Write your own Adventure Games in BASIC

- SD-8516 edition

From the Author

When I was growing up, somehow, my friend lent me his VIC-20 in exchange for my intellivision to play the new D&D game. I remember playing games like GOLF and PIRATE ADVENTURE. Then when I was 9 an uncle gave me the D&D Red Box for Christmas. Later, I got a C128 and started playing games like Temple of Apshai, Sword of Fargoal, The Wizard, Zork, the Hitchhiker's Guide to the Galaxy, and Trinity. During these formative years, I spent hours typing in programs in hexidecimal from Compute! and BYTE magazine. RUN magazine. Dr. Dobb's Journal. I read all the books in the school library about computers. Books like Usborne's “Write Your Own Adventure Programs for your Microcomputer”. I was hooked.

This is not *that* book, but it is inspired by it and all the great BASIC programming books of yesteryear. It is my sincerest wish that this book finds a place in your heart alongside the old greats like those.

Chapter I: What is an adventure game?

Types of Adventure Games

In 1974 Peter Langston wrote Wander, a kind of text adventure game creator. Then in 1975, William Crowther and Don Woods wrote “Colossal Caves”, also known as “adventure”. It's a good game.

The genre took off, and by 1978 Scott Adams had released “Pirate Adventure”. I played it on the VIC-20 when I was 5. It was a masterpiece. Later, games improved. I recall with fond memories “Ulysses and the Golden Fleece” and many similar adventures. Then the genre expanded; Alice in Wonderland, Phantasie Star, The Bard's Tale, Ultima 4. There were so many games to play!

The thing that I remember most of all, is that these works of art allowed me to become a part of a fantasy or science fiction world. The narrative is what mattered most. I was just at home in a Peter Jackson and Ian Livingstone Sword and Sorcery novel as I was in a D&D campaign. And, these kinds of games on the computer brought that experience into the real world in a way that fascinated me.

To me, that's what an adventure game really is; a story, a story that draws you in and lets you become a part of it. From Maniac Mansion to Trinity, from Ultima 4 to World of Warcraft, this is lesson #1. Your adventure program needs a strong story. It needs context – a world – it needs art. These are the fundamental building blocks of our imagination, and these factors alone will make or break your adventure game, no matter what form you choose for it to take.

How to write programs

When you first start writing a computer program, especially a BASIC program, it's good to understand

what your job is and what the computer's job is clearly. The computer is responsible for telling the story of the adventure, and this means the adventure has to be typed in to the computer. This will become a large amount of data, and whenever you have a large amount of data it pays to plan how to organize it ahead of time.

The computer's responsibilities:

- Initialize the game
- Present room descriptions
- Handle commands
- Check conditions of the game

Chapter II: The Brute Force Approach

When all else fails, just start coding.

The first style of program we can look at is known as the “Thousand Rooms” style, where each block of 1000 BASIC lines is reserved for a different room. You can have 30 to 60 rooms in an adventure like this before you run out of line numbers. You could RENUMBER, but that would destroy the structure of the line numbers.

If we plan to organize the initialization in the line numbers 10 to 900, then the rooms can each take a 1,000 block. I.E. Room 1 is 1000-1990, room 2 can take lines 2000-2990, room 3 line 3000, and on.

After this we will need subroutines to handle commands. Let's give the game ample space for rooms and begin commands at line 25000, reserving again each 1000-block for a separate command. Okay, now we know enough to plan out the program.

