**BASIC**

**COMPUTER**

**GAMES**

MICROCOMPUTER EDITION

101 Great Games to Play on Your Home Computer.

By yourself or with others. Each complete with

Programming and sample run. Edited by David H. Ahl

**BASIC**

**COMPUTER**

**GAMES**

MICROCOMPUTER EDITION

**BASIC**

**COMPUTER**

**GAMES**

MICROCOMPUTER EDITION

Edited by David H. Ahl

Program Conversion by Steve North

Illustrations by George Beker

All of the programs listed here run without error, in Microsoft Basic Version 3.0 or higher. While most users will encounter no problems in entering and running te games, some microcomputer Basics may require program conversion. If you are a newcomer o personal computing , do not attempt to enter the longest program first. This will only result in frustration and confusion. You must become familiar with your Basic’s capabilities and limitations before attempting one of the longer programs.

Copyright © 1978 by Creative Computing

All rights reserved. No portion of this book may be reproduced – mechanically, electronically or by any other means, including photocopying—without written permission of the publisher. Published simultaneously in Canada by Saunders of Toronto, Inc.

Trade edition published in cooperation with Creative Computing.

Library of Congress Cataloging in Publication Data Main Entry under title:

Basic computer games.

Includes index.  
 Games—Data processing.  
Microcomputers—Programming.

1. Ahl, David H.
2. North, Steve.

GV1469.2.052 1978B. 794 78-17624  
ISBN: 0-89480-052-3

Cover Design: Paul Hanson  
Illustrations: George Beker

Workman Publishing  
1 West 39 Street  
New York, New York

Manufactured in the United States of America  
First printing October 1978

20 19 18 17 16 15

**ACKNOWLEDGEMENTS**

With a book like this one, it’s difficult o know where to begin with acknowledgements.

**Game Authors.** Thank you, each and every one for creating your game in the first place.

**Steve North**. For working way beyond the call of duty transcribing, modifying, converting, and debugging all the games in MITS BASIC. A truly herculean task!

**John Lees**. For assisting Steve North in the conversion effort.

**Jim West**. For listing and playing all the “standard” BASIC games from my timesharing accounts.

**Teletype Corporation**. For the loan of the 43 Teleprinter to do the listings and runs.

**Bob Albrecht**. For his continuing contribution to the world of computer games and for setting me straight as to who wrote what.

**Dartmouth College**. For recognizing games as a legitimate educational tool and allowing them to be written and played on the Dartmouth timesharing system.

**Microcomputer Manufacturers**. For putting computer games within the reach of every American in the comfort of their own home.

**Contents**

*{Reserved for Table of Contents, TBD if auto-generated by word processor or manually typed/recorded from original text}*

**Introduction**

You’re seated in a heavily padded chair which is vibrating ever so slightly. All of a sudden you feel a cold and your view screen, which has previously been black, shows the receding form of the lunar command module. It will continue to orbit around the moon two hundred miles up while you land your LEM and explore the moon. Your viewscreen now shows that you are beginning to tumble, so you hit the button on your right joystick to give a short burst on the starboard stabilization rockets. A few more short bursts right, left, then right again and your viewscreen shows that you’ve stabilized your craft and that you’re headed smoothly, if rather rapidly, toward the lunar surface. In panic you realize that you’ve taken a long time with your stabilizing maneuvers so you jab the retro rockets firing control on full. Your viewscreen is now obscured by the flames so you switch to a computer display which shows your spacecraft and it’s position as you approach the moon. You manage to slow your crat however, you see that you’re heading for the side of a crater and you don’t have enough time for positional maneuvers. You switch your viewscreen back on and hope for the best. Unfortunately, the crater walls ae steep and one leg of your LEM mk4es contact before the others. It buckles and the craft topples over. At this, Mission Control in Houston sends you a radio message, “Ground telemetry records severe craft damage. You have 13.2 days of oxygen. Information on rescue attempt to follow. Don’t panic.”

You punch another button on your console and your screen lights up, “Lunar Landing Simulation complete. Try again?” You lean back in your easy chair, palms moist with perspiration and type, “No. Get Football.” A moment later a referee and the hulking shape of Roger Staubach appears on the screen. The ref turns and asks you, “Heads or tails?”

