smartBASIC* BONANZA

15 PROGRAMS FOR THE ADAM*

*T.M. COLECO Ind. Inc.

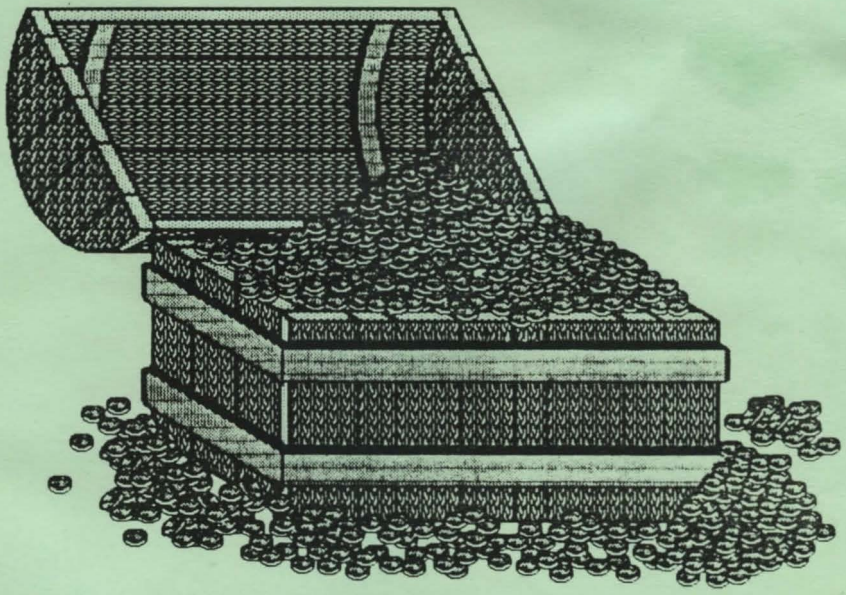

TABLE OF CONTENTS

Welcome to Bonanza

The 15 SmartBASIC Bonanza programs are all written in BASIC, so you must first load in BASIC and then use the load command from BASIC to use these programs. Many of these programs use nearly all of memory, and ADAM sometimes does odd things if a second large program is loaded in right after some other large program. It is a good idea to turn off the machine and load BASIC before using the next program.

THE MANSION

The Mansion is an adventure game, or in other words is a text game that transports you to another place and challenges you to find treasures and solve problems. You find yourself at the doorstep of an ancient mansion and you must get in and out with various treasures, but you must also remain alive. You are not alone, though, you travel with a helpful companion who acts as your eyes and who you instruct to do things. You give it commands in two word sentences such as, *go north*. It has a vocabulary of about 100 words, and if the words are used creatively it will, in most cases, keep you out of trouble.

Here are some helpful hints for those of you who haven't played adventure games before.

-try synonyms if your friend doesn't understand you
-look at everything
-read everything
-get all carryable items
-remember getting treasures requires tricky thinking.
-if you're really having trouble call for help
-check inventory

Here's a hint that might get you into the mansion: "GOPHER IT"

Everything else you need to know must be discovered as you travel through the mansion. Have fun.

MAGIC

This program actually consists of two separate illusions that will impress your friends and help answer the age-old question, "What good is this thing?" We have organized the program so that you can save them as separate programs if you wish. The first illusion, MENTAL, is between line numbers 1000 and 2000. The second, ANALYSIS, is between 2000 and 3000.

MENTAL creates the impression that your ADAM can actually respond intelligently to spoken and written questions and even to read minds. You need to be able to touch type, at least to some extent, and you will have to practice a bit before your first demonstration. (If you don't touch type, use the TYPER program to learn--it is something that will become increasingly necessary as computers proliferate.)

When you start MENTAL, the screen will clear and say, "So ask." It will appear that you type in a question like, "Who is nearest the door?" Then ADAM answers with the correct information! In fact you typed a capital D and the answer to the question, but like any good magician you stared at the screen as you typed, in order to distract your audience. The capital D told ADAM to start typing the "door" question, and it types one letter of the question for each letter you are typing in with the answer.

