Probe Lander Manual

By EC Holm 2024

Introduction

Welcome to Probe Lander, a turn-based game written in BASIC computer language. 1983 was a magical year for me as it introduced me to computer games. I had friends who had the Atari 2600 and we played many games that I previously only played in the arcade. Then there were additional games like E.T. and Adventure. That was an eye-opener. It meant this was an expansive world. I also had a friend who had a console made by Phillips. He had a game that was a lunar lander. I was mesmerized by that game mainly because of my love affair with the

US space program. I received my first computer that Christmas, a 16K Sinclair ZX Spectrum. And I love that thing. I played games and learned BASIC. I also tried to recreate a couple of games to include the lunar lander game I saw. Later, I met a man who had a pocket computer, about the size of a pocket calculator. He had a lunar lander game on there but it didn't have graphics instead, it had numbers. It was still fun.

Then I moved from Europe to the US and I couldn't use my computer because it used a TV as a monitor and the US TVs operated at a different frequency. So I got myself a pocket computer and successfully programmed the lunar lander game with numbers I had previously seen. Fast forward several years, and I'm exploring emulators of retro computers. And I'm trying to recreate the lunar lander game but it eludes me. Fast forward some more years and it hits me. Many lunar lander games have been made that were graphic-based active and responsive. What if I went the other way? Sounds like a step backward. I made it a turn-based game with numbers. Seeing it run well, I realize that it resembles the experience of robotic spacecraft teams. You sit at mission control watch the numbers and issue commands and then wait for the results. Hopefully, you won't crash.

That's how this game came about. And it is all made in BASIC so I'm porting it over to different platforms.

BASIC Writing

Why did I write this in BASIC and not assembly? BASIC was the first computer language I learned and it led me to have a job as a Visual Basic 6.0 developer job developing business software. One advantage BASIC had over assembly is that it was fairly standardized, so you could port a piece of software written in BASIC from one platform to another with relative ease.

With that in mind, I decided to make this game on 6 different platforms: ZX Spectrum, ZX81, Commodore 64, Vic 20, QB64, and Bywater BASIC.

Of course, I used emulators to write these games. I used Fuse emulator for the ZX Spectrum, EightyOne for the ZX81, Vice for both Commodore 64 and Vice 20. QB64 is a modern project and app for MS Windows that's inspired by the old Microsoft QuickBASIC. Bywater BASIC is an interpreter that runs on Linux.

How To Play

I'll put specific instructions for the different platforms later in this manual. This section holds the common gameplay.

After loading the game in the platform, retro computer, or emulator, issue the RUN command.

You should see an intro screen explaining that you are sitting in mission control and that your probe is about to make a landing attempt on the moon. Hit ENTER/RETURN/New Line or any key to continue.

Then you should see telemetry data: Horizontal velocity, Vertical Velocity, Altitude, Fuel, Burn Time, etc. Much like the picture in the following figure, fig. 1