Far-fetched? Not at all. Technologically this is all possible today. From an economic standpoint it will take a few more years before systems with these capabilities are within reach of the average consumer. But even today some amazing games are possible with the current breed of home computers. That’s what this book is all about, games for home computers, minis, timesharing systems and even large mainframes—in short, any computer that speaks Basic.

Computer games are not a new phenomenon. Back in 1952 shortly after the first commercial computers were introduced, A. L. Samuel at IBM wrote a checkers program for the IBM 701. It was written with the idea that a great deal could be learned about the human though process if one could simulate it on a computer. This also was the reason that Newell Shaw and Simon a few years later at Rand Corporation wrote the first computer chess program. But even to those uninitiated in the field of artificial intelligence research, these programs were great fun as games even if they didn’t play outstanding chess or checkers.

But while these programs were being written as part of research projects, a much larger group of people were furtively writing and playing games at lunchtime and before and after work on their employers computers. There were at least two or three of these fanatical game players, sometimes more, at each computer installation of any size. The advent of the minicomputer and timesharing networks in the early 1960s expanded this community of computer hackers and by 1966 they were meeting at various professional society meetings and laying out plans for a computer chess tournament.

The hard core of the hackers, the real cultists, were those that were into Spacewar. Originally written by some hackers at the MIT EE Department back in 1961-62 for a DEC PDP-1, the first minicomputer, Spacewar spawned a fanatical community of hackers who played, modified, improved and experimented with it.

“Ah, Spacewar, Reliably at any nighttime moment, hundreds of computer technicians are effectively out of their bodies, computer-projected onto CRT display screens, locked in life-or-death space combat for hours at a time, ruining their eyes, numbing their fingers in frenzied mashing of control buttons, joyously slaying their friends and wasting their employers’ valuable computer time. Something basic is going on.” (Stewart Brand in *// Cybernetic Frontiers*.)”

Item: October 1, 1972. The PDP-10 at the Stanford AI Laboratory is reserved from 8:00 pm. on for the “Intergalactic Spacewar Olympics.”

Item: October 1976. Cromemco announces Spacewar for the 8080 and TV Dazzler. Paper tape $15.00.

In layman’s terms what those two items mean is that in the short span of four years Spacewar went from a game that required the use of a mult-million dollar computer to a game that could be played on a $1,000 home computer.

What happens to a fanatical cult when you open the temple doors and let everyone take its source into their own homes? Obviously, we don’t know since the temple hasn’t been open that long, but it seems obvious that this same generation of kids that can’t do manual math or use a slide rule because of the pocket calculator may learn that a TV set can throw some actively challenging things their way instead of just a passive picture.

***About this Book***

**Basic Computer Games: Microcomputer Edition** is a major revision of my first book, 101 Basic Computer Games published in 1973 by Digital Equipment Corporation. The programs in the original book represented six different versions of Basic. The programs in this book all run in Microsoft Basic (more about that later on page XII). There were several programs that couldn’t be converted or weren’t worth converting to Microsoft Basic. These were deleted and replaced with new programs.

Program listings were done on a Teletype Model 43 terminal. Its 7 x 9 dot matrix printer coupled with its 13.2 characters per inch spacing gives it exceptionally high legibility, even when reduced to 64% as it has been done in here. I salute those of you with the first book who patiently, magnifying glass in hand, deciphered nearly illegible listings and got those programs up and running. This book should make that task considerably easier.

David H. Ahl  
September, 1978

**The Basic Language**

Be sure to read this section before entering any of the games in this book on your computer. It will save you time and minimize potential problems.

The games in this book were written by a wide cross section of people on a variety of computers over a ten year period. Most of the games were originally written for time-sharing systems such as the DEC Timeshared 8 and RSTS-11, HP 2000, and CDC 3600.

The first edition of this book was originally published in the early 70’s. All the programs were printed in their original dialect of Basic. One of the major functions of the book was to give computer users an opportunity to learn more about their own and other versions of Basic. For example, a dimensioned string variable such as A$(I) had an entirely different meaning in DEC and HP Basics. The designers of each version of Basic had good reasons for doing what they did and it was (and is) very instructive to understand how different approaches work and their respective rationales.

In 1975, a small company in the telemetry business, M.I.T.S., introduced the first computer for hobbyists, the Altair 8800. This signaled the start of an immense new industry, personal computers. Ed Roberts, president of M.I.T.S., contracted with a small consulting company started by two bright young programmers to write a version of Basic for the Altair.