- Line 10: Intialization
- Line 100: Introduction
- Line 1000: Room 1
- Line 2000: Room 2
- Line 3000: Room 3
- Line 25000: Subroutine 1
- Line 26000: Subroutine 2
- et cetera

We must also plan ahead for our use of variables. Stellar basic can use single-character integer variables very quickly, so we will stick to these for our first program. Let's initialize the ones we will need ahead of time:

```
10 LET T = 0
11 LET S = 0
12 LET R = 0
13 LET K = 0
14 LET M = 0
15 LET D = 0
16 LET L = 0
```

```
    17 LET C = 0
    18 LET G = 0
    19 LET G = 0
    20 LET F = 0
```

Here,

- T is for time (ticks, or 'turns'). One player action is one turn.
- S is for score. As certain milestones are passed, the score should increase.
- R is for current room number
- K is for KEY. The status of the key.
- M is for MAT. The status of the mat.
- D is for DOOR. The status of the door.
- L is for LAMP. The status of the lamp or light.
- C is for cheese. You have to find the cheese.
- G is for ghost.
- W is for water.
- F is for fridge.

This will do for our example game. In a real game, we might use many dozens of variables. The G/F/W is for an in-game puzzle. Before you can get to the treasure you have to take the water from the fridge and throw it on the floor so the ghost can't get you. Something like that.

In Stellar Basic, uninitialized variables return zero, so we do not need to initialize them in a technical sense. But declaring them at the start of a program is good practice so you know which variables will be used in your program. Do not overlook this discipline of programming because it appears simple.

Where to start

Let's start small, with a typical text adventure game. These are the easiest of all games to make, and if you start here with me, I will take you on a journey unlike any other! Let's begin on a country road, in a forest. It's a cool summer's day.

```
100 CLS
110 PRINT "Welcome to the Adventure!"
120 PRINT "Would you like instructions (Y/N)? "
130 LET A = GETKEY()
140 IF A = ASC("Y") THEN GOTO 200
150 IF A = ASC("y") THEN GOTO 200
160 IF A = ASC("N") THEN GOTO 500
170 IF A = ASC("n") THEN GOTO 500
180 GOTO 130

200 REM INSTRUCTOINS
210 PRINT "I will be your eyes and ears."
220 PRINT "Direct me with simple commands,"
230 PRINT "such as GO WEST (or WEST) to move,"
240 PRINT "TAKE LAMP, or EAT FOOD."
```

```
250 PRINT "You can type SCORE, INV, or QUIT."
260 PRINT "If you need help, type HELP."
300 PRINT ""
310 PRINT "Story:"
320 PRINT "Somewhere along this quiet rural route"
330 PRINT "lies an old farmhouse with a reputation"
340 PRINT "for forgotten secrets."
350 PRINT "Most weekend hikers pass it by without"
360 PRINT "a second glance. But on certain days,"
370 PRINT "when the light shifts just right, the"
380 PRINT "place seems to... wait."
390 PRINT "One sunny day, you are on a relaxing"
400 PRINT "stroll down an old country road to lose"
410 PRINT "the pressure of city life. Suddenly you"
420 PRINT "find yourself looking for shelter in"
430 PRINT "a coming rainstorm. You never expected"
440 PRINT "to find this house. It, however, *was*"
450 PRINT "expecting you..."
460 PRINT ""
470 INPUT "Press ENTER to continue:", A$

500 CLS
510 PRINT "Rural Route Bend"
520 PRINT "You are on a peaceful hike along an old"
530 PRINT "country road, enjoying a stress-free"
540 PRINT "weekend escape from the city. The day"
550 PRINT "started bright and sunny, but a large"
560 PRINT "cloud has drifted overhead, offering"
570 PRINT "welcome shade."
580 PRINT ""
1000 REM COUNTRY ROAD
1010 PRINT "As you round the bend, a long gravel"
1020 PRINT "driveway with a white picket fence"
1030 PRINT "and a field on the left winds westward"
1040 PRINT "towards a barn and an old one-story"
1050 PRINT "farmhouse. Its paint is faded, but it"
1060 PRINT "has the comfortable look of a long-"
1070 PRINT "loved family home."
1080 PRINT "It appears to be abandoned."
1090 PRINT ""
1100 INPUT "> ", A$
1110 PRINT "You typed: ", A$
```

This is the start of our game. Although it may seem like nothing special is going on here, this is the most important part of the story. A good start will increase the player's experience of the game.

