposes of disk file protection, not just by software vendors who wish to discourage piracy or prevent program alterations.

by BOBBY

Copy protection can be loosely divided into 4 methods.

1. copy-protected disks
2. software protection
3. hardware protection
4. memory protection

I say "loosely" because each area overlaps the others. Method 1 usually involves changing the format of information on a disk. The disk is not copyable by normal means. In method 2 the program goes back to the disk and searches for a particular pattern. The disk is seemingly copyable but bombs after the program is run. And in method 3 a card or other hardware



A diskette with the cover peeled away, and a sector sliced out and moved to page 5.

COPY-PROTECT your disks

is used by the program. The disk is copyable but requires the hardware to run. Method 4 is really a subgroup and is employed by the other three protection schemes. It includes such things as: locking-out the reset and control "C".

The number of ways that programs can be protected has yet to be counted. Some companies even use other computers to devise disk formats that cannot be duplicated by the Apple.

In this article we will look at method one: Protecting your program by altering the disk format. In order to to understand these changed formats you need to first look at a normal disk. The flexible (or floppy) diskette can be thought of as a disc-shaped piece of recording tape, and essentially that's all it is. The reason for the flat disc shape as opposed to using a flat strip as in a tape is for speed of information retrieval. For instance, if you were to have a program stored in the middle of a tape, the computer would have to READ in all of the tape preceeding the area where your program resides. This method of information retrieval is known as "sequential access". The disk, on the other hand, is set up in such a way that the computer can go directly to a piece of data or program by scanning the disk laterally. This method of information retrieval is known as "random access".

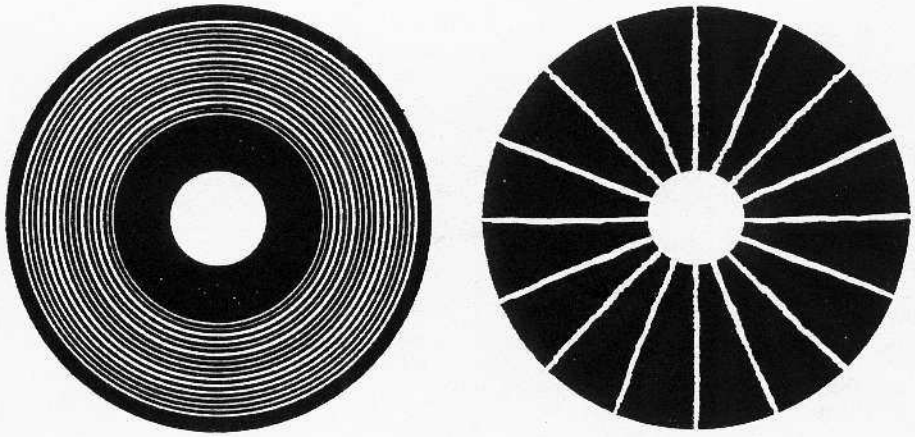


FIGURE 1

When a disk is INITialized the Disk Operating System (DOS) divides it into 35 concentric tracks (see figure 1). It then divides each track into 16 blocks called "sectors". Each sector contains an address mark and a data mark. These marks are a unique set of bytes that are used by DOS like street signs.

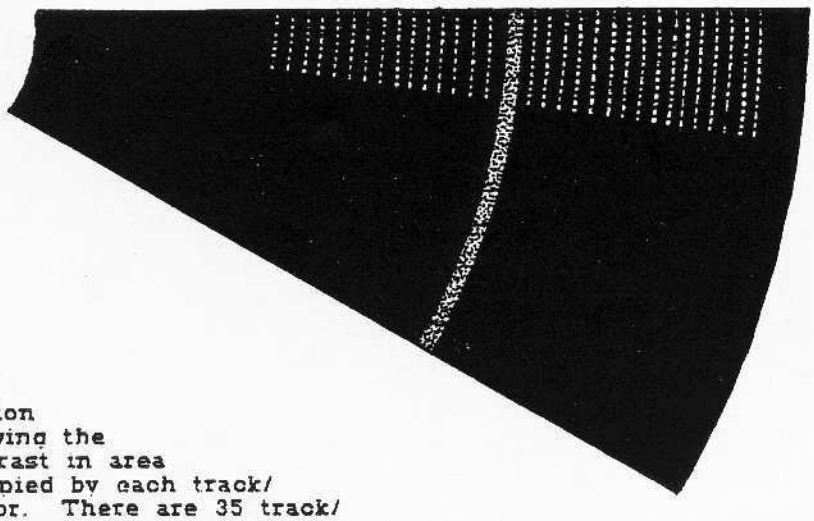
The address mark tells the DOS what track/sector it is currently READing. It contains the volume, track, sector and checksum information. The data mark surrounds the actual data. It tells the DOS where the data begins and ends and contains a checksum that is used to verify the accuracy of the data.

If you have ever tried to LOAD a program, and the disk drive started making a slight chatter, chances are that the DOS could not READ one of these markers. It then recalibrates the READ/WRITE head by moving it back to track zero and stepping (counting each track that it passes over) back out to where it was supposed to be.

The tracks are numbered from $00 (0) to $22 (34) and the sectors from $00 (0) to $0F (15). Track $00 thru track $02 (a total of three tracks: zero, one and two) contain the Disk Operating System or "DOS".

The DOS gives the Apple the ability to manipulate data on a diskette. In this program are all of the commands related to controlling the disk drive (i.e.: CATALOG, INIT, LOAD...) and a set of error messages which, unless you are either a magician or don't use the Disk II, you have probably seen before.

The disk controller card that connects the Disk II to the Apple also has a small program on it. When you BOOT a disk, this program tells the Disk II to READ track $00 (0), sector $00 (0) (remember, we start counting at zero instead of one) into memory. The program on track



A pie section showing the contrast in area occupied by each track/sector. There are 35 track/sectors (usually just called sectors) in each pie section. The grey area has been enlarged and moved to page 6.

FIGURE 2

$00, sector $00 contains the information required to READ in sectors $00 thru $09 on track $00. The program on sectors $00-$09 READs in the remaining information on track $00 - $02. Thus the Apple LOADs the DOS.

At this point DOS takes over and RUNs the program you have INITialized the disk with.

In order to find your HELLO program DOS goes to the Volume Table of Contents (VTOC) and Directory located on track $11 (17). The VTOC or "bit map" shows which sectors are in use and which are free. The Directory begins on sector $0F (15) and continues down to sector $01 (1). The VTOC and Directory are used by DOS whenever you SAVE or DELETE a file. The Directory contains a list of all the files on the disk. Each entry contains a pointer to the track/sector list, file-locked and file-type code, file-name and file-size. The track / sector list is a list of track/sector pairs that are used to store that program. This is why SAVEing a blank file always takes 2 sectors. One for the blank file and one for the track-/sector list.

DOS formats a track by first writing a unique byte called a "sync byte". This byte (normally $FF) allows the Disk II hardware to synchronize with the data on the disk. DOS then WRITEs an address mark, some more sync bytes and the data mark. At this time the data field is full of $00's.

