SCS ADVENTURE GENERATOR

BY J.M. NOWICKI

DISTRIBUTED BY

SECOND CITY

SOFTWARE

J.M. Nowicki
Box 32215
San Jose, CA 95152

Special thanks to Jorge Mir for his helpful hints, suggestions, and creative ideas in developing this software package.

SCS ADVENTURE GENERATOR

GENERAL INTRODUCTION

This package of programs and documentation allows the user to create complex adventure games that, once compiled, are 100% in machine language format and very fast in execution. When used with the Spectrum Projects VOICE PACK, the user may compile talking adventures.

For creating silent adventures, a 32K CoCo with 1 disk drive is required. Once the adventure has been compiled, it may be saved to disk or tape and run from either media.

For talking adventures, system requirements are a 64K CoCo, 1 disk drive, a VOICE PACK and its associated software. To run a talking adventure, the system requirements are the same as for creating it and it may also be saved and run from tape or disk. See the memory map page that shows the RAM allocations for silent and talking modes.

All data for rooms, objects, verbs, and responses are defined using a series of simple text editors that generate source code that will later be compiled along with your program plot to produce the finished adventure.

An ADVENTURE LANGUAGE is supplied with which the user may create scenarios and program very specific reactions to actions taken by the player. It is short and easy to learn and will give you all the tools you need to create simple or very complex adventures.

All editors have a printout feature for all data. While not mandatory, it is very helpful to have a printer. Having a hardcopy of everything in front of you will help bring all the elements of the adventure together in your mind.

PROGRAMS INCLUDED IN DISK

This software package contains the following program files on the enclosed disk.

SAG/BAS -- this is the main program from which all other programs are called. In order to start the system, you must type RUN "SAG" and press <ENTER>. You then follow the various menu selections to access the subroutines within the system.

BRAINS/BIN -- the machine language program that runs the silent adventure once it has been compliled.

TBRAINS/BIN -- same as BRAINS/BIN, except for talking adventures.

COMPILER/BAS -- a BASIC program that compiles the adventure from source code files containing data for verbs, objects, rooms and responses.

SAGLANG/BAS -- a BASIC program to create and edit command lines that control the adventure using a special language created for adventure writing.

SAGVOR/BAS -- a BASIC program to create and edit source code files for the verbs, objects and rooms that will be read by COMPILER/BAS.

SAGMF/BAS -- a BASIC program to create and edit text screen messages that will serve as

responses or object descriptions. This program also defines the meanings of condition flags as well as their value. The source code files created with this program will be read by COMPILER/BAS.

SAGPRINT/BAS -- a BASIC program that produces a hard copy printout of the adventure language steps. The printout is "translated" into the actual meaning of each step for ease of editing and reviewing the actual adventure program steps created by the user.

SAGSAVE/BAS -- a BASIC program that performs all the necessary steps, automatically, to merge the various machine language programs created with this system and save them to disk as a single program.

ALLRAM/BIN -- a short machine language utility that turns on your 64K RAM and copies the ROMs to RAM. You will need to use this program to compile and run talking adventures.

SPECIFICATIONS

VERBS -- up to 70 verbs may be used or however many will fit in the 400 byte VERB BUFFER. Using a mix of verbs with character lengths of from 3 to 10 letters will let you define on average 30 to 50 verbs.

OBJECTS -- up to 255 objects may be defined within an OBJECT BUFFER of 3100 bytes.

ROOMS -- up to 99 rooms may be defined within a 2800 byte ROOM BUFFER.

CONDITION FLAGS -- 255 general use condition flags are available and may be assigned any meaning within the program as a variable with a changeable value of 0 or 1.

HELP MESSAGES -- these may be defined in almost any number to provide responses based upon the players progress determined by testing FLAG values through the LANGUAGE.

RESPONSES -- these are in the form of TEXT SCREEN printout and may be defined with up to 510 available within the 8600 byte MESSAGE BUFFER.

The ADVENTURE LANGUAGE PROGRAM allows up to 700 program lines. The silent version has a PROGRAM BUFFER of 8600 bytes while the talking version has a PROGRAM BUFFER of 7680 bytes.

HELPFUL HINTS

To create an adventure you must understand the process involved in creating one. Think of yourself as an architect building a structure from the ground up. If you have a big ego think of yourself as a god with a universe waiting to be created or as a movie director assembling the set, actors, the script and all of the small details.

Before you start entering data into the editors you should make a map of your adventure and fill in some of the details. ROOMS do not have to be a room in a house, they can be an outdoor area, even planets that you travel to and from. A physical room can be split up and defined as more than one room. A hallway could have a north and south side, a beach could have east and west front, etc.

You should make a map of your room layout on paper and indicate room numbers and directions towards which the player can travel (N,S,E,W,U,D). Room layouts can have multiple levels or just one level.

Inside of each room on your map jot down the objects found in that room, and whether they can be seen and/or picked up by the adventurer.

If you own a wordprocessor or database system, you can enter a lot of your thoughts with ease and then print the data on paper so it is handy as you enter the data using this system. For example, the data may look like this:

Room number 10 - In front of a house -- objects: a box, a broom, a door -- door must be unlocked before it can be opened -- if the box is examined while in inventory, then a key appears otherwise messages are printed informing adventurer of what he needs -- Directions are N to Room 11, S to Room 21 -- whatever other information is needed to help you out in entering the data.

The best way to become acquainted with this system is to develop a short adventure. Say, an adventure with only a couple of rooms and a few objects. Maybe a door that needs to be unlocked with a key and after you enter the house the adventurer must find a note hidden in a desk with the adventure ending when the adventurer reads the note which says "END OF THIS ADVENTURE". Any simple adventure such as this one will be helpful in getting started. In other words: DON'T TRY TO CREATE ANOTHER BLACK SANCTUM YOUR FIRST TIME!

Enter all of this information into the system using the various editors. After this is done, you are then ready to write the adventure program.

Compiling the adventure to the finished product is the last step and is covered later in this manual.

V/R/O EDITOR

This section of the program allows you to create and edit source code text files for your verbs, objects and rooms. These files may be saved to disk and reloaded at a later time for revision. If you have a printer, you will find it handy to print out the data for future reference while entering additional data under other sections of the system.

Once you enter the V/R/O EDITOR, you are presented with a menu giving you six choices:

(1) CREATE NEW FILE

This option will take you to the editor where you may begin creating your new adventure. This will clear whatever was in memory and clear all buffers. Upon entering this subroutine, you will be given a choice as to which type of item you want to create (i.e., verbs, rooms or objects).

(2) LOAD OLD FILE

If you have previously saved a source code file you may load it into memory with this option. Enter the filename without any extension and press <ENTER>. The name may not exceed 8 characters.