Therefore, when asking the question “Where to start?” the basic answer is, you should have a strong story in mind and you should begin by describing the starting location in your game.

The Map

When you're just starting out in your programming journey, you should have a strong story in mind and begin by describing your first location. But you should also have a map. A map will help you think about the world you are creating and establsh the bounds of the adventure. Don't start writing a boundless game. There must be clearly defined limits to the adventure. This is not a bad thing, it's just good storytelling.

Also, every good map needs an “X”. X marks the spot. Don't consider this as an actual spot on the map – although it can be – consider it a goal. What are the goals the player is trying to achieve? While coming up with a plot is beyond the scope of this book, here are a few examples:

- Area Progression
  - In “Area Progression”, the map is sectioned off and each area requires solving a puzzle or finding a key to “progress” through the game.
- Open World
  - In “Open World”, the map is generally accessible from the start, and exploring it and solving quests and puzzles will help you progress in general.

The game “Trinity” is an area progression game. The game “Zork” is an open world game (more or less) which requires you to find and retrieve treasures from across the world. Of course, the idea of a “score” is some metric by which you can judge completion of a game. But it can be difficult to assign a credible score in a traditional sense without a zork-like treasure hunt.

Another idea?

- Quest progression
  - In this, you must solve an ever increasing series of quests which lead to a grand over-arching quest such as “fighting a boss”.

A lot of games work as a combination of these elements, but they are the basic elements.

For our farmhouse game, let's define the first area of the game. We will have the bend in the road, and we will have a location north of it and south of it. If the player travels south, they will be on the main highway and they will not want to progress in that direction. Traveling north takes them to a windy forest road, covered in leaves, clearly headed nowhere. We will also cause it to start raining by this point (a mechanic we will introduce later) to incite the player to seek refuge inside the farmhouse.

```
   R2
  /
R1
|
B
|
H
```

We start at “B”. The player can go north to R1 and then again but R2 will be written in as a dead end.

Handling commands

For now, each room can handle it's own commands. Let's start with room 1 (B), which we define as the 1000-block. Instead of line 1110 saying “you typed…” lets change it:

```
1110 IF A$ = "north" THEN GOTO 2000
1120 IF A$ = "n" THEN GOTO 2000
1130 IF A$ = "south" THEN GOTO 3000
1140 IF A$ = "s" THEN GOTO 3000
1150 IF A$ = "look" THEN GOTO 1000
1160 IF A$ = "l" THEN GOTO 1000
1150 PRINT "I can't understand your command."
1160 GOTO 1100
```

This block sets up the idea that you can type N or S to go north or south.

Next, let's write those other two locations (and R2) and hook them up with N or S commands as well:

The Highway

```
10 LET T = 0
1000 PRINT "Rural Route"
1005 LET T = T + 1

3200 PRINT ""
3210 INPUT "> ", A$
3220 IF A$ = "north" THEN GOTO 1000
3230 IF A$ = "n" THEN GOTO 1000
3240 IF A$ = "look" THEN GOTO 3000
3250 IF A$ = "l" THEN GOTO 3000
3260 PRINT "I can't understand your command."
3270 GOTO 3100
3000 PRINT "The Highway"
3010 LET T = T + 1
3020 PRINT "You walk back towards the highway."
3030 PRINT "There aren't many cars, but something"
3040 PRINT "about them, the smell of gasoline,"
3050 PRINT "The busyness of it all, turns you off."
3060 PRINT "You've been down that road before. You"
3070  PRINT "know where it goes. 'Not this time,'"
3080 PRINT "you tell yourself, 'I have something"
3090 PRINT "else I need to be doing.' You smile"
3100 PRINT "to yourself and turn back towards the"
3110 PRINT "north. The fresh air, the trees,"
3120 PRINT "the open country road. What more could"
```

```
    3130 PRINT "anyone need for a weekend retreat?"
    3140 IF T > 5 THEN IF T < 15 THEN PRINT "You feel a few drops of rain."
    1085 IF T > 5 THEN IF T < 15 THEN PRINT "You feel a few drops of rain."
```

Here we introduce the concept of a variable. A variable lets us change how things work in the world and helps us make it come alive. Here, T represents the passing of time. We can introduce certain events. In this case, if T (or TURNS) is greater than 5, the player will see a message that it is starting to rain. but, if it's 15 turns, the message will go away.