The following is a normal address mark:

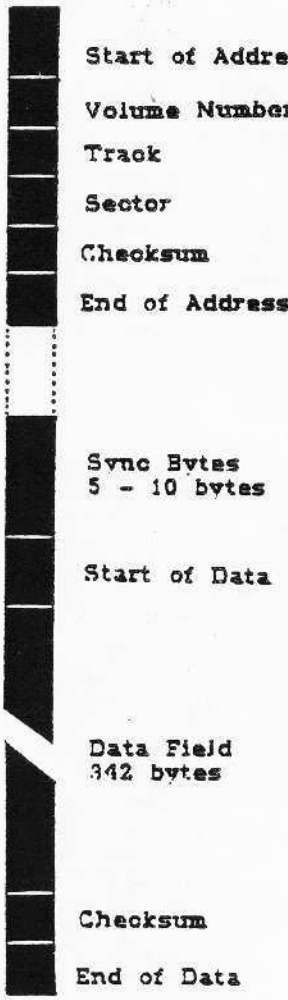
D5AA96FFFEAABBAEAAFBEFDEAAEB

It can be broken down into:

Start of address	- D5 AA 96
Volume number	- FF FE
Track	- AA BB
Sector	- AE AA
Checksum	- FB EF
End of address	- DE AA EB

FIGURE 3

SECTOR FORMAT



4 + 4 CONVERSION CHART

AA+AA=00	AF+BE=1E	BE+FA=78	EE+BE=9C
AA+AB=01	AF+BF=1F	BE+FB=79	EE+BF=9D
AA+AE=04	AF+EA=4A	BE+FE=7C	EE+EA=C8
AA+AF=05	AF+EB=4B	BE+FF=7D	EE+EB=C9
AA+BA=10	AF+EE=4E	BF+AA=2A	EE+EE=CC
AA+BB=11	AF+EF=4F	BF+AB=2B	EE+EF=CD
AA+BE=14	AF+FA=5A	BF+AE=2E	EE+FA=D8
AA+BF=15	AF+FB=5B	BF+AF=2F	EE+FB=D9
AA+EA=40	AF+FE=5E	BF+BA=3A	EE+FE=DC
AA+EB=41	AF+FF=5F	BF+BB=3B	EE+FF=DD
AA+EE=44	BA+AA=20	BF+BE=3E	EF+BA=9A
AA+EF=45	BA+AB=21	BF+BF=3F	EF+BB=9B
AA+FA=50	BA+AE=24	BF+EA=6A	EF+BE=9E
AA+FB=51	BA+AF=25	BF+EB=6B	EF+BF=9F
AA+FE=54	BA+BA=30	BF+EE=6E	EF+EA=CA
AA+FF=55	BA+BB=31	BF+EF=6F	EF+EB=CB
AB+AA=02	BA+BE=34	BF+FA=7A	EF+EE=CE
AB+AB=03	BA+BF=35	BF+FB=7B	EF+EF=CF
AB+AE=06	BA+EA=60	BF+FE=7E	EF+FA=DA
AB+AF=07	BA+EB=61	BF+FF=7F	EF+FB=DB
AB+BA=12	BA+EE=64	EA+AA=80	EF+FE=DE
AB+BB=13	BA+EF=65	EA+AB=81	EF+FF=DF
AB+BE=16	BA+FA=70	EA+AE=84	FA+EA=E0
AB+BF=17	BA+FB=71	EA+AF=85	FA+EB=E1
AB+EA=42	BA+FE=74	EA+BA=90	FA+EE=E4
AB+EB=43	BA+FF=75	EA+BB=91	FA+EF=E5
AB+EE=46	BB+AA=22	EA+BE=94	FA+FA=F0
AB+EF=47	BB+AB=23	EA+BF=95	FA+FB=F1
AB+FA=52	BB+AE=26	EA+EA=C0	FA+FE=F4
AB+FB=53	BB+AF=27	EA+EB=C1	FA+FF=F5
AB+FE=56	BB+BA=32	EA+EE=C4	FB+EA=E2
AB+FF=57	BB+BB=33	EA+EF=C5	FB+EB=E3
AE+AA=08	BB+BE=36	EA+FA=D0	FB+EE=E6
AE+AB=09	BB+BF=37	EA+FB=D1	FB+EF=E7
AE+AE=0C	BB+EA=62	EA+FE=D4	FB+FA=F2
AE+AF=0D	BB+EB=63	EA+FF=D5	FB+FB=F3
AE+BA=18	BB+EE=66	EB+AA=82	FB+FE=DE
AE+BB=19	BB+EF=67	EB+AB=83	FB+FF=DF
AE+BE=1C	BB+FA=72	EB+AE=86	FE+EA=E8
AE+BF=1D	BB+FB=73	EB+AF=87	FE+EB=E9
AE+EA=48	BB+FE=76	EB+BA=92	FE+EE=EC
AE+EB=49	BB+FF=77	EB+BB=93	FE+EF=ED
AE+EE=4C	BE+AA=28	EB+BE=96	FE+FA=F8
AE+EF=4D	BE+AB=29	EB+BF=97	FE+FB=F9
AE+FA=58	BE+AE=2C	EB+EA=C2	FE+FE=FC
AE+FB=59	BE+AF=2D	EB+EB=C3	FE+FF=FD
AE+FE=5C	BE+BA=38	EB+EE=C6	FF+EA=EA
AE+FF=5D	BE+BB=39	EB+EF=C7	FF+EB=EB
AF+AA=0A	BE+BE=3C	EB+FA=D2	FF+EE=EE
AF+AB=0B	BE+BF=3D	EB+FB=D3	FF+EF=EF
AF+AE=0E	BE+EA=68	EB+FE=D6	FF+FA=FA
AF+AF=0F	BE+EB=69	EB+FF=D7	FF+FB=FB
AF+BA=1A	BE+EE=6C	EE+BA=98	FF+FE=FE
AF+BB=1B	BE+EF=6D	EE+BB=99	FF+FF=FF

The volume, track, sector and checksum are in 4+4 coded format (see 4+4 conversion chart). This means that 4 bits in each byte is actual data. The first byte is rotated left and logically ANDed with the second byte to recover the data.

The data mark consists of:

Start of Data -- D5 AA AD
Data Field -- (342 bytes)
Checksum -- (2 bytes)
End of Data -- DE AA EB

The data field is encoded in a 2+6 format. Six bits of each byte is valid data.

For those of you who are interested in more details and think I glossed over some things (and I did), the best book I have seen that talks about DOS is BENEATH APPLE DOS by Don Worth and Pieter Lechner. A worthwhile addition to any Apple computerist's library.

There are two things to consider when you decide to protect your disk. Protecting against normal DOS copiers and protecting against bit (nibble) copiers.

The first is fairly simple. Almost any change in the normal disk format will cause DOS to generate an I/O ERROR. The end-of-address and end-of-data bytes (DE AA) are not used in the initial BOOT. Therefore, changing these bytes will allow a disk to BOOT but not allow normal DOS to copy it. The easiest method of accomplishing this is to change the appropriate locations in DOS and INITialize a disk with the changes. However, LOADing anything other than your "hello" program onto this disk could be tedious. You would have to change the DOS back to normal, LOAD your program from a normal disk then change those same DOS locations and SAVE the program to your protected disk. Another method would be to LOAD the programs from the cassette input and SAVE them to the disk. The best method would be to use Muffin to upLOAD your programs from a normal disk. This works because Muffin uses an internal DOS (RWTS) image to READ data and the resident DOS to WRITE data. A more elegant method is afforded by taking advantage of the way that DOS WRITEs data to the disk. When DOS WRITEs a sector, it first looks for the address mark. When it has found the correct mark for the sector it wants, DOS WRITEs a whole new data mark. It first WRITEs 5 sync bytes then the complete data mark with the new data. Every time you SAVE data to a disk this process is repeated. NOTE:

The address mark is only written once when the disk is INITialized. A program could be written that would READ the data from a normal disk and WRITE it to another in a changed format. This changed disk would not be copyable by any normal DOS. Table 1 is a list of address and data mark locations and their normal values for a 48K Apple.

CAUTION: Always change both the READ and WRITE locations for any byte you alter. Also, not all values are valid. See table 2 for a list of valid bytes to use when making changes.

The second part of disk protection involves bit or nibble copiers. These programs do not depend on a rigid structure of address and data marks to find valid data. Instead they make as few assumptions about the disk format as possible. These copiers use pattern recognition to find the data. They are not as reliable as a normal DOS copy program since they don't use checksums. Many of the current bit copiers do make one assumption. They look for a $FF sync byte. These copiers can be defeated by INITializing a disk with a different sync byte. I am sure that the new crop of bit copiers will have corrected this oversight but until then this is a valid protection method. The location of the sync bytes are given in table 1. CAUTION: not all values are valid as sync bytes. See table 2 for a list of usable bytes.

Upcoming Special Program:
A NIBBLER
that allows you to examine the encoded data on any disk.