(3) SAVE CURRENT DATA

This option lets you save what you have done up that point by storing the contents of all buffers to disk. Enter a filename without an extension and press <ENTER>. The filename may not exceed 8 characters. All source code files are automatically assigned the extension "/VOR".

(4) PRINTOUT DATA

If you have a printer, you may selectively print out the verbs, objects and rooms. You will be prompted as to which of these three items you want printed. You will also be asked to get your printer ready before proceeding.

(5) DATA EDITOR

This will take you to the EDITOR whether you have a previously loaded adventure in memory or not. Whenever editing verbs, objects or rooms, the bottom of the screen will show the various editing functions available to you which are accessible by pressing a single key as described below:

Add -- By pressing "A" you will be allowed to enter a new item.

Delete -- By pressing "D", the item you are viewing will be deleted from the buffer. Please note that when an item is deleted, the remaining items are renumbered. You need to be very careful when deleting items because of this renumbering. If you do not want to renumber the rest of the items, then just change it to some dummy item (for example, call the item "XXX").

Edit -- By pressing "E", you can edit (change) any item which has been previously entered.

Next -- By pressing "N", the program will advance to the next item in memory. If there are no more items, nothing will happen.

Previous -- By pressing "P", the program will go back to the previous item in memory. If you are viewing item 1, nothing will happen.

Quit -- By pressing "Q", the program will return to the main menu in this section of the system.

(6) MAIN MENU

Pressing the "6" key from the menu will take you to the system's main menu.

ENTERING AND EDITING VERBS

Once you select to enter or edit verbs, you will be shown the list of verbs currently in the verb buffer. The system has built in a few commonly used verbs. These are: GO, GET, DROP and EXAMINE.

The screen will contain ten verbs at a time along with the related verb number for identification purposes when entering the various steps using the language editor (this is further explained elsewhere in the documentation). You can view verbs, 10 at a time, by pressing "N" for next screen or "P" for previous screen.

Also shown on the screen will be the number of bytes available for new verbs. You have a total of 400 bytes in the verb buffer. The maximum number of verbs you can enter is 70, but this depends on the length of the verbs. Longer verbs will, of course, use more bytes in the verb buffer and you may run out of buffer space even though you may have entered less than 70 verbs.

If you press "A" for adding new verbs, you will be prompted for the verb name following the number assigned to the new verb. You can continue to enter verbs until done, at which time you need to press <ENTER>. You can edit any verb previously entered, by selecting "E", after which you will be asked for the verb number to be edited. Just enter the revised verb when prompted to do so.

RULES THAT MUST BE FOLLOWED:

Maximum number of verbs is 70 with a maximum buffer of 400 bytes.

You cannot edit the verbs GO or GET.

All verbs must contain at least 3 characters.

Only the first 4 characters of a verb are recognized by the program (3 characters, if a 3-letter verb).

The first three characters of each verb must be different in every case. For example, you cannot use the verbs DECAPITATE and DECAFINATE in the same adventure or you might get unpredictable results.

ENTERING AND EDITING ROOMS

You will notice that ROOM 1 has a "dummy" definition which must be changed as you create an adventure. DO NOT DELETE ROOM, instead, replace it with the description of the first room in your adventure. Please note that your adventure need not start in room one. When you compile your completed adventure you will be asked the room number in which the adventure is to start.

The description of each room can be up to 250 characters long, although a practical limit is around 200. You can use the character "/" as a carriage return if the description exceeds 31 characters so that it is properly formated on the screen. Of course, you can also use blank spaces, but the use of the "/" will conserve buffer space.

For talking adventure you must use leading and trailing spaces to line up the next word on the next line. Also, you cannot use the slash symbol for talking adventures.

The room buffer has a limit of 2800 bytes. As with the verbs, you can use as many rooms as will fit in the room buffer with a maximum of 99 rooms.

Use the "V" option to check buffer space available for additional rooms.

Once you enter the description of a room, you will be asked to enter the obvious directions out of the room. You can enter any combination of the letters "NSEWUD" to indicate NORTH, SOUTH, EAST, WEST, UP or DOWM.

These directions will serve as signposts only and may not be the only directions available nor is exit in those directions need be automatic. For example, you can indicate that EAST is one of the directions but you can block passage in that direction until the player does something to unblock it as determined by your program. You may also go in directions not indicated. For example, in a room that says you can go EAST and WEST but also contains a magic mirror, you can program the mirror to take you into another room if you ENTER MIRROR. More on this later.

RULES YOU MUST FOLLOW:

Number of characters in a room description cannot exceed 250.

Maximum number of rooms is 99 with a maximum buffer of 2800 bytes.

The slash bar ("/") to format text on the screen cannot be used in talking adventures.

Room descriptions must contain leading and trailing spaces in talking adventures.

ENTERING AND EDITING OBJECTS

There are 8 OBJECTS already in the buffer (NORTH, SOUTH, EAST, WEST, UP, DOWN, ROOM and INVEN). You can have up to a maximum of 255 objects with a maximum of 3100 bytes in the object buffer.

The first 6 of these are directions and cannot be changed. Object ROOM can be changed to AROUND. It is used in the adventure language to look at where you are and what is there. Many times, after doing various things in a room, the room description, directions and object list may scroll up the screen. In order to re-examine the room

after the description scrolls off the screen, you need to EXAMINE ROOM, after which the screen will clear and the effect will be the same as if you had just entered the room.

Your own objects will start with object number 9 and each object name may contain up to a maximum of 31 characters.

Once you have described an object, you will be asked to enter a four-character keyword which the program will use to reconize the object. For example, if the object is A RING OF KEYS, you could use RING or KEYS for the keyword. If the object is A SILK SMOKING JACKET you should use JACK as the keyword. Each object must have its own unique keyword (called the OBJECT REFERENCE STRING), so no two objects can contain the same keyword or the results will be unpredictable.

After you enter the keyword for an object, you will be asked a series of questions to which you are to answer "Y", "N" or a number. The questions are as follows:

Can the object be gotten? -- Many of the objects described in a room should not be gotten. For example, your room description might include objects such as "A BLUE CONVERTIBLE", "AN OLD TRACTOR", etc., which surely should not end up in inventory. Also, you may want the player to be able to get the item only if certain conditions are present, such as an object which is nailed to something else and can only be gotten if the player has a crowbar or a hammer, etc.

In which room does the object start? -- You are to indicate the room number to this question. Please note that objects can end up in any room, so this is just to indicate where it starts out in the adventure.

Can the object be readily seen? -- In some instances, you may not want to show the object until certain conditions are met. For example, a key may be in a desk drawer and can only be seen if the drawer is openned by the player.

Which message number to be printed? -- Here you need to indicate the message number related to the object if it is examined. Message numbers describing objects can be from 1 to 255 and are entered using the MESSAGE EDITOR subroutine described elsewhere. If no specific message is used to describe the object, then enter a zero. As an example, if the player says EXAMINE CHEST the message number specified could contain "IT LOOKS LIKE AN OLD PIRATE'S CHEST!".