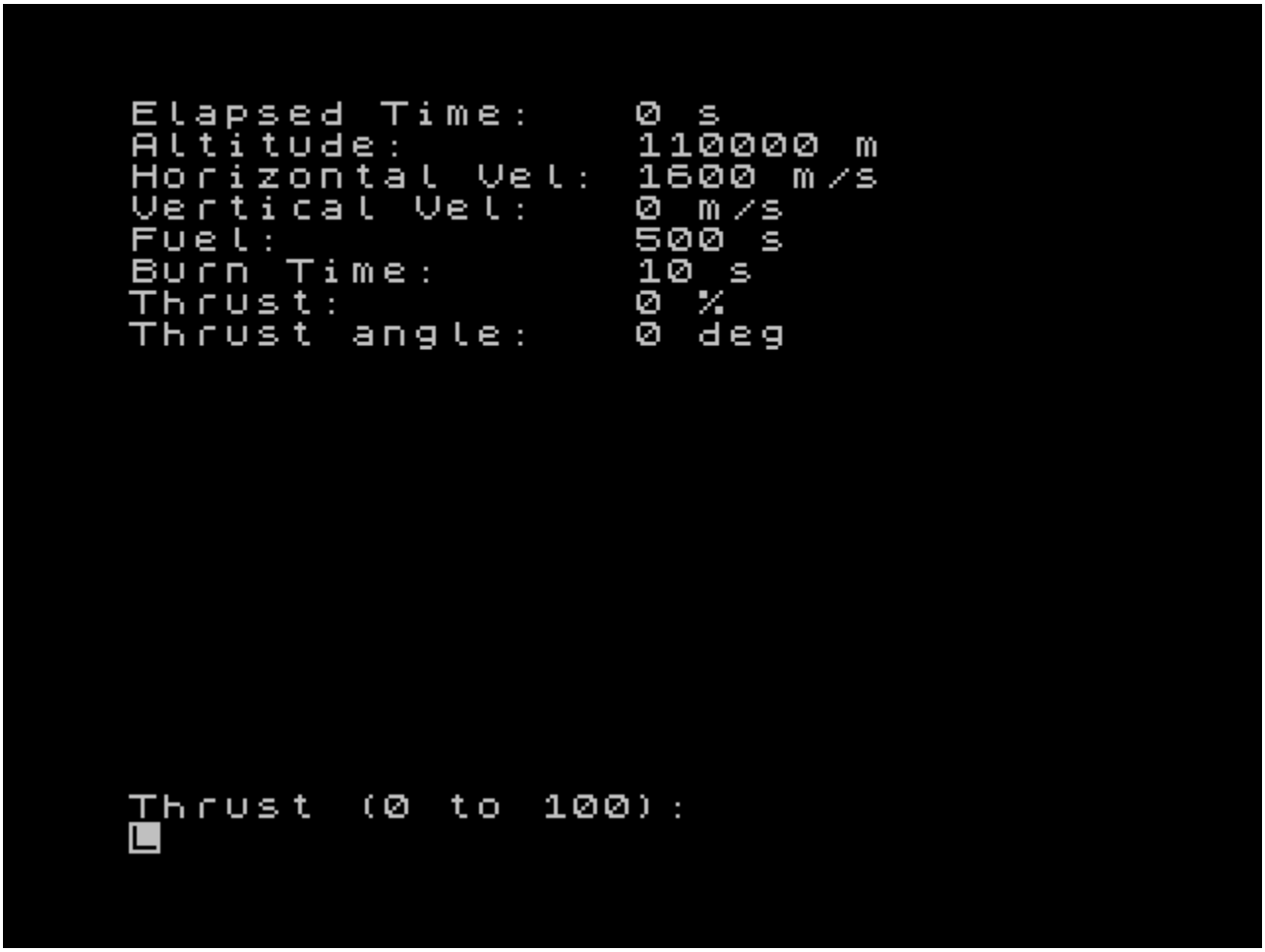

Fig. 1 telemetry screen from ZX Spectrum emulator

On every turn, you will be prompted to enter how much thrust as a percentage you want to apply, at what angle, and for how many seconds. If you put 0 thrust, Burn Time represents the time when there is no thrust.

Now, notice that can only select between -90 to 90 degrees on the angle. Your craft is orbiting the moon at a breakneck speed. You need to slow it down. You only have around 8 minutes of fuel at 100% thrust. 90 degrees represents pointing the engine nozzle horizontally toward the forward direction, thus to slow down. -90 degrees represents pointing the engine nozzle

horizontally towards the opposite direction, thus to speed up. 0 degrees represents pointing the engine nozzle down vertically to slow your descent. Remember, the lunar gravity is always acting on the spacecraft, so vertical velocity will tend to increase if you don't apply any thrust, that is inputting 0 for thrust. And you will have to do that. Knowing when to do that is all part of the game just as knowing how much thrust to apply for how much time and at what angle.

The spacecraft computer will override your command if it takes the craft in the wrong direction. For example, it will not allow you to have a negative horizontal velocity and thus change direction. But you can end up going up instead of down. That will be indicated by a minus sign on the vertical velocity.

Warning, the craft's engine is very powerful. I advise caution from using 100% of thrust until necessary.