Old Forest Road

```
2000 PRINT "Old Forest Road"
2010 LET T = T + 1
2020 PRINT "As you walk north into the forest,"
2030 PRINT "Things start to get quieter."
2040 PRINT "There are a lot of leaves on the ground"
2050 PRINT "this fall. The crunch of your steps"
2060 PRINT "on the gravel echoes over the trees."
2110 PRINT "This is rather long road, and you're"
2120 PRINT "not sure where it leads."
2130 IF T > 7 THEN IF T < 17 PRINT "You feel a few drops of rain."
2200 PRINT ""
2210 INPUT "> ", A$

2220 IF A$ = "north" THEN GOTO 2000
2230 IF A$ = "n" THEN GOTO 2000
2220 IF A$ = "south" THEN GOTO 1000
2230 IF A$ = "s" THEN GOTO 1000
2240 IF A$ = "look" THEN GOTO 2000
2250 IF A$ = "l" THEN GOTO 2000
2260 PRINT "I can't understand your command."
2270 GOTO 2200
```

The old forest road represents a “dead end”. If the player keeps going north, the discouraging message will make this location seem less interesting. This kind of “funneling” is can make a game feel linear but in this case it is an extra location for the player to look at instead of going directly to the farmhouse. In that sense, it has some use in the game.