RULES YOU MUST FOLLOW:

The object description cannot exceed 31 characters.

Total number of characters for all objects cannot exceed 3100.

Message numbers describing objects must be in the 1 to 255 range (up to 512 messages can be entered in the system).

The object keyword must contain at least 4 characters.

LANGUAGE EDITOR

This language is simple and easy to learn and will allow you to create simple adventures or incredibly complex situations that will make it very difficult for the player to succeed. It's all up to you.

There are 2 types of commands: CONDITIONAL INPUT and UNCONDITIONAL OUTPUT commands. In other words, IF certain conditions exist, THEN a certain outcome will take place.

CONDITIONAL INPUT COMMANDS

There are only 6 INPUT COMMANDS but they will provide you with all the input that your program will need. These are:

Ixxx -- If object xxx is currently in the players inventory

Fxxx -- If condition flag xxx is set to 1

Lxxx -- If condition flag xxx is set to 0

Nxxx -- If the noun (object) inputed is xxx

Rxxx -- If the room currently occupied is xxx

Vxxx -- If the verb used is xxx

The I command in a program followed by an object number will test to see whether or not the player is carrying a particular object. Let's say you want to LIGHT LANTERN. You would have the program first check to see if the LANTERN is in inventory befor it can be lighted.

The F command will check the status of one of the 255 condition flags. The program will check the condition of the flag to see if it is set (value = 1). In the above LIGHT LANTERN example, you could use a flag to indicate if the lamp is lit or not.

The L command will check the condition flag specified to see if it is reset (value = 0) and return to the program with this status.

The N command is used to specify a particular object or direction. The NOUN numbers specified in the V/R/O EDITOR source code file defines the noun number used here and are both the same. For example, if you specified object 27 as SWORD then N027 would tell the program to check and see if the noun used is number 27 (SWORD).

The R command lets you check the room you are currently in if you want specific events to happen only in a particlar room.

The V command identifies the verb in a similar way using its VERB NUMBER. Verb 1 is GO, and in a command line would be V001.

Using the INPUT commands above, you create a string of IF/AND statements. Each command used in a program line is a re-enterant subroutine that checks for the condition specified. If it returns the appropriate value, it is considered a match and continues checking the next command to see if it is also true. If all the conditions are met by the time it sees the equal sign (=), it then starts executing the output commands. If

one condition is not found to be true, then the program checks the next program line for a match.

If none is found, then the program returns the message "NOTHING SPECIAL HAPPENED".

Here is an example of how input commands are used:

R027V042N214 = do output commands

This program line says: If the player is in ROOM 27, VERB being used is number 42 and OBJECT is number 214, then do the output commands that follow.

Here is another example:

R054V044N202F018= do output commands

Assuming you have predefined that VERB 44 is KILL, OBJECT 202 is the DRAGON, FLAG 18 set to 1 means the DRAGON is alive else a 0 means he is dead and cannot be killed again then this command line would mean: If the player is in ROOM 54 and he inputs KILL DRAGON and the dragon is still alive, then execute output commands.

Another example:

R018V034N178I118F014L247= do output commands

If you are in ROOM 18 and you input OPEN COFFIN and you have a HAMMER that is object 118 in your inventory and you have removed the coffin nails and broken the seal, then execute the output commands. FLAG 14 with a 1 in it tells you that the coffin nails have been removed, while L247 tells you that the seal has also been broken thus allowing you to open the coffin. Here it is assumed that VERB 34 is OPEN, and that NOUN (object) 178 is COFFIN.

There are only 6 INPUT commands and the concepts used are quite simple but when using them in clever combinations, particularly the FLAGS, you can create some very intricate and complex situations.

OUTPUT COMMANDS

There are a total of 12 OUTPUT COMMANDS that can alter many of the conditions refered to by the INPUT COMMANDS.

Bxxx -- Makes an object ungetable. Opposite command is G.

Exxx -- Makes the player enter a new room. That is, moves the player from the room he is currently in to the new room as specified by xxx. The screen is cleared and displayed on the screen is the room description, objects that can be readily seen, obvious ways to go followed by the prompt to enter a VERB NOUN.

Gxxx -- Makes an object getable. Opposite command is B.

Jxxx -- Changes the status of an object so it can be readily seen when entering a room so that it can be examined. Opposite command is U.

Pxxx -- Prints a message as specified by xxx from the first message buffer (messages 1-255).

Qxxx -- Prints a message as specified by xxx from the second message buffer (messages 1-255 are equivalent to messages 256-512 in this buffer).

Sxxx -- Sets the FLAG specified by xxx to a value of 1. Opposite command is T.

Txxx -- Resets the flag specified by xxx to a value of 0. Opposite command is S.

Uxxx -- Makes an object invisible so that it will not be listed as readily seen when you enter a room and so that if you try to examine it it can't be seen. This command is the opposite of J.

Here is an example of a complete program line:

```
R017V021N114F017=T017P125
```

If you are in Room 17 and if the verb used is Verb 21 (KILL) and the noun used is Noun 114 (DRAGON) and if Flag 17 is set to a value of 1 (DRAGON ALIVE) then Kill the DRAGON by reseting condition Flag 17 to 0 (DRAGON DEAD) and print message 125 that says,"THE DRAGON MOANS AND FALLS TO HIS DEATH".

GENERAL NOTES ON PROGRAMMING

You will notice that when you are entering letter commands using the LANGUAGE EDITOR they will be displayed as lowercase letters to help you see them. DO NOT SHIFT TO UPPERCASE!. The language understands lowercase command letters only. The keyboard is shifted by software automatically for you and unshifts it when you exit the editor to save or read in a disk source code file.

All numbers that follow a command letter must be in a 3 digit format. If the VERB command is used and you specify "V9" as one of the conditions, the program will add leading zeroes and will display the above as "V009" next time you see it.

The BRAINS of the program that contains the program interpreter has some error trapping so that, if you enter a wrong or non existant letter command, the program will not crash or display an error message. If no match is found for the user's input line, then the program will return the message: "NOTHING SPECIAL HAPPENNED".

When you have completed your adventure program, you will compile it by selecting the COMPILE PROGRAM option from the main menu. The program will be compiled from the source code in memory at the time and assembled to disk to be read in by the COMPILER/BAS program.

If your program is too long, the screen will display a byte overflow message. If this is the case, you will have to delete some program lines to make it fit. For silent adventures, you have a buffer of 8600 bytes and, using the compressed language format, you will usually have room left over.

For talking adventures, you have a program buffer of 7680 bytes. You must be in 64K RAM mode to compile or run talking adventures. Use ALLRAM/BIN to enter 64K RAM mode.