When there is no thrust, the craft will fall because lunar gravity is acting on it. This of course depends on time. So, the vertical velocity will diminish by the multiplication of the current vertical velocity and gravity and time. Gravity in this case is a constant being one-sixth of the Earth's gravity. So, keep your eye on your current vertical velocity when choosing how much time to apply to your thrust. That's the best advice I can give you.

When you land with horizontal velocity equal to or less than 10 m/s and vertical velocity equal to or less than 5 m/s then you land successfully and landed and win the game.

The Math

I wanted to use the orbit equation but it was a little bit beyond me. So, I opted for the next best thing. I had the craft follow a spiral pattern down to the surface if no thrust was applied. The equation I used for that was:

$$verticalVelocity = verticalVelocity + (gravity * burnTime)$$

The gravity I used was 9.8/6 m/s^2 because it's the moon which is 1/6 that of the Earth's. After the vertical velocity was calculated, the craft would just fall from orbit on its own. So, I added another component to it, I multiplied a percentage to the gravity based on the horizontal velocity.

$$verticalVelocity = verticalVelocity + (gravity * (absolutevalue(horizontalVelocity / 1600 - 1))) * burnTime$$

This at least lets it sit in orbit until you apply thrust.

Of course, I made the appropriate trigonometry equations to calculate the horizontal and vertical thrust at the selected angle.

$$verticalVelocity = verticalVelocity - 0.302 * burnTime * thrust * \sin(90 - angle)$$

$$horizontalVelocity = horizontalVelocity - 0.032 * burnTime * thrust ¿ \sin(angle)$$

I derived the 0.302 from data from the Appollo LEM, specifically the decent engine. This number represents the acceleration the engine provides. And it gets multiplied by the thrust number which is chosen by the user and is also a percentage.

ZX Spectrum

For the Sinclair ZX Spectrum, I provided a .tap file. If you are a regular user of speccy games and programs, you already know how to load this file. If you don't, and you have actual hardware you can browse YouTube and find many videos on several devices and software of your choosing that will help you load .tap files on your speccy. If you are using an emulator, consult with your emulator's documentation. Most emulators will readily load .tap files. I like Fuse emulator but there are many to choose from.

The game is a simple BASIC program and you can LIST it once you LOAD "" it.

Just execute the RUN command to run it. After the intro, you are asked for numbers. If you enter some letters, the program will break, that is it will stop. You can run it again from the beginning, so not all is lost.

I also provided the listing of the code in a text file and 2 image files. Each file is named planderbasic. You can use these to type-in the program if that is your preference and want to experience that process. This program is easy to type-in. The ZX Spectrum keyboard is something you have to get used to, but once you do, it's not too bad.

Have fun and try not to crash.

ZX81

For the ZX81 I kept the native colors, black and white. I provided a .p and a .tzx files for your preference. If you have hardware, you probably know how to load such files. If you are using an emulator you use the instructions on the emulator. I like the EightyOne emulator personally.

Juse execute LOAD "" and then RUN commands to start the game.

If you want to stop the game before you land or crash, you can type letters for any input and it will stop the game.

I also provided two image files that have the code listed. They are called planderbasic. You can

use these to type-in the game if you prefer to do that and thereby learn your machine better.

Have fun and try not to crash.

Commodore 64

For the Commodore 64, I made the game as close to the ZX Spectrum version as possible, with sound and color. Of course, the Commodore behaves a little differently with its input command. It adds a question mark when it asks for the input. It took me a little time to get used to the Commodore and how it likes to edit the BASIC code and how it likes ot handle sounds. But I got the hang of it.

I provided a .d64 file with the PLANDER program on it. You can use

LOAD”*”,8 or LOAD”PLANDER”,8 to load it and then issue the RUN command. You will have to crash the spacecraft if you want to end the game early or simply reset the computer. But crashing the spacecraft allows you to get back to the BASIC code once you say N to the question PLAY AGAIN?

I also provided printouts of the code in the form of .bmp and .png images. You can use them to typ-in the game if you so choose to do so.

Have fun and try not to crash.