Country Road Complete Program Listing

```
10 LET T = 0
11 LET S = 0
12 LET R = 0
13 LET K = 0
14 LET M = 0
```

```
15 LET D = 0
16 LET L = 0
17 LET C = 0
18 LET G = 0
19 LET G = 0
20 LET F = 0

100 CLS
110 PRINT "Welcome to the Adventure!"
120 PRINT "Would you like instructions (Y/N)? "
130 LET A = GETKEY()
140 IF A = ASC("Y") THEN GOTO 200
150 IF A = ASC("y") THEN GOTO 200
160 IF A = ASC("N") THEN GOTO 500
170 IF A = ASC("n") THEN GOTO 500
180 GOTO 130

200 REM INSTRUCTIONS
210 PRINT "I will be your eyes and ears."
220 PRINT "Direct me with simple commands,"
230 PRINT "such as GO WEST (or WEST) to move,"
240 PRINT "TAKE LAMP, or EAT FOOD."
250 PRINT "You can type SCORE, INV, or QUIT."
260 PRINT "If you need help, type HELP."
300 PRINT ""
310 PRINT "Story:"
320 PRINT "Somewhere along this quiet rural route"
330 PRINT "lies an old farmhouse with a reputation"
340 PRINT "for forgotten secrets."
350 PRINT "Most weekend hikers pass it by without"
360 PRINT "a second glance. But on certain days,"
370 PRINT "when the light shifts just right, the"
380 PRINT "place seems to... wait."
390 PRINT "One sunny day, you are on a relaxing"
400 PRINT "stroll down an old country road to lose"
410 PRINT "the pressure of city life. Suddenly you"
420 PRINT "find yourself looking for shelter in"
430 PRINT "a coming rainstorm. You never expected"
440 PRINT "to find this house. It, however, *was*"
450 PRINT "expecting you..."
460 PRINT ""
470 INPUT "Press ENTER to continue:", A$

500 CLS
510 PRINT "Rural Route Bend"
520 PRINT "You are on a peaceful hike along an old"
530 PRINT "country road, enjoying a stress-free"
540 PRINT "weekend escape from the city. The day"
550 PRINT "started bright and sunny, but a large"
```

```
560 PRINT "cloud has drifted overhead, offering"
570 PRINT "welcome shade."
580 PRINT ""
1000 PRINT "Rural Route"
1005 LET T = T + 1
1010 PRINT "As you round the bend, a long gravel"
1020 PRINT "driveway with a white picket fence"
1030 PRINT "and a field on the left winds westward"
1040 PRINT "towards a barn and an old one-story"
1050 PRINT "farmhouse. Its paint is faded, but it"
1060 PRINT "has the comfortable look of a long-"
1070 PRINT "loved family home."
1080 PRINT "It appears to be abandoned."
1085 IF T > 5 THEN IF T < 15 PRINT "You feel a few drops of rain."
1090 PRINT ""
1100 INPUT "> ", A$
1110 IF A$ = "north" THEN GOTO 2000
1120 IF A$ = "n" THEN GOTO 2000
1130 IF A$ = "south" THEN GOTO 3000
1140 IF A$ = "s" THEN GOTO 3000
1150 IF A$ = "look" THEN GOTO 1000
1160 IF A$ = "l" THEN GOTO 1000
1150 PRINT "I can't understand your command."
1160 GOTO 1100


2000 PRINT "Old Forest Road"
2010 LET T = T + 1
2020 PRINT "As you walk north into the forest,"
2030 PRINT "Things start to get quieter."
2040 PRINT "There are a lot of leaves on the ground"
2050 PRINT "this fall. The crunch of your steps"
2060 PRINT "on the gravel echoes over the trees."
2110 PRINT "This is rather long road, and you're"
2120 PRINT "not sure where it leads."
2130 IF T > 5 THEN IF T < 15 THEN PRINT "You feel a few drops of rain."
2200 PRINT ""
2210 INPUT "> ", A$
2220 IF A$ = "north" THEN GOTO 2000
2230 IF A$ = "n" THEN GOTO 2000
2220 IF A$ = "south" THEN GOTO 1000
2230 IF A$ = "s" THEN GOTO 1000
2240 IF A$ = "look" THEN GOTO 2000
2250 IF A$ = "l" THEN GOTO 2000
2260 PRINT "I can't understand your command."
2270 GOTO 2200

3000 PRINT "The Highway"
3010 LET T = T + 1
3020 PRINT "You walk back towards the highway."
3030 PRINT "There aren't many cars, but something"
```

```
3040 PRINT "about them, the smell of gasoline,"
3050 PRINT "The busyness of it all, turns you off."
3060 PRINT "You've been down that road before. You"
3070 PRINT "know where it goes. 'Not this time,'"
3080 PRINT "you tell yourself, 'I have something"
3090 PRINT "else I need to be doing.' You smile"
3100 PRINT "to yourself and turn back towards the"
3110 PRINT "north. The fresh air, the trees,"
3120 PRINT "the open country road. What more could"
3130 PRINT "anyone need for a weekend retreat?"
3140 IF T > 5 THEN IF T < 15 THEN PRINT "You feel a few drops of rain."
3200 PRINT ""
3210 INPUT "> ", A$
3220 IF A$ = "north" THEN GOTO 1000
3230 IF A$ = "n" THEN GOTO 1000
3240 IF A$ = "look" THEN GOTO 3000
3250 IF A$ = "l" THEN GOTO 3000
3260 PRINT "I can't understand your command."
3270 GOTO 3200
```

Chapter III: Parallel Arrays

Just like Pirate Adventure!

