The craft is controlled by means of the keyboard, which varies the rate of change of thrust. If no keys are pressed the thrust automatically decays to zero. All the keys may be used - the idea is to press in the direction you want thrust - dead centre is between the H and J keys and the force applied is proportional to the weighted distance from there.

You have a limited amount of fuel and should you run out the spacecraft will fall to the surface. You will lose all your fuel if you collide with the 'edge of the moon'. The spacecraft is represented by a bell (07) or if your fuel becomes exhausted the 'flames' go out and the spacecraft becomes a lower case 'o'.

As you approach the surface, the program gives you an expanded view showing the surface in detail, including boulders (which must be avoided).

An unhealthy landing will result in a minor explosion!

The game executes at OE6A. Note that the last column of figures in the listing is the checksum and should not be entered.

SO.. WHAT ABOUT ZEAP ??

ZEAP stands for Z80 Editor Assembler Package.

OK, so what is an assembler?

An assembler is a program which is used to take the 'donkey work' out of writing machine code programs. When you are converting your tediously written mnemonics into instruction codes (and fumbling through the book, 'cos you can't remember what they are) and at the same time assigning them to addresses, you are "assembling the program". Now an assembler does this for you, not only that, but it keeps track of where you put the subroutines and workspaces, converts lines of text into ASCII, and tells you when you started making up instruction mnemonics.

Fair enough, but what does the editor bit do?

Well most programs don't work straight off, because you got something wrong, or left something out, and these errors usually mean inserting or deleting some mnemonics, which of course sods law dictates that program will either be longer or shorter. Now the editor allows you to change, insert or delete at will, and when you re-assemble, all the addresses assigned to subroutine etc. will all come out in the right places; because the assembler part keeps track of where you put them. No more ploughing your way through 2K of object code, changing all the subroutine calls, just because you left out two bytes at the start.

Well, I've seen the output of ZEAP, but I still get lost, why is that?