The name of the consulting company was Microsoft, and the Basic that Bill Gates and Paul Allen devised soon came to be known, appropriately enough, as Microsoft Basic. It was modeled on Basic-Plus, originally a creation of Nathan Teichholtz at Digital Equipment Corporation. Nathan is an unsung hero in the history of the Basic language and deserves a great deal of credit for this vastly improved version of the language. And in the kudos department, we mush always remember to bow low to John Kemeny and Tom Kurtz of Dartmouth, the creators of the original Basic.

In any event, in 1977 it appeared that Microsoft Basic was fast becoming the standard Basic for microcomputers, and the programs in this book were all converted to Microsoft Basic, Revision 4.0. For about three years, this Basic truly was the standard.

However, three things conspired against it becoming the all-time standard. First, it was written for the 8080 (and Z80) microprocessor, and later mpu’s had capabilities (and idiosyncrasies) that the 8080 did not have, hence, slight differences started creeping into Microsoft Basic on later computers.

Second, not all computer manufacturers wanted to contract with Microsoft to write Basic for their computers and so some wrote their own. This has led to some particularly interesting (and confusing) dialects of Basic.

Third, as computer hardware became more powerful with 16-bit mpu’s, special chips for graphics and sound. Winchester disk drives, modems, and other peripherals, various extensions and changes had to be made to the software. Some of these changes are in the operating system and are transparent to the user, but others affect the Basic language.

So, what you are holding is a book of thoroughly debugged programs that can be entered directly and will run perfectly on some computers, while on other computers they will not run at all. What you can do about it if you have a computer on which the programs will not run?

First, do not call or write us. You have paid less than a dime per program and, after everyone has taken his share, we have made less than 1 cent per program, for that, we can’t afford to act as personal consultants.

Second, please keep in mind that every program runs perfectly in Microsoft Basic 4.o. The sample runs are not faked; they came off a real computer exactly as they appear. There are no typographical errors in the programs—misspellings maybe—but the functional code is absolutely correct.

Third, the early versions of the books had programs in 15 or 20 vastly different dialects of Basic, yet hundreds of thousands of purchasers managed to convert the programs to their machines and get them going.

The key to converting the programs is to understand how Microsoft Basic works compared with your Basic. While we cannot present an entire manual on Microsoft Basic, we have included in the next section information on the key elements of the language and those likely to be different in other implementations.

If direct conversion doesn’t work, do some “reverse engineering.” (as it known in the computer industry). This means taking apart a program and drawing a flowchart of the logic. With a flowchart in hand, you can then write your own program to do the same thing, but often faster and better.

Since many of the programs were originally written in what today would e considered a primitive version of Basic, there are many ways of substantially shortening and improving some of these programs.

You should also keep in mind that all of these programs were written on computers which used an ASR-33 Teletype terminal. These are massive clunkers with 72 (or80) columns of output, upper case only, and no graphics capability whatsoever. Naturally, you will have to do a fair amount of reformatting if your computer has only 40 columns (Apple, etc.), 32 columns (Sinclair), 28 columns (TI), or 20 columns (Epson HX-20). It can be done; for another book, I converted Hammurabi, Lunar Lander, and Gunner, all of which use 72 columns, to all the computers mentioned above.

*{This page modified from the original publication for easier entry. Page may be modified at a later date to reflect original content format.}*

The programs in tis book use the following statements and functions in Microsoft Basic:

Statements

Definition

Sample

DATA

Holds numeric or string data for a READ statement  
Defines any function of X

20 DATA 4,8,”AHL”

DEF FNA (X)

Defines any function of X

20 DEF FNA(X)=3\*X-2  
20 DEF FNA(X)=SIN(X/57.3)

DIM

Declares maximum size of string or numeric array. Array subscripting begins at 0 although many programs do not use the zero subscript.

20 DIM A(50)  
20 DIM A$(25),B1$(50)

END

Last statement in program

9999 END

FOR TO (STEP)

Executes a loop. The test for ending the loop is made after the loop has been executed. Upon exiting, the counter value equals the upper limit plus the step. For example,

10 FOR J=1 TO 3  
20 PRINT “HI”  
30 NEXT J

will print “HI” three times, and J will equal 4 when the loop is finished.