The Verify command from the editor will allow you to check how many free bytes remain in the program buffer. This number reflects how many bytes the source code will be compiled down to, not the size of the program source code buffer.

USING THE PROGRAM EDITOR

Like the other editors in this software package, the same commands are found: Add, Delete, Edit, Next, Previous, and Verify. You also have the Save to disk and Load to disk and get printed hardcopy from the main menu.

Unlike the other editors, you can delete a program line without worrying about the renumbering problem since line numbers are not compiled with the source code.

WRITING THE ADVENTURE PROGRAM

You will eventually develop your own programming style like you do with BASIC or Assembly Language. You should write a few very short and simple adventures even if they have little or no plot, just to get the hang of using the adventure language. Use about 7 rooms, 15 verbs and 20 objects. Later, when you fully understand how the commands work, you can start developing longer and more complicated adventures.

The first program lines you enter should pertain to moving from room to room. Then build the plot based upon the rooms and objects being used.

As mentioned earlier, you can do tricks with directions and objects that can serve as portals. For instance, you could have a room with a magic mirror in it so that if you ENTER MIRROR you enter another room. Using ENTER as verb 15 and mirror as noun 22 and the mirror located in room 84 the program line would look like this:

```
V015N022R084=E085
```

This line would cause you to exit room 84 and enter room 85. Use your creativity and imagination in using the adventure language to produce your adventure.

RESIDENT PROGRAM LOGIC

Some things are already taken care of for you, such as getting and dropping objects, examining your surroundings, objects and inventory.

When you entered new objects you already specified if they could be readily seen and if they could be gotten. The status of these object related flags can be changed using the LANGUAGE.

If you try to examine an object that is not in the currently occupied room, is not in the player's inventory or cannot yet be seen, the program will tell you:

I DON'T SEE THAT HERE!

Same thing happens if you try to get that object.

Objects that you try to get that cannot yet be gotten will return the message:

I CAN'T GET THAT RIGHT NOW!

When the player gets something to put in his personal inventory, the program checks to see if it is in the currently occupied room, if it can be seen, and if it can be gotten. If all these conditions are true, then the object is placed in inventory. When something has been gotten, a response will say:

OK, I GOT IT!

When the player drops an object from inventory, the response will be:

OK, I DROPPED IT

If the player tries to drop an object that is not being carried in inventory, the response will be:

I DON'T HAVE THAT WITH ME!!

When the maximum number of objects to be carried has been reached and the player tries to get something else the response will be:

I CAN'T CARRY ANY MORE!!

FLAG EDITOR

This program lets you define the default values of up to 255 condition flags. The values used should be the condition you want them to be in at the beginning of the adventure.

Once you enter this section of the system, you will be presented with the following choices:

(1) Load a flag source code file from disk

(2) Save a source code file to disk

(3) Editor

(4) Printout flag file.

All source code files generated by the program and read in by the COMPILER have an extension of "/FLG", so all you have to do is type in the file name of the adventure as with the other editor programs.

From the editor you can enter a phrase of up to 31 characters to remind you what that flag is for.

For example:

DRAGON ALIVE = 1 DRAGON DEAD = 0

You then enter either a 0 to reset or a 1 to set the flag.

Each of the 255 flags can be edited and printed.

The descriptions of the flags are not used by the COMPILER. They are for your reference only and you should use them in order to remind you of their meanings as you develop the adventure.

The flag values are essential but you can write an adventure without them if you don't use the FLAG change commands.

As an example, you could use flag 1 for a LANTERN. A value of 1 could indicate that it is lit, while a 0 means it is not lit. These flag values can be changed during the course of the adventure with the LANGUAGE.

For a more complete description of using FLAGS, see the section on using the ADVENTURE LANGUAGE.

MESSAGE EDITOR

This program is used to create and edit text messages that the program will use as responses to specific actions taken by the player, or to further describe an object when it is examined.

When you enter this section of the system, you will be presented with the following menu:

(1) CREATE A NEW FILE

(2) EDIT MESSAGES

(3) LOAD FILE FROM DISK

(4) SAVE FILE TO DISK

(5) PRINTOUT MESSAGES

When you want to begin creating a new adventure you would choose menu selection 1. This will clear the buffers and start you out with message #1 in buffer P which contains a dummy message. Select the Edit option and put your first message in the slot. Most times you will want to enter object descriptions before doing anything else because buffer P must be used for this purpose.

After having loaded an existing message source code file from disk, you enter the editor to work on them by selecting option 2 from the main menu.

There are 2 buffers for messages, BUFFER P and BUFFER Q. Each buffer can contain up to 255 messages but you have to fill buffer P before using buffer Q. These two buffers, P & Q, are related to the P & Q output commands used in the adventure language. This gives you a total of 512 messages to use for object descriptions, responses or help messages.

When you enter the editor, you can select the starting message to view or edit. Messages 1-255 are stord in buffer P while messages 256-512 are stored in buffer Q. To work on the messages in buffer P, select the number of the message within the range displayed on the screen with a maximum number of 255. To work on buffer Q, enter a number between 256 and the upper limit displayed on the screen.

For instance, if you want to work on message 27 in buffer P, you would enter 27. If you want to work on message 14 in buffer Q, then you would enter 269 (255 + 14).

The messages 1-255 in BUFFER P must be used for further describing objects as they are examined. Not every object has to have this object elaboration, so after you have used as many as needed for objects you can use the rest for responses.

The ADVENTURE LANGUAGE will call these messages up by their number during the course of the adventure play. You have the same options available as when you edited the verbs: Add, Delete, Edit, Next, Previous, Quit, and Verify. You may also remove a message and fill it with a dummy message such as "XXX" so as not to disturb the numbering of the messages.

The slash "/" is used for screen text linefeeds like it did in creating rooms. If the message will be longer than 31 characters, you must use a slash to effect a linefeed or format the message by adding blank spaces so that words are not split at the end of the screen.

Here is an example:

THE DRAGON HUFFS AND PUFFS AND/BLOWS A GUST OF FLAME AT YOU!

This is the source code form of a message using the slash symbol (/) for a line feed. When you enter this message into the editor, you will see displayed both the source code form and the way it will actually appear in the adventure you create.

The above example will screen print as follows:

THE DRAGON HUFFS & PUFFS AND
BLOWS A GUST OF FLAME AT YOU!

When using this editor to produce talking adventures, you must not use the slash (/) symbol for linefeeds. If you want your message to be longer than 1 line of 31 characters, you must fill the end of the line with spaces to line up the next word at the next screen line.

The above example, when used in a talking adventure, could look like this:

THE DRAGON HUFFS & PUFFS AND BLOWS A GUST OF FLAME AT YOU!