TABLE 1

DOS 3.3 ADDRESS AND DATA MARK LOCATIONS

DOS 3.3	READ LOCATIONS -HEX-	-DECIMAL-	WRITE LOCATIONS -HEX-	-DECIMAL-
START OF ADDRESS	B955 D5	47445 213	BC7A D5	48250 213
	B95F AA	47455 170	BC7F AA	48255 170
	B96A 96	47466 150	BC84 96	48260 150
END OF ADDRESS	B991 DE	47505 222	BCAE DE	48302 222
	B99B AA	47515 170	BCB3 AA	48307 170
START OF DATA	B8E7 D5	47335 213	B853 D5	47187 213
	B8F1 AA	47345 170	B858 AA	47192 170
	B8FC AD	47356 173	B85D AD	47197 173
END OF DATA	B935 DE	47413 222	B89E DE	47262 222
	B93F AA	47423 170	B8A3 AA	47267 170
SYNC BYTE			BC60 FF	
SYNC BYTE				

TABLE 2

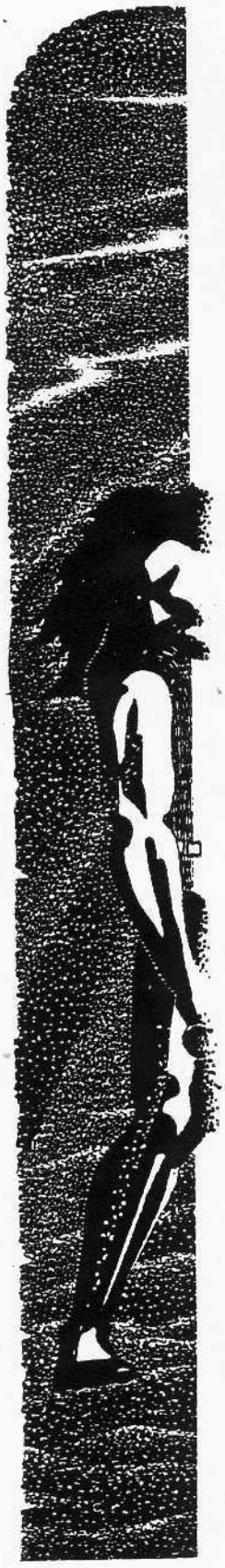
Legal values that are used to change the start and end bytes of the address and data marks

HEX	DEC	HEX	DEC
96	150	D5*	213
97	151	D6*	214
9A	154	D7*	215
9B	155	D9	217
9D	157	DA*	218
9E	158	DB*	219
9F	159	DC*	220
A6	166	DD*	221
A7	167	DE*	222
AA*	170	DF*	223
AB*	171	E5	229
AC*	172	E6	230
AD*	173	E7	231
AE*	174	E9	233
AF	175	EA*	234
B2	178	EB*	235
B3*	179	EC*	236
B4*	180	ED*	237
B5*	181	EE*	238
B6*	182	EF*	239
B7	183	F2	242
B9	185	F3	243
BA*	186	F4*	244
BB*	187	F5*	245
BC*	188	F6*	246
BD*	189	F7*	247
BE*	190	F9	249
BF*	191	FA*	250
CB	203	FB*	251
CD	205	FC*	252
CE	206	FD*	253
CF	207	FE*	254
D3	211	FF*	255

* - Use only these values as sync bytes

part 1 INTRODUCTION TO HARDCORE'S

Customizable Modular



This is not a column on the principles of adventuring or a critique on adventure programs. It is, rather, a series of articles and programs that, when completed, will result in a customizable, hi-res color action (arcade--style) and text adventure game for the AII+ 48K with Applesoft in ROM and disk drive. It will consist of numerous sub-programs (Modules) that will be sequentially RUN or accessed so that customizing becomes a simple matter of providing different Modules to make it a different game.

There will be 8 installments to be published in the next three issues and intervening updates beginning with this one.

(1) Update 1.1 -- this article.. Introduction to Role-Playing Adventure Computer Games.
(2) Issue #2 -- Oct. Maze-maker, Personality Profiler, and Map-displayer programs.
(3) Update 2.1 -- Nov. Moving through a hi-res maze (using shape tables), adding text files of maze data.
(4) Update 2.2 -- Dec. The command interpreter part I: 3-mode network and a large vocabulary!
(5) Issue #3 -- Jan. The command interpreter part II: The command code decoder and associated text files.
(6) Update 3.1 -- Feb. Preliminary RUNning, adding additional features: Arcade modules, HI-res pics.
(7) Update 3.2 -- Mar. Customizing the command and description textfiles and variables, adding another set of arcade modules.
(8) Issue #4 -- Apr. RUNning the final adventure (How all the pieces fit together).

This "adventure" will be a true role-playing game utilizing a "maze" as a background through which you must travel in quest of your elusive goals.

So... How is this different from other adventures? Simple. This one is being written after the others have been out for a while and we have learned from the mistakes of those pioneers. It has NOT already been written. This means that reader input, whether they be requests for game characteristics, complaints, or actual program changes, will be considered in its design. You are encouraged to write in so that this game becomes a reader-designed game and not a pre- packaged, "copy-protected" "black- box" game like so many now available. For those advanced in programing, your input is needed. And for those just beginning to learn Apple--soft, this is a great way to supplement your growing familiarity with this version of BASIC

This game will be written in discrete modules, or stages, that can be easily debugged of typos. It can, and will be explained in detail (that's why it will take so long to publish). You will learn how it works, why it works, and how to change it to fit your own ideas of what a true adventure game should be.

It will use the hi-res page to display an animated color map and all the arcade (conflict) scenarios as well as any of the detailed hi-res pictures (pics) that will be loaded at important junctions of the game sequence. (These pics can either be purchased from the HARDCORE library or can be drawn using almost any of the commercial hi-res graphics packages or possibly with our own... At that time, we will be actively soliciting hi-res pics to illustrate the games. These pics will be purchased from the reader-artists and offered for sale from the program library. At that time we will also be purchasing your own customized modules.

All of this means that this adventure will actually be a combination of adventure, "interactive fiction", role--playing, maze - marching, and arcade "shoot-em-ups" linked together in one consistent but highly variable game.

The game will be almost entirely written in Applesoft, so that user modification is fairly easy to accomplish. (Some of the arcade sequences may have machine language subroutines, but these will be poked in from the Applesoft program. The same goes for the numerous shape tables used to animate the maze Map.)

The complex command-parser or interpreter will probably be quite large so that it can handle almost any command the stalwart adventurer can type in. And because I plan on using both hi-res pages, all the rest of the program will be stored as text-files on the disks. That means frequent disk access (and possibly a modified DOS???).

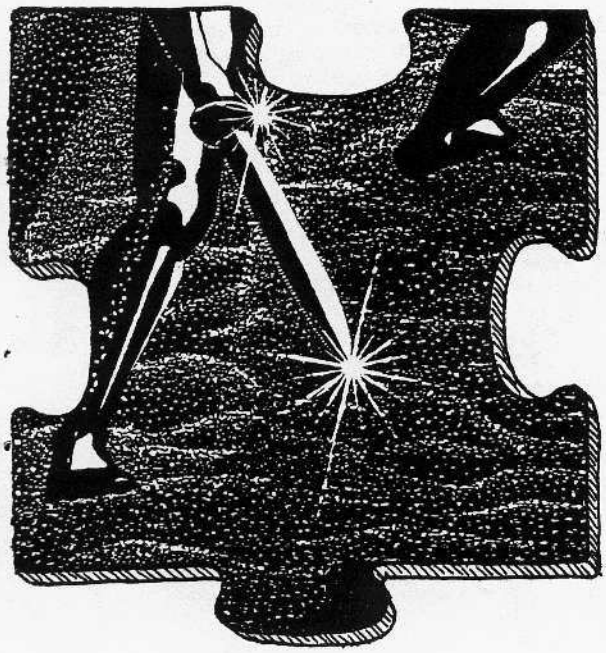
Each game-sequence will initially begin with a randomized Maze-maker and Personality-maker where you may create a set of characters to occupy the maze with you and interact with you. These game personalities will roam the maze just as you do, some seeking you or your possessions , some accompanying you, some avoiding you, and some not really caring whether you come or go. You may make them your friends or your enemies by your actions and inactions.

by B (Leo) BRYTE

Adventure-arcade Game

Here, you may choose the inanimate objects to occupy the mazes, determine their importance and power, and select the randomizing factor that will change your choices so that the element of surprise is not lost. Here also, you must select your own role, power, age, rank, sex, fantasy-nationality, height, weight, build, skills, knowledge, and beliefs (religion). Since this is a role-playing game. These characteristics will change as you travel through the maze.