20 FOR I=1 TO 30  
20 FOR J=22 TO N STEP 3

GOSUB n

Branch to subroutine n.

20 GOSUB 200

GOTO n

Branch to statement n.

20 GOTO 50

IF…THEN n

Branch to statement n if condition is true.

20 IF A>1 THEN 50

IF…THEN stmts

Executes statements if condition is true. Drops to next numbered line if false.

20 IF Z<5 THEN A=1:PRINT B

IF…THEN n ELSE m

Branch to n if true or to m if not true.

20 IF X=Y THEN 50 ELSE 90

IF…THEN stmts  
ELSE stmts

Does statements after THEN if true, stmts after ELSE if false.

20 IF Z>R THEN X=1 ELSE X=2

INPUT

Requests data from keyboard. The prompt string is optional.

20 INPUT N  
 20 INPUT “YES OR NO”:Z$

LET

Assigns value of expression to variable. The word LET is optional.

20 LET A=1

NEXT

Marks end of FOR loop.

20 NEXT J

ON m GOSUB

Branch to mth subroutine.

20 ON X GOSUB 100,200

ON m GOTO

Branch to mth line no.. In these statements, m must be an integer starting at 1 and increasing by 1.

20 ON Y GOTO 50,80,120

PRINT

Displays strings, constants, and variables. Calculations can be done within a PRINT statement.

20 PRINT “A=”;A  
 20 PRINT Z$;10\*A+B

READ

Moves values of DATA into variables.

20 READ N;X1;A$

REM

Remark. Does not execute.

20 REMARKABLE PROGRAM

RESTORE

Resets DATA pointer to first item in list.

20 RESTORE

RETURN

Go to statement following last GOSUB.

20 RETURN

STOP

Terminates program.

20 STOP

*Functions*

ABS (X)

Absolute value.

ASC(X$)

Returns the ASCII value of the first character in the string argument. ASC(  
“A”) is 65, ASC(“B”) is 66, etc.

ATN(X)

Arctangent.

CHR$(X)

Converts ASCII value to a character string. CHR$(65) is “A”, etc.. CHR$(7) is a bell ring.

COS(X)

Cosine.

EXP(X)

Value of e raised to the X power.

INT(X)

Integer function.

LEFT$(X$,Y)

The left most Y characters in X$.

LEN(X$)

Number of characters in X$.

LOG(X)

Log of x to the base of e.

MID$(X$,Y,Z)

Takes Z characters from X$ starting at position Y.

RND(1)

Returns a random number between 0 and 1.

RIGHT$(X$,Y)

The rightmost Y characters in X$.

SGN(X)

Sign function. Returns -1 if X is negative, 0 if x is zero, 1 if x is positive.

SIN(X)

Sine.

SQR(X)

Square root.

STR$(X)

Converts X to a string of decimal digits, for example if X was 8.45 it would be converted to the string ”8.45”.

TAB(X)

Spaces over to position X on the terminal.

TAN(X)

Tangent.

VAL(X$)

Returns the numerical value of the string of digits in X$. Opposite of STR$(X).

In Microsoft (Altair) BASIC, any expression may be evaluated as either true or false. A true condition will return a value of -1, and a false condition 0. Thus if we say LET Q=-(X=Y), Q=1 if X=Y and Q=0 if X=Y. Tis logical evaluation of expressions is only used in the Hexaspawn game in the user-defined function and with a little ingenuity could be replaced with a look-up table. A few other games use the logical AND and OR operators, which work in a straightforward manner.

The programs in this book were printed on a printer that uses a caret (^) to indicate exponentiation. This is equivalent to an up arrow. Incidentally, exponentiation and taking roots are among the least accurate functions on small computers. For example, try this program.

10 INPUT N  
20 I=SQR(SQR(N))  
30 J=(I12)12  
40 PRINT N,J  
50 GOTO 10

Chances are good that N and J will not be the same 25% or more of the time. You can improve the accuracy by substituting J\*J for J squared or J\*J\*J for J cubed.

Microsoft Basic permits more than one statement on a line when the statements are separated by a colon(“). As noted above, in an IF…THEN statement, if the condition is false, control drops to the next number line, not to the next statement on the same line.

This means that for TI and other computers that do not permit multiple statements on one line, you will have to insert additional lines. This may be difficult when line numbers are close together. One easy solution is to add a zero to all line numbers, but you must remember to do that in IF…THEN, GOTO and GOSUB statements as well as changing all the numbered lines.