When you type in a period, ADAM stops listening to the answer, but you should keep typing random letters until ADAM has printed the whole question on the screen. When a period or question mark appears on the screen, stop typing. As soon as you press "return" ADAM will magically answer the question.

There are seven specific questions built into MENTAL (plus seven general questions, which we will get to in a minute).

Code letter.	Question
W	What is the weather like out?
L	Who has a secret love?
T	What am I thinking about now?
F	What does the future hold for us?
D	Who is nearest the door now?
M	How can I get some money quickly?
G	Which team won the last big game?

As you can see, with a little imagination and practice, you

will be able to stun your friends.

There are also seven general questions or statements that you can use to get ADAM to respond to oral questions. If you first type R (for random), ADAM will type some general phrase like, "Here's another question." Of course, this is what is appearing on the screen while you type in the answer to whatever question you plan to ask. Again, when you type in a period, ADAM will assume you are at the end of the answer, but will continue to type in the general phrase as you hit random keys until a period or question mark appears. Then, just before you press "return", ask your question orally. ADAM hears and answers!

As with any illusion, it works best if it is only done a few times. People not familiar with computers can be fooled by this for a long time, but eventually ADAM starts repeating himself and this looks suspicious. One way to get out of such a situation is to first type the code letter Q. ADAM will say, "STOP! I am sick and tired of these silly questions. I QUIT!" You then apologize for ADAM's testiness and quit.

ANALYSIS is a completely different kind of illusion. It is based on what psychologists call "The Barnum Effect." In one experiment, a large group of people were all given the same "Personality Profile" that was supposedly based on careful testing. Eighty-nine percent of the people said that the profile was a "good" or "excellent" description of themselves--even though everybody had exactly the same profile!

This principle is used in such things as astrology columns, personal readings, and those "computer handwriting analysis" booths that one often sees in shopping malls. ADAM will now demonstrate the same level of deep insight and probably do a better job.

The program will ask for some personal information such as birthdate, color preferences, etc. Then there will be a brief pause for "analysis time", although we had to slow ADAM down at this point by wasting time in the program. If he came up with your own personal analysis too quickly, you would suspect he hadn't thought very carefully about it--and of course you would be right. You can print the "analysis" on the screen or the printer (to show to your shrink, of course).

THIS PROGRAM MUST NOT BE TAKEN SERIOUSLY!

It is a complete scam, a con, a cheat. The problem is that the phrases used in the analysis are cleverly written to be

statements that are true of nearly everybody but appear to be highly personal statements. Many people are fooled by "The Barnum Effect" and spend lots of wasted money because of it. ADAM really pays almost no attention to the "personal information" entered, except to personalize some of the statements. The program randomly selects and organizes each "profile" from a pool of statements.

Obviously, you cannot pull off this illusion very often with the same person at the same time. Even if a person enters identical personal information (but a different "favorite number")a second time, he or she will get a different "profile"--slightly suspicious. (Each "favorite number" will get a different set of randomly chosen "personality description" statements.) However, a clever operator could just refer to ADAM's "further insights" or "unconscious changes" in the subject or some other such nonsense.

Be sure to tell your audience the truth about this program at the end.

OTHELLO

Now ADAM does some thinking (OK, almost thinking). The game of Othello is played on an 8X8 board, and you are pitted against ADAM, who is no genius but plays fairly well, or against another human. The board starts out with two white markers and two red markers in the center of the board. ADAM will ask which you want to play. White goes first. To play, you must place a new marker on the board so that it traps at least one of the opponent's markers between you new marker and one of your markers already on the board. When you do this, all of the trapped markers "flip over" and become your color. You can trap markers along the horizontal, vertical and diagonal axes of the board. Then your opponent tries to trap at least one of your markers. If one of you cannot trap an opponent's marker(s), you must pass, and the other person makes a play. The game ends when the board is full or neither of you can make a legal move.