All of the important elements of the game will be saved to a text file each time an important decision is made by you. And the entire "map" is saved as one of the pics as it is updated in your travels. Another person may play the same game yet that person will have a different maze, different personalities, different goals and ambitions, even though the "same" game is being played... That's because the Maze-maker and Personality-maker will customize the game for that person. Your map,consistent in every game you play, will be different from the other person's map. That means that you can continue to play the game you started, or create a new one for yourself, or for a friend. In this way, you can even create a scenario to challenge a friend's adventuring spirit. This game is not all "kill or be killed". As a role-playing game, all the relationships are more subtle and complex than standard adventure games. In this adventure, you can cultivate lasting friendships , help them build bridges and fortresses, help defeat their enemies, fall in love, have children. You can even change your goals, since no score is kept. Or you may decide that family life is boring, and continue your quest before you die of old age...

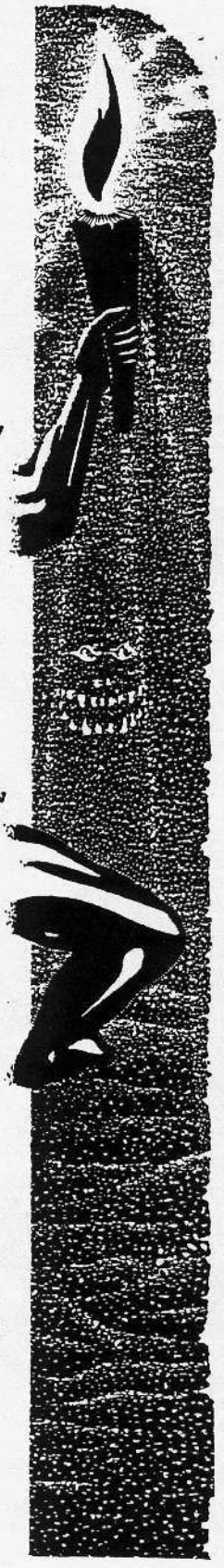


There is no need to greedily accumulate wealth, because sometimes it can lead to your detriment. Some valuables were stolen, and their rightful owners are searching for them and the "thief". Some valuables are cursed, bringing doom to their possessors. Others are evil, or good, or simply fake. Your wealth is valuable in acquiring land, assistance, property and goods for a long voyage , but sometimes , fortune falls only on the pauper and begger.

Although you may make the primary choices, the program will decide what it is you can actually do. If you are drunk, or injured, or otherwise incapacitated, your command to fight will lead to early defeat and imprisonment. If you are cursed, your best intentions will lead to disaster until you remove the evil aura.

Your adventure is an odessey and is recorded in a special history text file (if you choose) so that you may review your past for clues to your present predicament...(what led you to this dilemma? Was it because you failed to help the dying monk, or was it because you helped the woman in the bar? How can you make up for your act?). This chronicle will have dates beginning with the year 2000 D.A. (During Adventure), and will also be documented with national and religious holidays, birthdates of important adventure characters (like the king's ball, which you must attend), and other miscellaneous historical notes which may offer clues to your quest's attainment.

Is this too much to ask of an adventure? No. And there must be more...

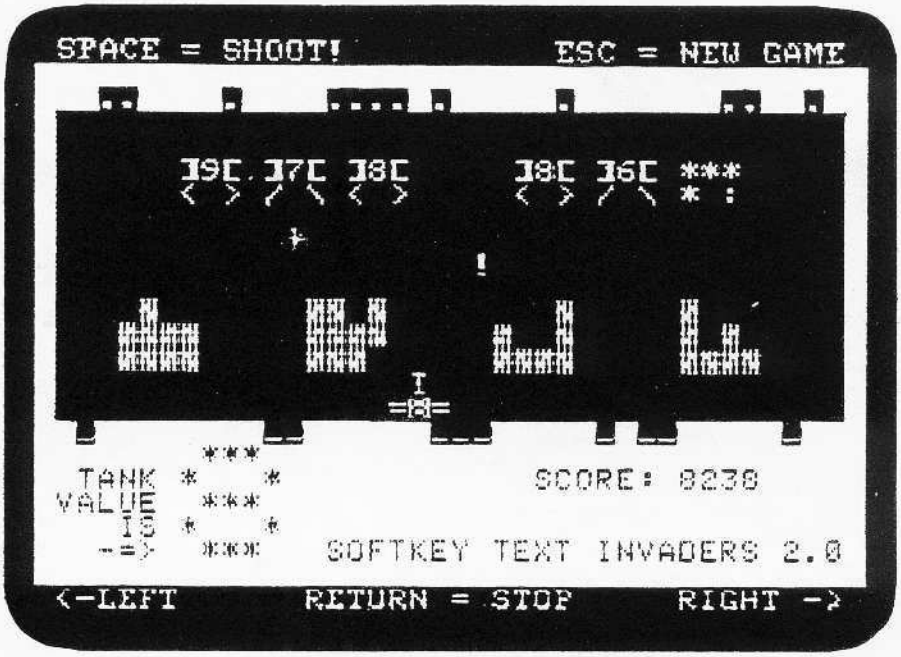


Next time: How to write a variable maze--maker and personality (character) profiler (the initial module) and a short program to display the "map" so that we can examine the maze structure in order to understand exactly what a maze consists of and how it can be customized. The map-maker program will use shape tables DRAWn, SCALEd, and ROTated on Hi-Res page 1 (hgr) so you might want to study that section in your apple-soft manual.

Until next month, think about what else this adventure needs and send in your ideas to: B. (leo) Bryte, c/o HardCore Computing.

TEXT invaders 2.0

by Bev HAIGHT



Text Invaders 2.0 was written as an exercise in applesoft string$ manipulation and evolved into the 2.0 version listed here. It will be printed in HardCore Computing #2 as the first Apple Softie column where it will be listed in "RUNning" segments for easy debugging. It will then be a lesson in using Applesoft string$ functions such as MID$, LEFT$, RIGHT$, LEN, STR$, VAL, etc. But for now, it is presented as an interesting arcade-style game that you can type in now.

Like the arcade counterpart, Text Invaders has the "thump! thump!" of advancing invaders. And their peculiar march back and forth and down is duplicated in this text version by using the two slashes and these: "< >" as the legs (see illustration of screen image).

However, unlike the arcade version, these invaders are not destroyed by just any missile strike. Oh no... You see, your tank has a "hit power" of 1 through 9. The invaders also have that range of "hit power". If your hit power is less than that of the invader your missile strikes, that invader is not destroyed, only lowered in hit power. Each time the invader drops one of its bombs (a plus sign in this version), its hit power is reduced by one until it reaches zero... when it vanishes. And each time you are destroyed by one of the descending bombs, your tank also loses a hit power until you reach zero... when the game ends.

Your skill choice, elected at the beginning of the game, determines the number of advancing invaders. Pick level 1 (beginner) and you get only five invaders. Wipe them out and your skill level advances by one and new invaders appear, closer this time. First time out, don't pick 5 (expert).

The bunkers that protect you from the falling bombs will disentegrate slowly, but if you shoot through it, you'll carve a neat hole all the way through.

The bombs that make it past the bunkers and miss you will dig a hole in the ground. But don't worry...it's just there for the effect. Likewise, your overshot missiles will dig holes at the top, too.

What are the keyboard commands?

The two arrow keys determine the tank's direction of travel. The RETURN key will make your tank stop. And the space bar will fire a missile (an exclamation mark !). Refiring a missile will cause the destruction of any missile already in flight. Finally, the ESC key will start a new game.