The random function can be especially irksome as it is one that differs widely on different versions of Basic. In Microsoft Basic, RND(1) returns a value between 0 and 0.999999. This is the convention used in all programs in this book. On some computers, you may have to use RND(0), and on others just RND.

**Conversion to other Basics**

**Radio Shack BASIC**: Approximately 25 of the games which do not use strings will run under Level 1 BASIC. Another 20 or so may be converted relatively easily by converting to all numeric input (1 instead of YES, 0 for NO. etc.) In Level 2, change RND (1) to RND (0). Also function definitions, when used, must be expanded.

**Apple II BASIC**: All programs will run directly in Applesoft Basic with little or no modification. In Integer Basic, strings are similar to HF Basic. Also, multiple statements on one line work differently than with Microsoft Basic.

**DEC BASIC PLUS**: Programs written in Altair BASIC should be completely compatible with DEC BASIC PLUS.

**HP BASIC**: Some conversion will be required to get certain programs to run in HP BASIC. First, HP BASIC does not have a zero subscript for arrays. So if, in a program, the zero subscript is used, HP BASIC will report a subscript out of bounds error. The solution is to add one to all the array subscripts if it is determined that the program does use the zero subscript.

Second, HP BASIC character strings differ from Altair BASIC. There are no string arrays. So if string arrays are used in some program, a conversion must be made. Possibly the string array could be converted into a single string, or a numeric array, or a series of data statements, or possibly a data file. Also, HP BASIC does not have RIGHT$, LEFT$ or MID$ functions. Instead substitute the normal HP string subscripting conventions [for instance, RIGHT$(X$,2) becomes X$(LEN(X$)-1, LEN(X$)].

Third, HP BASIC does not have multiple statements on one line. Convert multiple statement lines to several separate lines.

**SWTPC 6800 BASIC**: Versions 2.0 and above of this BASIC should be nearly compatible. Like HP BASIC, SWTPC BASIC does not allow the zero subscript. Also, character strings may not contain more than 32 characters. Note that in Altair BASIC, RND(1) is used to get random numbers, rather than RND(0) as in SWTPC BASIC. These comments also apply to MSI Disk BASIC.

**TDL ZAPPLE 8K and 12K BASIC**: All the programs are executable in both TDL BASICs. However TDL ZAPPLE BASIC has a RANDOMIZE verb which should be inserted at the beginning of a program to get true pseudorandom numbers.

**Sol BASIC**: The programs will work directly in SOL Extended Basic. Many programs will run in 5K Basic but the majority will require considerable conversion.

**PET BASIC**: Commodore PETW BASIC is identical to the BASIC used in this book. Thus, all the games will run without any conversion.

**Cromemco 16K Extended BASIC**: Cromemco BASIC allows multiple statements, but certain statements must be the last statement on a line, so some rearrangement may be necessary. Two-dimensional arrays must be dimensioned explicitly (there is no default to a 10 by 10 dimension). Also, strings in Cromemco BASIC are implemented like HP BASIC, not Altair BASIC.

**Ohio Scientific BASIC**: Most programs will run directly on Challenger 2P and larger systems. The 32-characters per line display of the Challenger 1P and superboard will require conversion o the program output routines.

**IMSAI 8K BASIC**: Programs are executable with little or no modification.

**North Star Disk BASIC**: North Star BASIC has character strings like HP BASIC, so some conversion will be required. Also note that the function SQRT is used in place of SQR.

**PolyMorphic 11K BASIC**: The multiple statement separator is a backslash (/), not the colon. Like North Star BASIC, SQRT is used instead of SQR. In Poly BASIC you’re not allowed to exit a FOR loop prematurely, except by using the EXIT verb. So whenever you see an IF…THEN out of a FOR loop, use the EXIT verb. Also note that in order to get random numbers, RND must be used with a “see” value between 0 and 1, which determines at what point in the sequence of random numbers the RND function starts. The see value is usually based on the time of day. After initializing the RMD function, use RND(0) to get random numbers.

**BASIC-E**: (runs under CP/M Disk Operating System\_. All arrays must be dimensioned (there is no default). The RANDOMIZE verb should be used a the beginning of the program, to get random numbers. Also, multiple statements are not allowed, so conversion to several single statement lines will be necessary.