ADAM will check the legality of moves, ask you to determine a level of difficulty,and count up the score at the end. There is considerable strategy involved in Othello. Capturing corners should be your highest priority. Good strategy is to place markers in the outer squares. It is dangerous to place markers in rows and columns adjacent to the outer squares, but this is often unavoidable.

TYPER

This program can serve as a general typing instructor and as a way to sharpen up your typing accuracy. The opening "menu" lets you choose the lesson of interest or a skill sharpening game. The instructional material follows the general format of most typing courses, starting with "home row" etc. The lessons are arranged in order of level of difficulty. In general, TYPER is self-explanatory. The advantage ADAM gives you, of course, is that it can present both individualized instruction and personal drill materials at your own speed of development.

ASSEMBLER

This program and DISASSEM are both for more advanced users. They will be useful to those familiar with assembly language. SmartBASIC is a wonderfully easy computer language to use, but in order to make it such a "high level" language, it contains such things as checking procedures which slow it down.
There are also some things that the SmartBASIC that came with your ADAM is not designed to do, such as create sounds on the ADAM. Assembly language is essentially the "native tongue" of the computer and produces programs that run extremely fast and permit you to do anything the computer is capable of. The problem is that it is very complicated. The ASSEMBLER translates your assembly language program into "machine language" which is the primitive pattern of ones and zeroes that the machine understands.

If you want to learn assembly language programming, you should take a course or study books like William Barden, Jr.'s The Z-80 Microcomputer Handbook, Nat Wadsworth's Z80 Software Gourmet Guide & Cookbook, and Leventhal & Saville's Z80 Assembly Language Subroutines.

Our program, SOUNDER, contains a short assembly language program you might want to study.

ASSEMBLER consists of an Editor--for developing your program, modifying it and saving it to tape or disk and the Assembler itself--for translating your assembly language program into machine language. For ease of use, ASSEMBLER is "menu-driven", so you don't have to remember all of the commands available.

EDITOR. Since each line in an assembly language program tends to be quite short, editing within a line is not provided for. It is possible to delete and insert new lines etc.

Enter the editor from the main MENU. The prompt C: will appear. Enter one of the following 1-letter commands--in lower case only. You will be prompted for line numbers when necessary.

d-delete. Enter 0 to quit without deleting any lines.

e-exit to BASIC. ASSEMBLER will still be loaded and ready to run, but any text you had in the buffer will be gone. Be sure it is saved to tape first.

i-insert text. Just push ENTER with an empty line to leave the insert mode.

LINE FORMAT. If no label is intended, a line must start with a space before the opcode. A * in the first column indicates the whole line is a comment. If a comment is to follow the opcode, a semi-colon must separate it from the opcode. LABELS may be used as addresses, up to maximum of 20 labels. Labels must be exactly six characters long. At least one space separates the label from the opcode. Following standard format, the opcode is separated from the argument by a space.

The backspace does not work during input, so if you make a mistake you should delete the line and reenter it.

Do not use the characters "tilde"or "exponent", in the text file. All data must be in hex, using the format $nn or $nnnn. Addresses may also be referred to as labels. Displacements must be in hex ; thus, if you create a disassembled file with DISASSEM and load it into ASSEMBLER, you may have to change some displacements in the source file. For example, DISASSEM might give you JR NZ,address. You must replace the address with the correct displacement, but you can get this value from the listing provided by DISASSEM. Input and output ports are also in hex.

Text may be entered in lower or upper case. ASSEMBLER will convert to upper case.

l-list lines. Control the listing with control-s. Push q to quit listing. CAUTION! We have learned the hard way that an experienced user may habitually stop the listing with a control-c because that is how it works in BASIC. However, if you do that, the whole ASSEMBLER program will be stopped, and you will be back in BASIC and lose your ASSEMBLER file. To repeat, control-s will make the listing pause, and the letter q will simply quit the listing.

m-return to main MENU.

n-new; clear all text from buffer.

p-print lines on printer. This works similarly to the l command.

r-replace lines. This is a combination of delete and insert.An empty line will stop replacing if you insert fewer lines than were indicated. However, all of the lines indicated will have been deleted.