The listing is in an experimental format. Because the program is Applesoft, only the first two characters of any variable name are important. Those two characters have been left capitalized while the remainder of the variable name is in lower case. All REMarks have been removed, but you can type them in by using a line number just one number below the line you are explaining. In this way, all remarks will have 9 as a final digit.

Since there are many PRINT statements, all spaces within such statements have been replaced by a squiggle so that space counting is simpler and easier.

There are three "destruction modes" for the invaders, depending on whether your missile has hit them on the left or right side, or smack in the middle. Scoring also depends on where you hit them. A direct hit is best.

There are many improvements that can be added to it... which I leave to those interested. Some that I suggest:

Make more than one row of invaders.
Let them drop more bombs.
Let your tank shoot more than one missile.
Use the entire screen for a battle field.
Use the paddles.

And most of all... Compile it with one of the new Applesoft compilers now available...and speed it up! Of course, you'll have to change some of the sound routines. And you'll have to slow down their march toward you since the game will be running incredibly fast once it is compiled.

Use a character generator to change the strings into unique hi-res shapes more like the arcade versions.

One last note: Notice the way that the SCRN command is used on the text page...

```
          HARDCORE Computing
           Text Invaders 2.0
            Copyright 1981
         by SoftKey Publishing
            P.O. Box 44549
           Tacoma, WA 98444


REMs replaced by these symbols:

'============ SUBROUTINE ==============

'-------Elements In A Subroutine--------

'**********CENTRAL PROGRAM*************

'.....................................
'Spaces in PRINT Statements replaced by ~
'.....................................


0     REM TEXT INVADERS 2.0
1     GOTO 10000 : REM   C.1981 SOFTKEY
PUBLISHING, POB 44549, TACOMA, WA 98444

'==========GET ASCII OF X,Y SCRN=========
10    XXkey% = X% - 1 : YYkey% = 2 * (
      Y% - 1 )
20    XKey% =  SCRN ( XXkey% , YYkey% )
      + 16 *  SCRN ( XXkey% , YYkey% +
      1 ) : RETURN

'============MOVE TANK MISSILE==========
30    VTAB TVray% : HTAB THray% : PRINT
      "~" ; : RETURN
35    VTAB TVray% : HTAB THray% : PRINT
      "!" ; : RETURN
40    IF TVray% > 0 THEN  GOSUB 30 :
      HTAB THray% : PRINT "~" ;
45    THray% = HGun% : TVray% = VGun% -
      2 : GOSUB 35 : RETURN
50    IF TVray% < 1 THEN  RETURN
52    GOSUB 30 : TVray% = TVray% - 1 :
      GOSUB 35 : IF TVray < 5 THEN
      GOTO 30
55    X% = THray% + 1 : Y% = TVray% - 1
      : GOSUB 10

'=======HAS MISSILE HIT ANYTHING?========
'----------(SPACE) move on up----------
60    IF XKey% = 160 THEN  RETURN

'----------( + ) hit a bomb-----------
65    IF XKey% = 171 THEN 800

'------not near invaders, so return-----
70    IF TVray% < VInvader% OR TVray% >
      VInvader% + 1 THEN RETURN
72    SS% = 1

'----( ] < / ) hit invader left side----
75    IF XKey% = 221 OR XKey% = 188 OR
      XKey% = 175 THEN GOSUB 600 : RAn%
      = X% - HInvader% : GOTO 910

'----( [ > \ ) hit invader right side---
80    IF XKey% = 219 OR XKey% = 190 OR
      XKey% = 220 THEN GOSUB 620 : RAn%
      = X% - HInvader% - 2 : GOTO 910

'---( anything else ) hit dead center---
85    GOSUB 610 : RAn% = X% - HInvader%
      - 1: GOTO 910

'--------( destroy the missile )--------
90    FOR A = 1 TO 4 : VTAB TVray% :
      HTAB THray% : PRINT "$" ; : BUZZ
      = PEEK ( NOise ) : FOR B = 1 to 5
      : NEXT B : HTAB THray% : PRINT
      "~" ; : BUZZ = PEEK ( NOise ) :
      NEXT A : TVray% = 0 : RETURN


'=========HAS KEYBOARD BEEN HIT?=========
100   KEy% =  PEEK ( KEy ) : IF KEy% >
      127 THEN POKE STrobe , 0
110   IF DIsplay% AND FShoot% < 0 THEN
      KEy% = 160


'----------(ESC) start new game---------
120   IF KEy% = 155 THEN GO% = 1 : GOTO
      10000

'-------(SPACE) shoot off cannon--------
150   IF KEy% = 160 THEN AU% = 1 :
      GOSUB 40

'-------(RETURN) stop tank motion-------
155   IF KEy% = 141 THEN TTravel% =
      PAuse%

'--------( -=> ) tank goes right--------
160   IF KEy% = 149 THEN TTravel% =
      RIght% : DIsplay% = 0 : GOTO 200

'--------( <=- ) tank goes left---------
170   IF KEy% = 136 THEN TTravel% =
      LEft% : DIsplay% = 0 : GOTO 300

'----------( make tank travel )---------
190   ON TTravel% GOTO 200,300,350


'********** TANK GOES RIGHT ************
200   HGun% = HGun% + 1 : IF HGun% > 36
      THEN HGun% = 36 : TTravel% =
      LEft% : RETURN
```

```
********** MOVE TANK ******************
------(clear old tank from screen)-----
210   VTAB VGun% : HTAB 1 : CALL  -868 :
VTAB VGun% - 1 : CALL  -868

----------( make new tank )----------
220   VTAB VGun% : HTAB HGun% - 1 :
      PRINT "=" ; : INVERSE : PRINT
      POwer% ; : NORMAL : PRINT "=" ;
230   VTAB VGun% - 1 : HTAB HGun% :
      PRINT "I" ; : RETURN

************ TANK GOES LEFT ***********
300   HGun% = HGun% - 1 : IF HGun% < 3
      THEN HGun% = 3 : TTravel% =
      RIght% : RETURN

************ TANK STOPS **************
350   GOTO 210


========== DISPLAY TANK POWER =========

400   B% = 1 : FOR A = 19 TO 23 : FOR
      AA = 1 TO 5
410   VTAB A : HTAB AA + 5
420   PIece$ = MID$ ( POwer$ ( POwer% )
      , B% , 1 )
430   INVERSE
450   PRINT PIece$ ; : NORMAL : B% = B%
      + 1 : NEXT AA , A : RETURN

========== MOVE INVADER BOMBS =========

500   IF IHray% = 0 THEN 300
510   IVray% = IVray% + 1 : X% = IHray%
      + 1 : Y% = IVray% : GOSUB 10

----------( Just open space )----------
520   IF XKey% = 160 OR XKey% = 170
      THEN 700

---------( bomb hits missile )---------
530   IF XKey = 161 THEN 800

----------( bomb hits bunker )---------