The next technique I will show you is the 'Parallel Arrays' technique. This is the canonical approach from the late 70s/early 80s Scott Adams games. It's a DATA-driven format where rooms, objects, and actions are stored as numbered data rather than hard-coded logic. The key patterns:

- Room data: description string
- Room exits: N/S/E/W/U/D exit pointers (0 = no exit) stored in DATA statements and read into arrays at startup.
- Object data: name, description, current location (room number, 0 = carried, -1 = nowhere). OB$(i) for name, OL(i) for location.
- Inventory: just scan the object array for items where OL(i)=0 (0 = player's inventory).

```
1 -- 2 -- 3
     |    |
     4 -- 5
     |
     6 -- 7 -- 8
```

The rooms will be connected in the above order.

```
10 DIM N(10)
20 DIM E(10)
30 DIM W(10)
40 DIM S(10)
```

The above defines the map data structure; now let's link rooms. We will start by linking the EAST from room 1 to go to room 2. Then we will define WEST, SOUTH and EAST for room 2 to go to 1, 4 and 3, etc:

```
50 LET E(1) = 2
60 LET W(2) = 1
70 LET S(2) = 4
80 LET E(2) = 3
90 LET W(3) = 2
100 LET S(3) = 5
110 LET N(4) = 2
120 LET E(4) = 5
130 LET S(4) = 6
140 LET W(5) = 4
150 LET N(5) = 3
160 LET N(6) = 4
170 LET E(6) = 7
180 LET W(7) = 6
190 LET E(7) = 8
200 LET W(8) = 7
```

Now, if you want to go to a room, and R is your current room (K is key, N means the ascii for north) you can do:

```
IF K = N THEN LET R = N(R)
IF K = E THEN LET R = E(R)
IF K = W THEN LET R = W(R)
IF K = S THEN LET R = S(R)
ON R GOTO 1000,2000,3000,4000,5000,6000,7000
```

For a simple game, hard-coding rooms like this (and in Chapter 1) is fine. However, once you exceed about 10 rooms or objects, DATA/READ becomes essential because it separates content from engine and makes the game expandable without touching logic code. Here is the DATA approach:

```
10 FOR I = 1 to 5
20 READ A
30 READ B
40 READ C
50 READ D
60 LET N(I) = A
70 LET E(I) = B
80 LET S(I) = C
90 LET W(I) = D
100 NEXT I
110 DATA 0,2,0,0
120 DATA 0,3,1,4
130 DATA 0,0,2,5
140 DATA 2,5,0,6
150 DATA 3,0,4,0
160 DATA 4,7,0,0
170 DATA 0,8,6,0
```

```
180 DATA 0,0,7,0
```

If you imagine a game with 20, 30 or 40 (or more) rooms, this DATA method makes sense.

Let's use this map to write the next section of the game; starting “On the driveway” – part 2 of the “Country Road” game.

```
10 REM SET UP GAME
20 GOSUB 9000
30 MODE 2
40 CLS
50 PRINT RN$(R)
60 PRINT RD$(R)
70 INPUT A$
80 REM END GAME
90 GOTO 9900

9000 REM SETUP VARIABLES
9010 GOSUB 9100
9020 REM LOAD ROOM EXIT DATA
9030 GOSUB 9200
9030 REM SET UP ROOM DESC DATA
9040 GOSUB 10000
9095 RETURN

9100 DIM N(10)
9110 DIM E(10)
9120 DIM W(10)
9130 DIM S(10)
9140 LET T = 0
9150 LET D = 0
9160 LET K = 0
9170 LET M = 0
9180 LET R = 7

9195 RETURN

9200 FOR I = 1 to 5
9210 READ A
9220 READ B
9230 READ C
9240 READ D
9250 LET N(I) = A
9260 LET E(I) = B
9270 LET S(I) = C
9280 LET W(I) = D
9290 NEXT I
9295 RETURN
```

```
    9300 DIM RN$(10)
    9310 DIM RD$(10)
    9310 LET RN$(8) = "On the Driveway"
    9315 LET RD$(8) = "Stepping onto the driveway, you feel like you are
entering a different world.   On your left, a picket fence separates the
driveway from the field. Ahead of    you on the right is a pine tree near
some bushes and a large rock."
    9320 LET RN$(7) = "Hedges, Tree and Rock"
    9325 LET RD$(7) = "About halfway up the driveway you find a pine tree on
the other side of some bushes, with a large rock in front of it. It's a
little to large to serve as a chair, but you imagine a goat could jump on it
easily."
    9395 RETURN

    9900 REM END GAME
    9910 MODE 1
    9920 END
    10000 DATA 0,2,0,0
    10010 DATA 0,3,1,4
    10020 DATA 0,0,2,5
    10030 DATA 2,5,0,0
    10040 DATA 3,0,4,0
    10050 DATA 3,0,4,0
    10060 DATA 4,7,0,0
    10070 DATA 0,8,6,0
    10080 DATA 0,0,7,0
```