MEMORY MAP. You can assemble routines at any location if you

just produce a listed version of the assembled program. If you want to assemble into RAM, you will be limited to the area from 27407 to 28107. This is room for a fair amount of object code. ASSEMBLER can protect this area with a LOMEM statement; it is the area ususally used for routines to be called from BASIC. If you want your routines high in memory, before loading assembler, do HIMEM:52931 and delete the LOMEM statement in line 20 of ASSEMBLER. Then the protected area will be from 52932-53631. Other areas are used by BASIC, DOS, and ASSEMBLER. ASSEMBLER comes configured with NO memory protected (LOMEM:27407). This provides for printed assemblies. We did this because we find that SmartBASIC is frustratingly erratic when the limits of memory are pushed. When a large block of memory is protected, a file might assemble perfectly one time and have odd characters injected in it the next time. Thus, you should figure just how much memory you need for the object file (with a preliminary assembly) and protect only that amount. If you wish, you can experiment with changing the LOMEM value, but this will leave more or less room for your source file. In the early lines of the ASSEMBLER program you will find the LOMEM statement and the value mz, which determines the size of the text buffer.

Most assemblers assemble at address 0000 unless instructed otherwise by an ORG statement. ASSEMBLER uses 27407 as the default, since this is the area most likely to be used with the ADAM. Of course, this address can be changed as any time with an ORG.

To convert ASSEMBLER to disk, change dr$ to 3 in line 60 of the program.

You can use ORG and DEFB directly. EQU can be simulated with a label and DEFB. TXT can be simulated with DEFB, by entering the hex value of the character wanted. Use only one byte with each DEFB. The DEFS pseudo-op can be simulated with labels and ORGs.

To save the assembled object file to tape or disk, assemble it into RAM, return to BASIC and save it with a BSAVE (see the BASIC manual.)

ASSEMBLER does a lot of error checking, but it doesn't check everything. You especially must be careful not to use duplicate labels. If you try to use too many labels, the editor will accept them, but your assembly will abort.

ASSEMBLER. When you select "assemble" from the main menu, you will be presented with an assemble menu, from which

you can select:

ASSEMBLE TO:

1.Screen and RAM

2.Screen, printer and RAM.

3.Screen.

4. Screen and printer.

LIMITATIONS OF THE ASSEMBLER. This is advertized as a "mini assembler" for two main reasons. Because it is written in BASIC, it is much slower that most commercial assemblers. Second, with BASIC and ASSEMBLER both loaded, there is not room in memory to develop large programs. You must be careful, for example, to provide a protected area in memory with a HIMEM or LOMEM command if you plan to assemble your program into memory for debugging. Having done this, it is possible to assemble the program into memory, return to BASIC and run the program to see if it works.

DISASSEM

As with the ASSEMBLER, this program is for more advanced users who are familiar with assembly language. A disassembler looks at a particular part of the computer's memory and translates the bit patterns of ones and zeroes into an assembly language file. It is used to figure out machine language programs that are in memory. Once you have the translated file, you can have it listed on the screen or printer or save it to tape as a file that will be compatible with the ASSEMBLER. Thus, you can take someone else's assembly language program, disassemble it and then modify it or add to it for your own purposes.

Another use for the disassembler is to disassemble the BASIC interpreter. This is the program that permits you to communicate with ADAM in SmartBASIC. All the things that BASIC can do it does through routines written in assembly language (and stored in the computer in machine language produced by an assembler). The BASIC interpreter resides from memory locations 256 to 27407, and the operating system routines are from 54160 to 65535. Much of these areas is unused in SmartBASIC, perhaps to leave room for a more sophisticated interpreter in the future. Even if you go to the trouble of disassembling BASIC, it won't make much sense unless you are very sophisticated with assembly language.