540   IF XKey% = 92 OR XKey% = 111 THEN
      850


-----(inverse SPACE) bomb at bottom----
550   IF IVray% > 17 THEN750

-----------( tank is hit )-------------
570   INVERSE : GOSUB 13200 : POwer% =
      POwer% - 1 : IF POwer% = 0 THEN
      21000
580   VTAB VGun% - 2 : HTAB IHray% :
      IHray% = 0 : PRINT "~" ;
590   GOSUB 400 : RETURN

======= INVADER HIT ON LEFT SIDE =======
600   FOR ZAP = 0 TO 2 : VTAB VInvader%
      + 1 : HTAB X% - 1 : GOSUB 630 :
      VTAB VInvader% : HTAB X% - 1 :
      GOSUB 630 : HTAB X% : GOSUB 630 :
      HTAB X% + 1 : GOSUB 630 : VTAB
      VInvader% + 1 : HTAB X% + 1 :
      GOSUB 630 : HTAB X% : GOSUB 630
      : NEXT ZAP
605   BPoints% = 0 : GOSUB 1600 : GOTO
      640
======== INVADER HIT DEAD CENTER =======
610   FOR ZAP = 0 TO 2 : VTAB VInvader%
      : HTAB X% - 1 : GOSUB 630 : HTAB
      X% - 2 : GOSUB 630 : HTAB X% :
      GOSUB 630 : VTAB VInvader% + 1 :
      HTAB X% - 2 : GOSUB 630 : HTAB X%
      : GOSUB 630 : HTAB X% - 1 : GOSUB
      630 : NEXT ZAP
615   BPoints% = 10 : GOSUB 1600 : GOTO
      640
======= INVADER HIT ON RIGHT SIDE ======
620   FOR ZAP = 0 TO 2 : VTAB VInvader%
      + 1 : HTAB X% - 1 : GOSUB 630 :
      VTAB VInvader% : HTAB X% - 1 :
      GOSUB 630 : HTAB X% - 2 : GOSUB
      630 : HTAB X% - 3 : GOSUB 630 :
      VTAB VInvader% + 1 : HTAB X% - 3
      : GOSUB 630 : HTAB X% - 2 : GOSUB
      630 : NEXT ZAP
625   BPoints% = 0 : GOSUB 1600 : GOTO
      640

---------- Destroy Invader -----------
630   PRINT ZAP$ (zap) : BUzz = PEEK
      ( NOise ) + PEEK ( NOise ) :
      RETURN
640   THray% = 0 : TVray% = 0 : GOTO 370


==========CONDENSE INVADERS==========

-------( Condense Right Side )---------
650   IF MID$ ( INvader$ (1) , LEN
      ( INvader$ (1) ) - 2 , 1 ) = "~"
      THEN  FOR AA = 0 TO 2 : INvader$
      (AA) =  LEFT$ ( INvader$ (AA) ,
      LEN ( INvader$ (AA) ) - 4) : NEXT
      AA : IF  LEN ( INvader$ (1) ) > 4
      THEN  GOTO 650

-------( Condense Left Side )----------
660   IF  MID$ ( INvader$ (1) , 4 , 1 )
      = "~" THEN FOR AA = 0 TO 2 :
      INvader$ (AA) = "~" +  RIGHT$
      ( INvader$ (AA) , LEN ( INvader$
      (AA) ) - 5 ) : NEXT AA :
      HInvader% = HInvader% + 4 : IF
      LEN ( INvader$ (1) ) > 6 THEN 660
690   RETURN
```

```
============MOVE INVADER RAY==========
700   VTAB IVray% : HTAB IHray% : PRINT
      "+" ; : VTAB IVray% - 1 : HTAB
      IHray% : PRINT "~" ; : RETURN

========INVADER RAY HITS BOTTOM========
750   VTAB IVray% - 1 : HTAB IHray% :
      PRINT "~" ; : FOR T = 1 TO 5
770   INVERSE : VTAB IVray% : HTAB
      IHray% : PRINT "^" ; : GOSUB 1530
780   NORMAL : HTAB IHray% : PRINT CHR$
      (223) ; : GOSUB 1530
790   NEXT T : IHray% = 0 : RETURN

============ BOMB HITS RAY ============
800   VTAB IVray% : HTAB IHray% : PRINT
      "~" ; : IVray% = 0 : IHray% = 0
810   FOR A = 1 TO 10 : BUzz =  PEEK (
      NOise ) : NEXT A : VTAB TVray% :
      HTAB THray% : PRINT "*" ;
820   FOR A = 1 TO 10 : BUzz =  PEEK (
      NOise ) : VTAB TVray% : HTAB
      THray% : PRINT "#" ; : FOR B = 1
      TO 5 : NEXT B : HTAB THray% :
      PRINT "~" ; : NEXT A
830   TVray% = 0 : THray% = 0 : RETURN

============ BOMB HITS BUNKERS==========
850   FOR T = 1 TO 6 : VTAB IVray% - 1
      : HTAB IHray% : INVERSE : PRINT
      "~" ; : VTAB IVray% : HTAB IHray%
      : PRINT "%" ; : GOSUB 1530
860   VTAB IVray% - 1 : HTAB IHray% :
      NORMAL : PRINT "~" ; : VTAB
      IVray% : HTAB IHray% : PRINT "~"
      ; : GOSUB 1530 : NEXT
880   IHray% = 0 : RETURN

========== INVADER RAY ATTACK =========
900   R% =  RND (1) * N% : RAn% = R% *
      4 + 3 : IF LEN ( INvader$ (1) ) <
      5 THEN  VTAB VInvader% : HTAB 1 :
      CALL -868 : VTAB VInvader% + 1 :
      CALL -868
905   IF RAn% >  LEN ( INvader$ (1) )
      THEN  RETURN
910   TEmp$ = MID$ ( INvader$ (1) ,
      RAn% + 1 , 1 )
920   TEmp% =  VAL ( TEmp$ ) : IF TEmp$
      = "~" THEN 900
930   IF TEmp% > 9 THEN TEmp% = TEmp% /
      10 : GOTO 930
935   IF SS% THEN IF TEmp% < = POwer%
      THEN TEmp% = 1
940   TEmp% = TEmp% - 1 : QInvader% =
      IVray% + 1 : TEmp$ = STR$ ( TEmp%
      ) : QHit% = RAn% + HInvader% : IF
      TEmp% < 1 THEN TEmp$ = "~"
945   IF SS% THEN 960
950   IVray% = VInvader% + 1 : IHray% =
      RAn% + HInvader% : GOSUB 700 :
      VTAB VInvader% : HTAB IHray% :
      INVERSE : PRINT TEmp$ ; : NORMAL
960   SS% = 0
970   INvader$ (1) =  LEFT$ ( INvader$
      (1) , RAn% ) + TEmp$ +  MID$ (
      INvader$ (1) , RAn% + 2 )
980   IF TEmp$ = "~" THEN FOR AA = 0 TO
      2 : INvader$ (AA) = LEFT$ (
      INvader$ (AA) , RAn% - 1 ) +
      "~~~~" + MID$ ( INvader$ (AA) ,
      RAn% + 3 ) : NEXT AA
990   GOTO 650

********** MAIN GAME SEQUENCE *********

1000  IF VInvader% = 15 THEN 21000

************ RHYTHMIC TIMER ***********
1010  FOR RHythm = 0 TO 100 STEP
      VInvader% * 10

*********** CHECK KEYBOARD *************
1020  GOSUB 100

********** MOVE INVADER BOMB ***********
1040  GOSUB 500
1050  IF AU% THEN  GOSUB 50
1080  NEXT RHythm
1090  GOTO 2000

========== >>thump<< >>bump<< =========
1500  FOR A = 0 TO 16 - VInvader% : IF
      FRhythm% > 0 THEN 1520

--------------->> thump ! <<------------
1510  ZZ =  PEEK ( NOise ) -  PEEK
      ( NOise ) : GOTO 1590

---------------->> bump ! <<------------
1520  ZZ =  PEEK ( NOise ) +  PEEK
      ( NOise ) : GOTO 1590

----------------- buzzzz --------------
1530  ZZ =  PEEK ( NOise ) : RETURN
1590  NEXT A : RETURN

===============PRINT POINTS============
1600  TPoints% = BPoints% * SKill% +
      POwer% + TPoints%
1610  VTAB 19 : HTAB 30 : INVERSE :
      PRINT "~~~~~~" ; : HTAB 30 : PRINT
      TPoints% ; : NORMAL : RETURN

==========<INVADER MOVEMENT>===========
2000  IF LEN ( INvader$ (1) ) < 5 THEN
      20000
2010  IF ITravel% = LEft% THEN 2500
```

```
********** INVADERS GO RIGHT **********
2400  HInvader% = HInvader% + 1 : IF