The structure of this game is a little different than the first one; we're leaning more heavily on the use of GOSUB to get important data out of the way of the start of the program. That way, people looking at the first few lines of the source code can more easily tell what the main idea of the program is.

Secondly you will notice the mode switch. I think it's a cool little way to demonstrate the capabilities of the SD-8516; each phase of the game can have a different look. Almost as if one is travelling through time. Oh! Interesting idea, maybe we can use that in the game?

Chapter IV. The Parser

Keep it simple.

The biggest mistake a potential adventure game programmer makes is assuming he needs a full-blown Infocom style ZIL or TADS engine to make an adventure game. Nothing could be further from the truth. Don't try to make the parser too smart. In BASIC, stick with two-word commands (“GET LAMP”, “OPEN DOOR”, “GO NORTH”). If someone wants to pick up a lamp they will either type TAKE LAMP or GET LAMP. Maybe – maybe, PICK UP LAMP. What else? REACH FOR LAMP? No. ATTEMPT TO ACCQUIRE LAMP? No. Don't worry about those.

Simple flag variables handle world state changes; Is the door open or closed? Is the monster alive or dead? Puzzle solved? A small array such as `F(20)` will serve for our first game. Actions check and set flags too;

```
IF F(3)=1 THEN PRINT "The door is already open.".
```

Now, we're going to assume the parser lives at line 5000. It doesn't have to, but I like the number 5000 and we need to pick a line.

We start with our assumptions:

```
5000 REM PARSER
5010 REM PLAYER COMMAND IN A$
```

The first thing we need to do is make sure everything is uppercase so we don't need to check for N and n, etc.:

```
5010 LET A$ = "hello"
5020 REM MAKE A$ UPPERCASE
5030 LET B$ = A$
5040 LET A$ = ""
5050 FOR I = 1 TO LEN(B$)
5060 LET T$ = MID$(B$,I,1)
5070 LET C = ASC(T$)
5080 IF C >= 97 THEN IF C <= 122 THEN LET C = C - 32
5090 LET A$ = A$ + CHR$(C)
5100 NEXT I
5110 PRINT A$
```

Now we can match the user's input more effectively.

The second job of the parser is to analyze the user's commands. The user is instructed to enter imperative commands, and therefore the first word on a line is always expected to be a verb (a command). The second and further words are expected to be nouns. Therefore, let us split A$ now, into V$ and N$ (verb and noun):

```
5000 LET A$ = "TAKE LAMP"
5200 LET V$ = ""
5210 LET N$ = ""
5220 LET S = 0
5230 FOR I = 1 TO LEN(A$)
5235 LET C$ = MID$(A$,I,1)
5240 LET C = ASC(C$)
5245 IF C = 32 THEN LET S = I
5250 NEXT I
5260 IF S > 0 THEN GOTO 5290
5270 LET V$ = A$
5280 GOTO 5300
5290 LET V$ = LEFT$(A$, S-1)
5295 LET N$ = RIGHT$(A$, LEN(A$)-S)
5300 PRINT "V:",V$
5310 PRINT "N:",N$
```

more to come

May 2026 update: I'm still here! I've been working on the PPU and ASU units, i.e. graphics and sound, and have left this on the back burner for now. I'll return to finish this likely in June. Stay tuned!