The 4 spaces between the words AND and BLOWS fills up the rest of the first line with spaces so the first character of the word BLOWS starts on the 2nd line. The effect when it is printed on the screen will be the same as if you used the slash ("/") for a linefeed. It doesn't really matter in silent adventures if you use the slash or spaces to effect a line feed as the result will be the same. The talking adventures speak each line and look for a linefeed (CHR$ 13) to end a spoken line. A slash will produce a CHR$ 13 in machine language once compiled and terminate the spoken line prematurely.

Messages may be up to 250 bytes long. Multiple messages may be displayed using the language for very long responses or help messages.

You may indent the first line of each message if desired for a better visual effect. A message may consist of upper or lowercase or any mixture you desire. Lines may be centered by using leading spaces as well.

You have a combined buffer space of 8600 bytes for message buffers P & Q.

To load an existing source code file of messages, press selection 3 from the main menu. Enter the name of your adventure without a disk file extension and press <ENTER>. The file will be loaded and the program will return you to the main menu. All source code files generated and read by the program and the COMPILER program have an extension of "/MSG".

To save a source code file of messages currently in memory, press selection 4 from the main menu. Enter the name of the adventure without a disk file extension and press <ENTER>. The file will be saved to disk and return you to the main menu.

To get hardcopy of your messages, press selection 5 from the main menu. Get your printer ready to go and press <ENTER>. Your message file will be printed out in source code form with message numbers for reference.

CREATING TALKING ADVENTURES

Using this software package along with SPECTRUM PROJECTS' VOICE PACK with its associated software you can create talking adventure games.

When you are creating your adventure you may be using unusual words and names that may not sound quite right when they are spoken. To make the pronunciation correct you may have to add these words to the dictionary that the TRNSLATE program refers to for special phonemes.

An example of a problem word is the name XERXES (pronounced ZERK SEES). Just about any word can be defined to sound like it should by using the phonemes to create strings of sounds that, when run together, say a word properly.

Use the 32K version of TRNSLATE for talking adventures.

ALLRAM/BIN

This is a short machine language program that is public domain and is included for free. It turns on your 64K RAM and copies the ROMs to RAM. All you do is LOADM "ALLRAM/BIN" and EXECecute it.

This program must be used to compile or run talking adventures. Due to the way the 64K mode works, if you hit the RESET button the mode will revert to 32K ROM mode and therefore make the data contained in the upper memory inaccessible. WHILE RUNNING A 64K TALKING ADVENTURE NEVER HIT THE RESET KEY!!!

ADVENTURE COMPILER

Once you have completed all aspects of the adventure, you will return to the main menu and will be ready to compile your entire adventure into machine language format.

Choosing Item 6 from the main menu (ADVENTURE COMPILER) will start the process.

Before the adventure can be compiled, you will be asked to supply some additional information as explained below:

The program will ask for the name of the adventure and when the process is completed, the adventure will carry this name along with the file extension "/BIN".

You are also required to indicate the room number in which you want the adventure to start.

The program also needs to know the maximum number of objects the player may carry in inventory at any one time as well as the total number of objects in your adventure.

Finally, you will be asked whether you want the adventure to be a silent or talking one.

Once the above data is entered, the program will start compiling all of the aspects of the adventure. The disks will start and stop running throughout the process, so don't worry about that; also, depending on the length of the adventure, this process may take a few minutes.

INSTRUCTIONS FOR RUNNING A COMPILED ADVENTURE

FOR SILENT VERSION

Power down your computer and insert the disk with the compiled adventure, then type LOADM "ADVNAME" and EXECute the program. ADVNAME is used as an example here. If your adventure is called DRAGON then you would LOADM "DRAGON" and EXECute it.

The program will begin in the room that you specified.

FOR TALKING VERSIONS

Turn the computer on from a cold start.

LOADM and EXEC "ALLRAM/BIN"

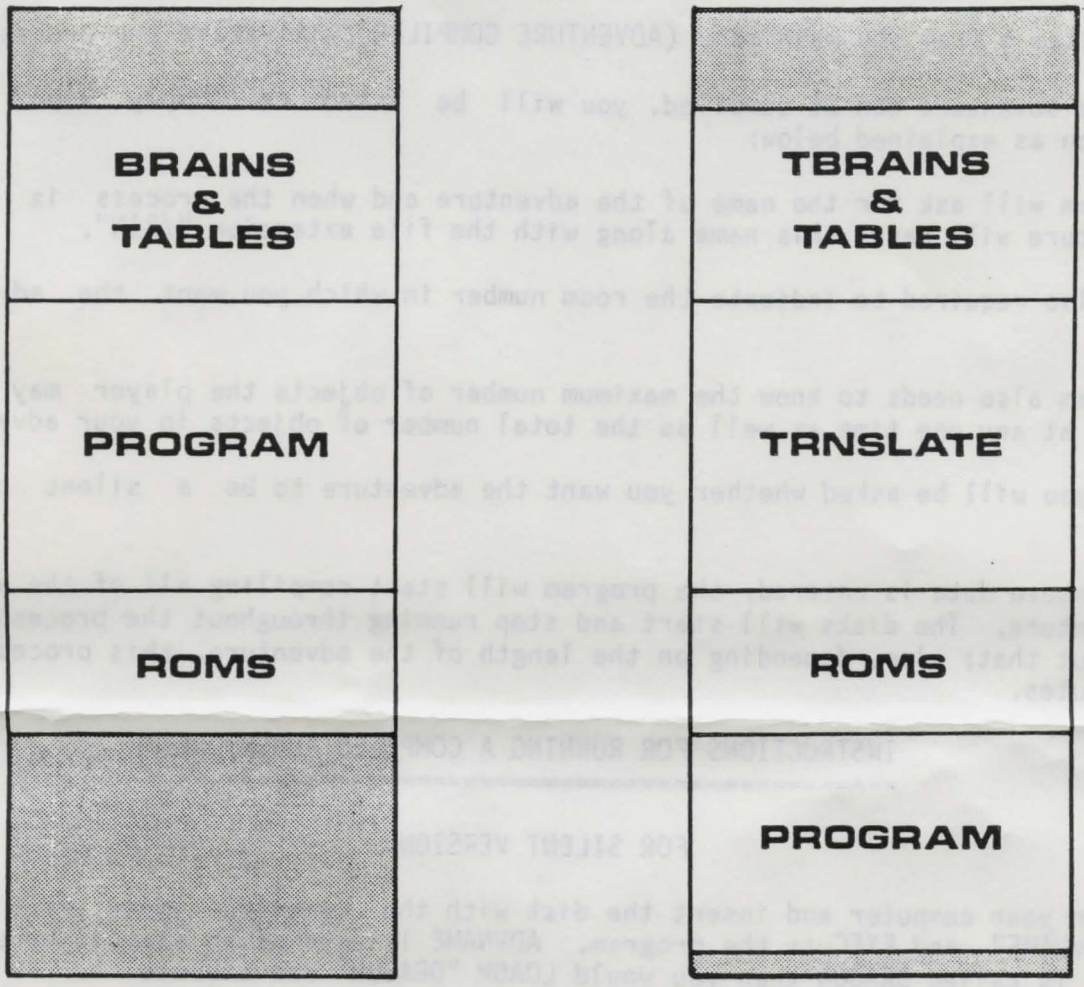
POKE 25,14:POKE 3584,0:NEW

LOADM "TRNSLATE"
This loads the VOICE PACK SOFTWARE.