      HInvader% > 37 -  LEN ( INvader$
      (1) ) THEN ITravel% = LEft% :
      VTAB VInvader% : HTAB 1 : CALL
      -868 : IF SKill% < 3 THEN
      VInvader% = VInvader% + 1
2410  VTAB VInvader% : HTAB 1 : CALL
      -868 : VTAB VInvader% + 1 : HTAB
      1 : CALL -868
2420  VTAB VInvader% : HTAB HInvader% :
      PRINT INvader$ (1) : VTAB
      VInvader% + 1 : HTAB HInvader% :
      PRINT INvader$ ( FRhythm% + 1 ) ;
2430  FRhythm% = - FRhythm% : GOSUB
      1500 : GOTO 1000

**********INVADERS GO LEFT**********
2500  HInvader% = HInvader% - 1
2510  IF HInvader% < 2 THEN VTAB
      VInvader% : HTAB 1 : CALL -868 :
      ITravel% = RIght% : VInvader% =
      VInvader% + 1
2520  GOTO 2410

********** FILL VARIABLES **********
10000 SPEED= 255 : NOTRACE : NORMAL :
      TEXT : HOME
10100 NOise = -16336 : STrobe = -16368
      : KEy = -16384
10110 HInvader% = 3 : VInvader% = 4 :
      ITravel% = 1 : I$ = "9"
10120 RIght% = 1 : LEft% = 2 : TRavel%
      = 2 : FRhythm% =  -1
10130 POwer% = 9 : TTravel% = 2
10140 HGun% = 19 : VGun% = 17 :
      DIsplay% = 1
10150 BUnker$ = CHR$ (220) + CHR$ (239)
      + CHR$ (220) +  CHR$ (239)
10160 IVray% = 0 : IHray% = 0 :
      THray% = 2
10170 ZAp$ (0) = "*" : ZAp$ (1) = ":" :
      ZAp$ (2) = "~"

========== DATA STATEMENTS ==========
10500 DATA " *** ** **** **** ** *** "
10510 DATA "  ##   ##   ##   ##   ## "
10520 DATA "****    ** *** **   *****"
10530 DATA "####    ## ###    ###### "
10540 DATA "** **** *******   **   **"
10550 DATA "#######   ####    ###### "
10560 DATA "**   **   **** ** ** *** "
10570 DATA "#####   ##  ##  ##   ##  "
10580 DATA " *** ** ** *** ** ** *** "
10590 DATA " ### ## ## ####   ##   ##"

********** FILL STRINGS **********
10600 IF GO% then 11000
10800 ERr$ = "CHOOSE~A~NUMBER~FROM~1~
      TO~5"
10900 FOR A = 1 TO 40 : SPace$ = SPace$
      + "~" : NEXT A
10910 FOR A = 0 TO 9 : READ POwer$ (A)
      : NEXT A

*************** GRAPHICS ***************
11000 INVERSE : FOR A = 3 TO 24 :
      VTAB A : HTAB 1 : PRINT SPace$ ;
     : NEXT A
11010 VTAB 19 : HTAB 24 : PRINT
     "SCORE:"
11020 VTAB 19 : HTAB 2 : PRINT "TANK" :
      PRINT "VALUE" : HTAB 4 : PRINT
      "IS" : HTAB 3 : PRINT "-=>" ;
11030 VTAB 22 : HTAB 14 : PRINT
      "SOFTKEY~TEXT~INVADERS~2.0" ;
11050 NORMAL
11060 VTAB 24 : HTAB 2 : PRINT "<-LEFT"
      ; : HTAB 15 : PRINT "RETURN~=
      ~STOP" ; : HTAB 33 : PRINT
      "RIGHT->" ;
11070 VTAB 1 : HTAB 2 : PRINT "SPACE~=
      ~SHOOT!" ; : HTAB 20 : PRINT
      "ESC~=~END~GAME" ;
11100 POKE 32 , 1 : POKE 33 , 38 : POKE
      34 , 3 : POKE 35 , 16
11110 VTAB 3 : HTAB 1 : CALL -958
11200 FLASH : FOR A = 13 TO 15 : VTAB A
      : HTAB 4 : PRINT BUnker$ ; : HTAB
      13 : PRINT BUnker$ ; : HTAB 22 :
      PRINT BUnker$ ; : HTAB 31 : PRINT
      BUnker$ ; : NEXT A : NORMAL
11210 GOSUB 400
11220 GOSUB 210

********** INPUT SKILL CHOICE *********
12000 VTAB 7 : HTAB 7 : PRINT "WHAT~IS~
      YOUR~SKILL~LEVEL?" ; : VTAB 9 :
      HTAB 9 : PRINT "1....2....3....4.
      ...5" ; : VTAB 11 : HTAB 14 :
      PRINT "<~PICK~ONE~>" ;
12010 A = 5
12020 KEy% =  PEEK ( KEy ) : IF KEy% <
      127 THEN 12050
12030 IF KEy% < 177 OR KEy% > 181 THEN
      VTAB 2 : INVERSE : FOR B = 1 TO
      LEN ( ERr$ ) : HTAB B + 5 : PRINT
      MID$ ( ( ERr$ ) , B , 1 ) ; : Z =
      PEEK ( NOise ) : NEXT B : PRINT
      CHR$ (7) ; : POKE STrobe , 0 :
      HTAB 5 : PRINT RIGHT$ ( SPace$ ,
      LEN ( ERr$ ) + 1 ) : NORMAL :
      GOTO 12020
12040 SKill% = KEy% - 176 : GOTO 12100
12050 VTAB 9 : HTAB A * 5 + 4 : INVERSE
      : PRINT A ; : ZZ = PEEK ( NOise )
      - PEEK ( NOise ) : FOR C = 1 TO
      20 : NEXT C : NORMAL : HTAB A * 5
      + 4 : PRINT A ;
```

```
12060 A = A + 1 : IF A > 5 THEN A = 1
12070 GOTO 12020
12100 VTAB 7 : HTAB 1 : CALL -868 :
      VTAB 9 : CALL -868 : VTAB 11 :
      CALL -868:
12200 GOSUB 13000 : VTAB VInvader% :
      HTAB HInvader% : PRINT INvader$
      (I)
12210 GOSUB 400
12900 GOTO 1000
12999 END

========== CREATE INVADER $ ==========
13000 IF SKill% < 6 THEN SKill% =
      SKill% + 1
13010 FOR A = 0 TO 2 : INvader$ (A) =
      "~~" : NEXT A
13020 N% = SKill% + 2 : FOR A = 1 TO N%
      : INvader$ (1) = INvader$ (1) +
      "]" + I$ +  CHR$ (91) + "~"
13030 INvader$ (0) = INvader$ (0) +
      "/" + CHR$ (220) + "~" : INvader$
      (2) = INvader$(2) + "(~)"
13040 NEXT A : RETURN

============ TANK DESTROYED ==========
13200 INVERSE : GOSUB 220 : GOSUB 13290
      : VTAB IVray% : HTAB IHray% :
      PRINT "." ; : GOSUB 13290 :
      NORMAL : GOSUB 220
13210 VTAB VGun% : HTAB HGun% - 2 :
      PRINT "=:#:=" ; : GOSUB 13290 :
      VTAB VGun% - 1 : HTAB HGun% :
      PRINT "^" ; : GOSUB 13290
13220 VTAB VGun% : HTAB HGun% - 2 :
      PRINT "- * -" ; : GOSUB 13290 :
      VTAB VGun% - 1 : HTAB HGun% :
      PRINT "." ; : GOSUB 13290
13230 VTAB VGun% : HTAB HGun% - 2 :
      PRINT "~~-~~" ; : GOSUB 13290 :
      VTAB VGun% - 1 : HTAB HGun% :
      PRINT "~" ; : GOSUB 13290
13240 VTAB VGun% : HTAB HGun% - 2 :
      PRINT "~~~~~" ; : GOSUB 13290
13250 FOR E = 1 TO 100 : NEXT E
13260 VGun% = 13 : GOSUB 400 : RETURN
13290 FOR E = 1 TO 13 : BUzz = PEEK
      ( NOise ) : NEXT E : RETURN

==========BEGIN ALL OVER AGAIN=========
20000 AGain% = AGain% + 1
20010 VTAB VInvader% : HTAB 1 : CALL
      -868 : VTAB VInvader% + 1 : HTAB
      1 : CALL -868
20020 VInvader% = SKill% + AGAIN% :
      HInvader% = 3 : ITravel% = 1
20030 GOSUB 13000 : GOTO 1000

==========OPTION TO PLAY AGAIN==========
21000 HOME : VTAB 5 : PRINT "YOUR~FINAL
      ~SKILL~LEVEL~WAS " SKill% ;
21010 VTAB 13 : HTAB 5 : PRINT "YOUR~
      SCORE WAS " PTs% ;
21070 VTAB 13 : HTAB 5: PRINT "DO~YOU~
      WISH~TO~TRY~AGAIN?~(Y/N)" ; : GET
      ANswer$ : IF ANswer$ = "Y" THEN
      GO% = 1 : GOTO 10000
21080 IF ANswer$ = "N" THEN TEXT : HOME
21090 GOTO 21070

************End Of Listing ************
```