To disassemble a program usually takes several passes through the program, since most programmers intermix segments of program with segments of data, such as ASCII messages and variable tables. You should disassemble the section of memory you are interested in and then study the listing. Parts of it will obviously be data (lots of NOP's for example), and our DISASSEM provides the ASCII value of each memory location, so you will be able to pick out areas of text easily. Other clues to look for are RETurns, which may mark the end of subroutines and PUSH's, which may mark their beginning.

Now disassemble the memory area again, indicating the suspected "data areas" when DISASSEM asks for them. DISASSEM will prompt you for "start address", "end address", and "any data areas?"

The listing produced by DISASSEM will include:
addresses in
decimal:hex ASCII values hex values mnemonic arguments

You will also be prompted whether you want the new assembly language file saved to tape. This tape file will be compatible with the ASSEMBLER, so you can later load it into the assembler editor for modifications.

SAVING PROBLEMS. A serious problem sometimes arises during the process of saving a file to tape. Sometimes a save will be perfect, and the next time there might be some strange characters inserted in the file. This appears to be a bug in BASIC, since it only shows up some of the time. One way to avoid the problem is to enter mon c,i,o before you run the DISASSEM program. This is the "monitor" command in BASIC and it will cause all input and output from tape or disk to appear on the screen as it occurs. Thus, you can watch your disassembled file being saved. If you see odd characters, you will know that this is a bad save and the file you save won't run properly when you try to load it into ASSEMBLER. These bad characters can usually be recognized because they are printed as black on a white background--reverse of usual.The same problem occurs sometimes when our FILER program saves a file.

FILER

One of the most useful applications of computers is for the filing of complicated information in a way that provides easy access to the information. FILER is a general "database" program that is designed to be modified for whatever kinds of information you need to file. You can keep mailing lists, for example, and later use our LABELS program to print out mailing labels for everyone on your mailing list who meets whatever criteria you set. You might want to send your annual reunion invitation only to people with the last name Smith who sent you a birthday card last year. In addition to your mailing list database, you can use FILER to create an inventory of all your posessions, their value, age, dealer from whom purchased, etc.

You can have any number of databases, each designed to suit your needs.

FILER can store up to 99 "records" in memory at any one time. Of course, you can have several tape or disk files of records if you need more than 99 records. Each record can have within it up to 10 categories, which you define for each database you want to create. For example, categories might include, last name, first name address, age, relationship, Christmas card?, dress size, or whatever you wish. Each record can contain up to 115 characters.

As with our other programs, FILER is "menu driven" for ease of use. There is a main menu you will see first. It will give you the choices:

1.Instructions
2.Add a record
3.Delete a record
4.Edit a record
5.View or print raw file
6.Select/sort records
7.Name data categories
8.Load a file
9.Save a file
10.Quit

The first thing you should do is enter 7 and name your data categories. The program will guide you through this process and tell you how to return to the main menu when you're done. Then enter 2 to start putting data into the file, etc.

Most of the functions are self-explanatory or are explained within the program. The selecting and sorting function permits you to select only those records that meet certain criteria. You can set criteria for each category or only for one or two. Only those records that meet all criteria will be selected. When the program asks you for criteria, you can enter "partial criteria." Thus, if you set the criterion for the category "last name" as Thom, the program will match to the four leftmost letters and pick all records with the last name Thompson, Thomson, Thomlinson etc.

Function 6 also lets you sort the records chosen alphabetically or numerically. The program will ask you which category you want to sort on. If you use zipcode, for example, the selected records will be sorted numerically. Last name would be sorted alphabetically. Note that alphabetical sorting lists all upper case letters before any lower case.

It takes a long time to sort long records.

To convert FILER to disk, just change dr$ to 3 in line 30 of the program.

PROBLEMS WITH SAVING. Sometimes a serious problem arises with saving files. There seems to be a bug in BASIC, since most times FILER saves perfectly, but ocassionally strange characters will be inserted and will either cause the whole program to crash during a save (giving you an I/O error) or saving successfully but producing a file that won't run properly the next time it is loaded. There are some strategies to get around these problems.