LOADM "ADVNAME/PRG"

LOADM "ADVNAME/BIN"

EXEC



32K SILENT

64K TALKING

"HOUSE" -- A SAMPLE ADVENTURE

On this program disk you will see several files with the name HOUSE. These are source code files for a complete adventure that you may analyze and compile so you will be familiar with the operation of SAG.

This adventure has 9 verbs, 6 rooms, 25 objects and 13 program lines. Elsewhere in this manual is a printout of all the source code files and a map of the physical layout of the adventure. You will need to refer to this as we step through the programming. You may consider formatting a new disk and typing in all the content of the source code printouts just to get the hang of entering data into the editors.

To be able to follow the construction of the program lines I will have to spill the beans and tell you what it is all about.

Your uncle has died and willed you his treasure that is contained somewhere inside his house. You have to find it to get it. No lawyers or probate courts can do it for you.

There are 5 normal rooms and one hidden room with a door in the kitchen that cannot be seen until you PULL HOOK. When you do this, a panel will slide back and reveal a doorway. You ENTER DOORWAY and find yourself in the secret vault room with the treasure chest in it. A guardian snake will not let you get or open the chest unless you first SHOOT SNAKE with the musket in room 4 that will blow him to smithereens and remove the snake from the game. Then, and only then, can you open the chest and get the gold.

The first 9 program lines deal with movement to and from rooms. Note that the movement is unobstructed when travelling in rooms 1-5. There is a room 6 but movement to that room is obstructed until some other things are done first.

Lets pick apart program line 5 which is typical of lines 1-9. It says:

r003v001n004=e002

Translated, it means: if the player is in room 3 (r003) and the verb entered is GO (v001) and the noun entered is WEST (n004) then (=) enter room number 2 (e002)

Now, analyze lines 1-4 and 6-9 which are also movement program lines and figure out what they mean. The verb will always be verb 1 (GO) and the noun will be noun 1-6 which are NORTH, SOUTH, EAST, WEST, UP and DOWN.

These lines contain everything that is needed to move from room to room without obstruction. You are not even aware that room 6 even exists until you do the right thing!

Program line 10 is a little more complicated:

r005v009n002=s001j011p017

This first half of the line before the equal sign means that if while in room 5 (r005) you enter the verb PULL (v009) and the noun HOOK (n022) then do the output commands (=)

The second half following the equal sign are output commands that sets flag 1 to a value of 1 (s001). Flag 1 is used to tell if the secret door has opened yet. Pulling the hook caused the door to open. Before this line was executed, flag 1 was preset to

a value of zero (0). This command sets the flag to indicate that the door is now open.

The next output command (j011) makes an object that was previously invisible, visible. In this case it is the doorway that now can be seen because you pulled the hook.

The next and final output command of line 10 says print message 17 from the P message buffer that is "A PANEL SLIDES OPEN TO REVEAL A HIDDEN DOORWAY".

Now, let's look at program line 11 which reads:

r005v008n001f001=e006

This line means if while in room 5 (r005) you enter ENTER (v008) DOORWAY (n011) and flag 1 is set, then enter room 6. Remember that flag 1 was set to 1 by line 10 to indicate that the door has been opened. This is the only way to get into room 6.

Before you enter room 6 from from 5, if you EXAMINE ROOM, you will see the only obvious direction to go is EAST but you will also have the doorway to enter. This is an example of how to block and unblock passages and make the passage conditional on any number of events or situations having been played out.

You could have a magic ring that will transport you to some distant place (another room) by entering RUB RING or INCANT RING. The program interpreter doesn't care how outlandish you get, only that you tell it what you want to do in terms that it can understand.

Now for program line 12, which is the hairriest of them all. It says:

r006v005n023f002i018=t002u023g024g025p018

This line is the key to the whole adventure that allows you get the treasure chest which by now you have found in room 6, but that snake is keeping you from getting it.

While in room 6 (r006), you enter SHOOT (v005) SNAKE (n023) and flag 2 is set to 1 (SNAKE ALIVE) and you have the musket to shoot him with (i018) then do output commands (=).

The output commands mean set flag 2 (t002) to zero (snake is dead), make the snake disappear (u023), make objects 24 and 25 so they can now be gotten (g024g025), and print message 18 from the P buffer (P018) which says, "THE GUNSHOT BLASTS THE SNAKE INTO A MILLION PIECES!!"

The last program line is number 13 that says

r006v006n0241002=p019k001

In room 6 you enter OPEN (v006) CHEST (n024) and if flag 2 is set to zero (1002) which means the snake is dead, then print message 19 from buffer P which says "YOU HAVE FOUND YOUR UNCLE'S TREASURE THAT HE WILLED TO YOU IF YOU COULD FIND IT. CONGRATULATIONS!! YOU HAVE WON THIS ADVENTURE!!"

Then k001 means to kill the game. This command must be the last command used on any program line.

The number following the k command is not important as it is used as dummy data. You could say k255 or k001 or any number between 001 and 255; it doesn't matter.

All other commands may be used in any combination or sequence as long as you don't put input commands after the = sign or output commands before it.

As you can see, this program is very short with only 13 program lines, yet it is enough to make a moderately challenging adventure. Make sure you understand the concepts involved in the programming language. In the program called SAGLANG that you enter program lines into you will see occupying most of the lower screen is a list of available commands imbedded in a word. These words will help you to conceptualize what is happenning.

When you are creating your adventure, take your time. Don't try to crank out a complete adventure the first sitting. Software houses that sell very complex and challenging adventure games may spend hundreds or hours before it is just right. You can always compile an adventure and play it. If there's something you don't like or want to add or change it's very easy to use the editors to change the source code files, recompile it and refine it until you are happy just like you would with a BASIC program.

ADVENTURE PROGRAMMING METAPHYSICS

To create an adventure using this software package, you will have to operate on three levels or worlds of reality.

The first level is the fantasy world of the adventure itself where anything can happen and probably will. In this world there are no rules or limits but that of your imagination and this manual cannot help you with that part. Create your world, divide it into rooms and populate it with whatever you want. Have a good idea of the plot and goal of the adventure and define what you want to happen in reaction to the player's VERB NOUN input.