NOTES & NOTICES

Our premier issue had no date on it (sort of like Nibble...). However, for those who must have dates... These are the dates you can expect the next issues and updates to be out:

```
Issue #2       -- October
  Update 2.1   -- November
  Update 2.2   -- December
Issue #3       -- January '82
  Update 3.1   -- February
  Update 3.2   -- March
Issue #4       -- April
```

We hope that by then we will be able to go bimonthly (and later, go monthly) but that will depend on how many subscribers we have and how many articles and programs we receive. So keep on recruiting hard-core computerists...

Good news for those who complained that our first issue was a bit thin. Issue #2 will have about twice the pages.

If you wrote letters to us asking questions... those letters will probably be answered in HARDCORE 2.

Also...watch for a review of a bit-copy program that we missed last time. It's called "NIBBLES AWAY" and we will be reviewing it in HARDCORE 2.

The SoftKeys in the first issue were not aimed only at people with Apple IIs. For you Apple II plus people who don't have an integer card, you can still use the SoftKeys.... details in the upcoming issue.

Creative Computing Censors Hardcore Advertisement

Creative Computing is still censoring ads and suppressing information about copy protection. It has just refused to publish the HARDCORE advertisement shown on this page.

What did the editors of CC find unprintable in our ad?

"Only two lines," explained their Advertising Manager, Jerry Thompson. "They are: 'back up any diskette', and 'do & undo copy-protection'."

Jerry said that either we agree to let them remove those lines or they would refuse to run the ad. Well, we stood by our editorial policy of fighting censorship and they stood by their policy of information suppression. So you won't see this ad in CC.

One of their editors, George Blank, defended CC's policy by offering the lame comparison to Reader's Digest's policy against running liquor ads. He added that at least CC will run ads for competitors.

Big Deal! All of them share CC's censorship policy. All of them... except HARDCORE COMPUTING. And they will not run our ads.

George's final defense was that CC had a readership of well over 100,000... so obviously they must be doing things right. What can I say to that?

And when I mentioned that the readers have had no alternative, he answered promptly, "Yes, they do!" (meaning his competitors). I retorted that they practice the same form of censorship.

"That might mean that we're right and you're wrong," he concluded.

So... The battle goes on.

CC is probably just the first computer magazine to censor HARDCORE's ads. We plan to try to put one in Call A.P.P.L.E., but unless Val Golding, the editor, has altered his policy, they will probably refuse to run our ad, too.

We'll keep you posted on further developments.

Attention Apple Clubs

Attention all Apple-users groups. We want to know more about your club. Please send us a letter telling us about your membership, goals, constitution and bylaws, founding date, dues, benefits of membership and newsletters, etc.

Address it to: A.U.G. SURVEY in care of this magazine.

DiskEdit updated, but late

All those of you who have ordered DiskEdit 1.0 will receive the updated version 2.0.

We have back-ordered the disks and they should be arriving in a week or so, so please be patient.

Yes, We're Copyable!

Material in HARDCORE (Computing, Updates, and Alerts) may be reprinted without permission by school and college publications. Apple-users groups may also reprint this material if they run (in the same issue) the AD shown on this page along with this notice:

COPYRIGHT c 1981
SoftKey Publishing

The AD may be larger than shown, but not smaller.

The reason for this is simple: We need more subscribers. And somebody has got to run our ads... We worked so hard on them...

Don't forget to send us a couple of copies of that issue. In fact, we'd be pleased if those clubs would add us to their mailing list so that we can keep up with your doings... OK?

Hardcore Worms

Try as we might to be perfect, we still made a few errors:

The most important worm is on page 28, DEMUFFINS... We left out the "less than" sign on DEMUFFIN PLUS. Steps 5b and 5c should read:

b. 1900<B800.BFFF ctrl Y* return
c. 1900<B800.BA10 ctrl Y return

The next worm is on page 29, $HEX-DECIMAL CONVERSION CHART. In the second column, the bottom row reads: $12-28. It should read $12-18. It's obvious from the way the numbers increment... but it slipped right by us.

Now on to page 31, DISK LOCKS, the second column (inside the artwork), the second paragraph (dealing with using MUFFIN to upload programs to your protected disks from a 13-sector disk). Well, step 2 says to BLOAD DEMUFFIN. It should say: BLOAD MUFFIN.

That's all we've caught so far.

Send in your nominations for the Garbage Award

Ever spend good money for a program that turned out to be just garbage, a program that was filled with worms (or Med-flies)? Want to air your complaints?

We are taking nominations for HARDCORE's lowest award: the soon-to-be-infamous GARBAGE AWARD!

Send in your list of the worst programs that you have purchased and explain why each is deserving of this highly-dishonored token of our contempt. Rank them according to their True Garbage Value (ie. #1 is your choice for Garbage Award Giveaway (G.A.G.) First Place).

Get them in soon so that we can let the world know who has garnered this prized trophy. We'll also have a lonnnnnnng list of Dishonorable Unmentions, so hurry.

Clip-out HARDCORE Subscription Postcards

YES! I want to subscribe to
HARDCORE computing
the apple Free-Press!

Start with issue #_____.

( ) Bill me $20 for a
1-year subscription.*

( ) Enclosed is $20 for a
1-year subscription.*

Mail subscription to:

Name:_____

Address:_____

City:_____

State:_____ Zip:_____

*U.S. rates only. Write for foreign rates.

YES! I want to subscribe to
HARDCORE computing
the apple Free-Press!

Start with issue #_____

( ) Bill me $20 for a
1-year subscription.*

( ) Enclosed is $20 for a
1-year subscription.*
Send the first issue(s) now!

Mail subscription to:

Name:_____

Address:_____

City:_____

State:_____ Zip:_____

*U.S. rates only. Write for foreign rates.

YES! I want to subscribe to
HARDCORE computing
the apple Free=Press!

Start with issue #_____.

( ) Bill me $20 for a
1-year subscription.*

( ) Enclosed is $20 for a
1-year subscription.*
Send the first issue(s) now!

Mail subscription to:

Name:_____

Address:_____

City:_____

State:_____ Zip:_____

U.S. rates only. Write for foreign rates.

YES! I want to subscribe to
HARDCORE computing
the apple Free-Press!

Start with issue #_____.

( ) Bill me $20 for a
1-year subscription.*

( ) Enclosed is $20 for a
1-year subscription.*
Send the first issue(s) now!

Mail subscription to:

Name;_____

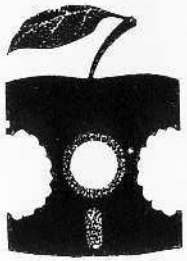
Address:_____

City:_____

State:_____ Zip:_____

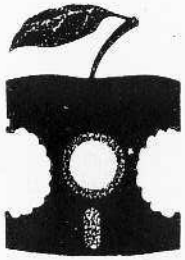
U.S. rates only. Write for foreign rates.

HARDCORE computing
P.O. Box 44549
Tacoma, WA 98444
(206) 531-5690



HARDCORE computing
P.O. Box 44549
Tacoma, WA 98444
(206) 531-5690

HARDCORE computing
P.O. Box 44549
Tacoma, WA 98444
(206) 531-5690



HARDCORE computing
P.O. Box 44549
Tacoma, WA 98444
(206) 531-5690