First, if you get an I/O error--for any reason, such as a misaligned or not-inserted data pack or because of an I/O error during a save, just enter GOTO 1100 and you will be back in the program with all your data intact. This is an especially important strategy because if you enter RUN, the program will start all over and whatever data you had entered will be lost--a very frustrating experience.

Once you are back in the program you can save the file again, using a different name. The reason for this is that the faulty save created a name on the tape or disk but never finished making it a valid file. The next time you try to save, if you use the same name, BASIC will see the duplicate name and not accept the save. Later you can delete the faulty file, just using BASIC.

It is possible to monitor the actual saving process on the screen. Before you RUN the FILER program, enter

mon c,i,o This tells BASIC to monitor all input and output on the screen. When you do a SAVE from FILER, you will see the data go by on the screen and will recognize good data when you see it. The first line will start with a line number followed by the category names for this file. Each category name is separated by the ^ character (which is why you must not use ^ in your file). Subsequent lines will be your records, each record packed onto one numbered line, again using the ^ as a category separator. If you see a character that looks like it is printed in black against a white background (reverse of normal), you will know that this is a bad save; BASIC has messed up a character and lost some of your information. Our practice at this point is to stamp our feet, snarl, enter GOTO 1100, and save the file again. ALWAYS back up important information with a second save.

With this second kind of error--bad characters in the middle of the file--the SAVE will be completed without an I/O error, but when you load in the file later and try to use it with FILER, you will get the message "illegal quantity in 8040". This means that FILER has run across the strange character.

We have saved files of 99 records several times in a row with no mishaps, but occasionally a save is bad. We have talked with customers who never have this problem and others for whom it is a frequent problem.

LABELS

This program is designed to use with files created by FILER. The main menu permits the selection of particular entries from your data base in the same way FILER does and then permits you to print out the addresses or other data entries in several different formats. The most likely application of LABELS is for printing on sheets of pre-gummed labels you can buy at any stationery store. You can print mailing labels for your whole sky-diving club (and then make deletions when necessary, using FILER). You can print up many labels with your own name and address if you wish--or labels for your record collection.

It is also possible to format the printout as a simple listing of particular entries, but that is also possible with FILER. With LABELS, you can format the printout in two columns per page--for example, to produce a name and address list for distribution to your club members. The main menu will guide you through the different formats available.

FINANCE

As with many of our programs, FINANCE is a combination of several programs which you can save as separate programs if you wish. We have also left gaps in the line number sequence to make it easier for you to modify the programs if you wish. The main menu will offer you the three most popular applications of microcomputers to home finances--monthly budgeting, savings and loans interest projections, and metric conversions. Each of these subprograms has its own menu, and you will see that the functions provided in the first two will permit you to do a lot of financial planning and record keeping that would be too tedious without the computer to do the calculating.

You likely will want to save the metric converter as a separate program if you will be using it frequently.

FUGUE

When you run this program your computer will play for you. The piece that you will hear is a Bach fugue (it's a kind a of music) in C minor from the Well-tempered Clavier. You should hear what a musician that Adam is, three instruments at once.

Some of the programming for this application was done in machine language as in SOUNDER. Again the machine language command and data are poked into memory, so that you have easy access to the machine code. By the way, you should be able to use this program as a means of typing in your own music. The data statements at the end contain the music: The notes are numbers from 1 to 49 (A to A 4 octaves), and the durations are as below

1 - 1/32 note	2 - 1/16 note
4 - 1/8 note	8 - quarter note
16 - half note	32 - whole note

In addition, the voices involved are marked by REM statements.

For instance,

```
REM voice 2
DATA 1,4,13,4,25,4,37,4,49,4
```

Will play the A in each of the four octaves as eighth notes in the second voice

SOUNDER

Where's the sound? Not as profound as where's the beef, but very appropriate. ADAM can make sounds, quite complex sounds if you know how. The problem is that very few people know how. This program will be very helpful to those who want to add sounds to their BASIC programs and to those who simply like to make a lot of noise.

The entire program is menu-driven so it is very easy to use. The initial menu will ask if you want to make music, make sound, play canned sounds, play canned music, or change various parameters (then of course there is number 9).