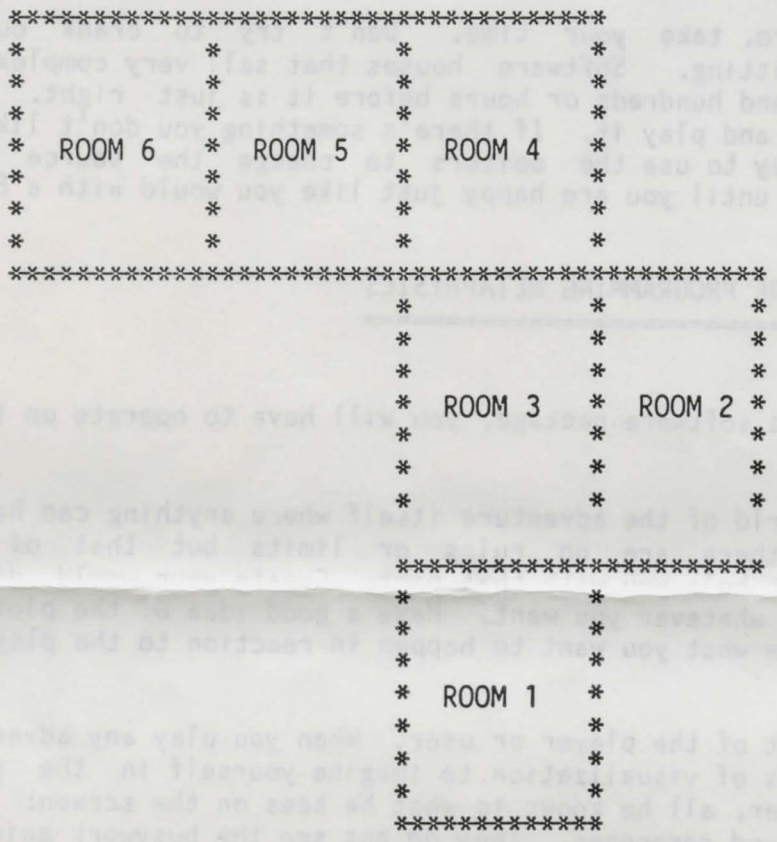
The second level or reality is that of the player or user. When you play any adventure you are supposed to use your powers of visualization to imagine yourself in the scene that is transpiring. To the player, all he knows is what he sees on the screen: room descriptions, objects, directions and responses. They do not see the busywork going on within the program setting and resetting flags, checking conditions, etc.

Try to use colorful, descriptive language that, like poetry, can invoke images in the player's mind. This requires a decent vocabulary and some creativity. Most users of this software package will probably be hard-line adventure addicts who have played quite a few adventures and through exposure to them will have taught themselves some of the possible embellishments and techniques used. This level is not to be taken lightly, especially once you start doing the programming required.

The third level of reality becomes apparent once you begin to translate your story idea into a full blown adventure and watch it bridge the gap between the two previously mentioned levels of reality.

This world involves using the programming language to convert your story into a running adventure using all the tricks and techniques inherent in the language. Think of yourself as a movie director who must do everything himself. He has to build the sets (define the rooms), build and install the props (define objects), rig the special effects and scenarios (use the general condition flags, object related flags and output commands) and direct each scene (define the text output to the screen), and lastly, to film it (compile the adventure).

HOUSE ADVENTURE MAP



/ERB LIST

VERB NUMBER 1 - GO
VERB NUMBER 2 - GET
VERB NUMBER 3 - DROP
VERB NUMBER 4 - EXAMINE
VERB NUMBER 5 - SHOOT
VERB NUMBER 6 - OPEN
VERB NUMBER 7 - FEED
VERB NUMBER 8 - ENTER
VERB NUMBER 9 - PULL

ROOM LIST

ROOM	EXIT CODES	ROOM DESCRIPTION
1	1 N	- I AM STANDING ON THE FRONT PORCH OF MY UNCLE'S HOUSE
2	7 NSE	- I AM IN THE HALLWAY
3	8 W	- I AM IN THE LIBRARY
4	10 SW	- I AM IN THE DINING ROOM
5	4 E	- I AM IN THE KITCHEN
6	4 E	- I AM IN MY UNCLE'S SECRET VAULT

OBJECT LIST

NO.	ROOM	GET?	SEE?	MSG#	ABREV.	OBJECT DESCRIPTION
1	0	NO	NO	0	NORT	NORTH
2	0	NO	NO	0	SOUT	SOUTH
3	0	NO	NO	0	EAST	EAST
4	0	NO	NO	0	WEST	WEST
5	0	NO	NO	0	UP	UP
6	0	NO	NO	0	DOWN	DOWN
7	0	NO	NO	0	ROOM	ROOM
8	0	NO	NO	0	INVE	INVENTORY
9	1	NO	YES	1	SWIN	A PORCH SWING
10	1	YES	YES	2	MILK	A MILKCAN
11	5	NO	YES	14	DOOR	A HIDDEN DOORWAY
12	2	YES	YES	4	PLAQ	AN IMPRESSIVE LOOKING PLAQUE
13	2	YES	YES	5	PAIN	A PAINTING
14	3	YES	YES	6	JACK	YOUR UNCLE'S SMOKING JACKET
15	3	NO	YES	7	CASES	BOOKCASES
16	3	NO	YES	0	DESK	A READING DESK
17	4	NO	YES	8	TABL	A CLAW FOOT TABLE
18	4	YES	YES	9	MUSK	A MUSKET
19	5	NO	YES	10	STOV	A STOVE

20	5	NO	YES	11	REFR	A REFRIGERATOR
21	5	YES	YES	12	FOOD	CANS OF FOOD
22	5	NO	YES	0	HOOK	A COAT HOOK
23	6	NO	YES	13	SNAK	A GUARDIAN SNAKE
24	6	NO	YES	15	CHES	A TREASURE CHEST
25	6	NO	NO	16	GOLD	GOLD BARS

NO. 1 - IT'S AN OLD FASHIONED PORCH SWING SUPPORTED BY RUSTY CHAINS

NO. 2 - WHAT A PIECE OF JUNK!!

NO. 3 - XXXX

NO. 4 - IT'S FROM THE ASSOCIATION OF DEMENTED ARCHITECTS AND PROCLAIMS YOUR UNCLE AS A

NO. 5 - IT'S A PAINTING OF YOUR UNCLE IN INDIA CHARMING A SNAKE

NO. 6 - THE JACKET IS MADE OF CHINESE SILK WITH DRAGONS ON THE BACK

NO. 7 - THE BOOKCASE IS FILLED WITH BOOKS ON THE OCCULT

NO. 8 - IT MUST BE OVER 100 YEARS OLD

NO. 9 - THE MUSKET IS LOADED AND READY TO SHOOT

NO. 10 - LOOKS LIKE A WOOD BURNING STOVE

NO. 11 - IT SAYS 'MADE BY KENMORE'

NO. 12 - JUST SPAM AND PORK AND BEANS

NO. 13 - THIS SNAKE LOOKS LIKE HE BITES FIRST AND ASKS QUESTIONS LATER

NO. 14 - IT LEADS TO HIDDEN CHAMBER

NO. 15 - IT'S YOUR UNCLE'S TREASURE CHEST

NO. 16 - THE GOLD BARS HAVE STRANGE ARABIC WRITING BUT THEY SURE DO SPARKLE!