The parameters are changed via graphic manipulations. You simply draw the ups and downs of the volume and of the pitch.

Some the program's features you will have to learn by doing, so just experiment.

DESIGN

Have you ever wondered how those game designers figure out what Buck Roger's space ship will look like? This program should give you some insight then. DESIGN will let you design some 16x16 figures in high resolution through the use of an enlarged grid. These figures could subsequently be used in other Basic programs (those of your own design).

You use the joystick to move a cursor around the grid and press a fire button to set or unset a dot. The number key pad is used to change the current color. Simply press a number and the color that you are working in will change.

The colors are as follows:

1- GREEN	2-DARK RED	3-WHITE
4-MAGENTA	5-MEDIUM RED	6-MED. BLUE
7-DARK YELLOW	8-DARK BLUE	9-YELLOW
0-GREY		

In addition to these features the program will allow you to load previously worked on pictures or to save your current picture. When you save a picture the picture is saved at location 27407 with the name that you give it. Of course, you don't need DESIGN loaded in to use these saved pictures just use the following program:

```
BLOAD _____
HGR
FOR Y-27407 TO 27662
PT-PEEK(Y)
HCOLOR-PT:HPLOT I, J
I-I+1:IF (Y-27407)/16-INT((Y-27407)/16) THEN
   I-I-16:J-J+1
NEXT Y
```

The blank above is for the name of the shape.

TENNIS

SKILL GAME 1

This is a smartBASIC version of the world's very first videogame. The original was a one color, flashing oscilloscope version, while this version is colorful and has many skill levels. You have probably played one version or another of this classic game. Two players move paddles up and down the screen at either endtrying to keep the ball from touching their side of the screen. If the ball touches the other side of the screen you receive one point, and if the ball touches your side of the screen the other player receives one point. Whoever is first to get 15 points wins the game.

Simple, right? Try all skill levels and speeds and be sure to have fun.

This program is especially appropriate for those of you who are interested in listing BONANZA programs to try and figure them out. It is suggested that you list the programs out on paper (PR#1:LIST:PR#0). The present program fits exactly onto an 8 1/2 x 11 sheet, and it contains many useful ADAM videogame techniques.

BREAKOUT

SKILL GAME 2

This is another popular video game. This is a one player game in which it is your job to break through a wall at the top of the screen by bouncing a ball with your paddle. However, if you miss bouncing the ball back up you lose it. Remember you only have five balls to use.

Once you break through the wall you will find another one just beyond it with higher point bricks.

One thing you might want to try after you get a feel for the game is to put english on the ball by pressing the fire buttons as the ball hits the paddle. With the buttons and some practice you can achieve a great deal of control over the ball.

TRY ME

EDUCATION GAMES

TRY ME consists of two educational games: a spelling game and a mathematical game. By the way, it is called TRY ME so that children are not immediately turned off by what sounds like hard work. In truth the games aren't hard at all; they can actually be fun. The player is faced with a challenge, and if the challenge is met then the player receives an audio-visual congratulations.

The first of the two games, Guess Who, asks the player to guess the animal that the computer is "thinking about". The computer will give increasingly specific clues until the guess is correct. If the guess is correct, but the spelling is incorrect then a message will appear indicating that. The child will then be given a second chance to spell the word; if the spelling is still wrong, the correct spelling will be displayed.

All correct responses will be rewarded, and the child will receive more points for faster and more accurate guessing. This program will teach the player not only spelling but also good guessing strategy.

It would be possible for you to add testing words to the program by adding the clues and spelling in data statements at the end, but be certain that the very last data statement reads:

```
DATA END
```

The second game, Racer, asks the player to answer mathematical questions in order to fill a fuel tank in a racer. There is a skill level choice:

1) add	2) 2 digit add
3) add/subtract	4) 2 digit add/subtrac
5) add/subtract/ multiply	6) 2 digit add/subtract/ multiply

This program will provide very useful practice for every child whether struggling or not. It might even be helpful for adults whether struggling or not.