NO. 17 - A PANEL SLIDES OPEN TO REVEAL A HIDDEN DOORWAY

NO. 18 - THE GUNSHOT BLASTS THE SNAKE INTO A MILLION PIECES!!

NO. 19 - YOU HAVE FOUND YOUR UNCLE'S TREASURE THAT HE WILLED TO YOU IF YOU COULD FIND IT
YOU HAVE WON THIS ADVENTURE!!

FLAG DESCRIPTIONS AND STATUS PRINTOUT

NO.	STATUS	DESCRIPTION:
1	0	DOOR OPEN
2	1	SNAKE ALIVE
3	0	
4	0	
5	0	

HOUSE ADVENTURE PROGRAM

```
 1 r001v001n001=e002
 2 r002v001n001=e004
 3 r002v001n002=e001
 4 r002v001n003=e003
 5 r003v001n004=e002
 6 r004v001n002=e002
 7 r004v001n004=e005
 8 r005v001n004=e006
 9 r005v001n003=e004
 10 r006v001n003=e005
 11 r005v009n022=s001j011p017
 12 r005v008n011f001=e006
 13 r006v005n023f002i018=t002u023g024g025p018
 14 r006v006n0241002=p019k001
```

STEP NO. 1 - r001v001n001=e002

IF: I AM STANDING ON THE FRONT PORCH OF MY UNCLE'S HOUSE / VERB IS: GO / OBJECT IS: NORTH /

THEN: I AM IN THE HALLWAY /

STEP NO. 2 - r002v001n001=e004

IF: I AM IN THE HALLWAY / VERB IS: GO / OBJECT IS: NORTH /

THEN: I AM IN THE DINING ROOM /

STEP NO. 3 - r002v001n002=e001

IF: I AM IN THE HALLWAY / VERB IS: GO / OBJECT IS: SOUTH /

THEN: I AM STANDING ON THE FRONT PORCH OF MY UNCLE'S HOUSE /

STEP NO. 4 - r002v001n003=e003

IF: I AM IN THE HALLWAY / VERB IS: GO / OBJECT IS: EAST /

THEN: I AM IN THE LIBRARY /

STEP NO. 5 - r003v001n004=e002

IF: I AM IN THE LIBRARY / VERB IS: GO / OBJECT IS: WEST /

THEN: I AM IN THE HALLWAY /

STEP NO. 6 - r004v001n002=e002

IF: I AM IN THE DINING ROOM / VERB IS: GO / OBJECT IS: SOUTH /

THEN: I AM IN THE HALLWAY /

STEP NO. 7 - r004v001n004=e005

IF: I AM IN THE DINING ROOM / VERB IS: GO / OBJECT IS: WEST /

THEN: I AM IN THE KITCHEN /

STEP NO. 8 - r005v001n004=e006

IF: I AM IN THE KITCHEN / VERB IS: GO / OBJECT IS: WEST /

THEN: I AM IN MY UNCLE'S SECRET VAULT /

STEP NO. 9 - r005v001n003=e004

IF: I AM IN THE KITCHEN / VERB IS: GO / OBJECT IS: EAST /

THEN: I AM IN THE DINING ROOM /

STEP NO. 10 - r006v001n003=e005

IF: I AM IN MY UNCLE'S SECRET VAULT / VERB IS: GO / OBJECT IS: EAST /

THEN: I AM IN THE KITCHEN /

STEP NO. 11 - r005v009n022=s001j011p017

IF: I AM IN THE KITCHEN / VERB IS: PULL / OBJECT IS: A COAT HOOK /

THEN: SET FLAG: DOOR OPEN / / PRINT MESSAGE: A PANEL SLIDES OPEN TO REVEAL A HIDDEN DOORWAY /

STEP NO. 12 - r005v008n011f001=e006

IF: I AM IN THE KITCHEN / VERB IS: ENTER / OBJECT IS: A HIDDEN DOORWAY / FLAG SET: DOOR OPEN /

THEN: I AM IN MY UNCLE'S SECRET VAULT /

STEP NO. 13 - r006v005n023f002i018=t002u023g024g025p018

IF: I AM IN MY UNCLE'S SECRET VAULT / VERB IS: SHOOT / OBJECT IS: A GUARDIAN SNAKE / FLAG SET: SNA
/

THEN: RESET FLAG: SNAKE ALIVE / CANNOT SEE: A GUARDIAN SNAKE / CAN GET: A TREASURE CHEST / CAN GET
UNSHOT BLASTS THE SNAKE INTO A MILLION PIECES!! /

STEP NO. 14 - r006v006n0241002=p019k001

IF: I AM IN MY UNCLE'S SECRET VAULT / VERB IS: OPEN / OBJECT IS: A TREASURE CHEST / FLAG NOT SET:

THEN: PRINT MESSAGE: YOU HAVE FOUND YOUR UNCLE'S TREASURE THAT HE WILLED TO YOU IF YOU COULD
NS!!! YOU HAVE WON THIS ADVENTURE!! / GAME ENDS /

Second City Software

'your FIRST choice for CoCo software...'

SECOND CITY SOFTWARE LICENSE AGREEMENT

SCS ADVENTURE GENERATOR

If you wish to market a program utilizing the SCS ADVENTURE GENERATOR, there is a one (1) time license fee of $30.00 (Thirty U.S. Dollars) payable to Second City Software. Please send your check or money order in U.S. Dollars to:

SECOND CITY SOFTWARE
P.O. Box 72956
Roselle, IL 60172
Attn: SCS\enterprises

We are always looking for new and innovative software programs to market and sell. Please contact us for information on our sales/marketing agreement and policy. SECOND CITY SOFTWARE is committed to our software authors by entering into a signed contracted royalty agreement to ensure full payment based on sales. Don't sell your tallent or programs on empty promises or offers to-good-to-be-true... Together, we can put an end to the 'sins' of the past!

.....

PLEASE FIND PAYMENT ENCLOSED FOR ( ) LICENSE AGREEMENT(S) FROM:

NAME: _____

ADDRESS: _____

CITY, STATE, ZIP CODE: _____

DAYTIME TELEPHONE: _____

EVENING TELEPHONE: _____

P.O.Box 72956 Roselle, IL 60172
Order: 312-653-5610 BBS: 312-307-1510