

Planning Your Mission to Mars, the Red Planet

[Part 1]

David J. Ritchie, Tim Winder, Craig Weber

©1998 by David J. Ritchie

[Note]

This was independently written to work with the Mars Simulation unit available from Interact, 5937 Darwin Court, Suite 106, Carlsbad, CA 92008.

Phone: (800)359-0961 Web: <http://www.teachinteract.com/>

The Mars Simulation is a copyrighted product of Interact.

They have not reviewed or endorsed this document.

**Mars Mission Science Camp
School District 203, Naperville, IL**

July 13, 1998

Summary

This is the first of five sessions in which you will use the computer programming language "perl" to:

plan your mission to Mars, the red planet.



<http://www.jpl.nasa.gov/files/images/browse/81958b.gif>

1.0 Introduction

This is the first of five sessions in which you will use the computer programming language "perl" to plan your mission to Mars.

You will have the responsibility to look at the needs of the mission in your role as either

1. a Medical Officer,
2. a Scientist,
3. a Military Officer, or
4. the Mission Commander.

You will make recommendations for the mission as to the amount of food to bring, whether or not to have artificial gravity on your voyage, the best place to land, and the type of trajectory to take from Earth to Mars and back.

Within a fixed cost, you as a team need to plan your mission so as to make the rewards (the time available for exploration and the probabilities of discoveries) as high as possible and the risks and dangers as low as practical.

Your decisions in one area will affect those that your teammates must make in other areas. Here is an example. Let's pretend the following:

1. The Medical Officer (perhaps this is you) recommends that there be artificial gravity on the space ship to reduce the health risk associated with weightlessness.
2. The Mission Commander, in order to provide artificial gravity, must change the design of the ship to one that is heavier. The rockets presently available are only so powerful. As a result, the heavier ship must take a slower path and will spend more days traveling to Mars and back.
3. The Military Officer requests more mission time be spent in solar radiation drills because the longer time in space increases the likelihood that you will encounter solar radiation from a solar flare.
4. The Scientist requests more food, water, and oxygen be placed on board because of the longer length of the mission.
5. Because the launch time to return to Earth is fixed, taking longer means that the Mission Commander will have fewer days on Mars to explore and the Scientist will have less opportunities to discover.

Or, should you take the risk, go without artificial gravity and substitute exercise instead? As a medical officer, you can figure out the risks in the medical area but you need to look at the total consequences for the mission. How can you do that?

The answer is "Let's Pretend."

1.1 Let's Pretend...

One way to determine the best decision, such as to have artificial gravity or not, is to pretend you are doing a mission to Mars and see what happens. As you pretend, you may find out that you need more food, more water, more protection from radiation, and so on, in order to make it safely to Mars and return.

You increase the food, water, radiation protection for your pretend mission, do the pretend again, discover what more you need, add it to the plan, do the pretend again, and on and on until you have accomplished your mission in pretend mode.

Then, you do the mission for real following your plan. If you have done a good job of thinking of everything that could happen and developing a good response to those things, you will have developed a good plan for your actual mission.

Then, when you do the actual mission, you will very likely be successful and make it to Mars and back.

If not--well....

1.2 "Pretend" ==> Simulate

Another word for "pretend" is "simulate." In this part of the Mars Science Camp, you will write a computer program, using the programming language "perl," to simulate your mission to Mars.

You will be assisted in this effort by the Mars Space Camp "perl" consultant. You will compare your simulation to that of your team mates and make the best decision.

Beware: You and your team mates will be betting your lives on the simulation. There are two things you have to do:

1. Make sure that the input data you give to the simulation is correct to the best of your knowledge. For this, you will rely on your research using the materials provided by this Mars Camp.
2. Make sure that the simulation is as accurate and correct as possible in portraying what will really happen. For this, you will have to rely on your "perl consultant" who wrote the simulation. But you must not just accept your consultant with blind faith. You must question him carefully about the assumptions he has made in programming the simulation and as a team decide for yourselves whether or not the results of the simulation should be believed.

These are two lessons that will be of lifelong benefit and apply for any simulation.

2.0 What is Perl?

PERL is a language that is used to give instructions to computers to make them do things, such as:

- to ask questions on the computer screen,
- to receive answers entered on the keyboard,
- to add, subtract, multiply, divide numbers to get results,
- to make other sorts of calculations based on those results,
- to print the results of those calculations

In planning your trip to Mars, you will need to calculate information for your trip, enter those into a simulation, and determine whether or not your mission will survive.

2.1 For Which Types of Computers?

Perl works for many different types of computers: unix-based computers, Windows-based computers, Macintosh-based computers, etc.

We will use perl on a Macintosh. You could also use it on a PC. In fact, the programs that work on a Mac will usually work on a PC with just a few changes.

2.2 How much?

Perl is "free-ware". It was written originally by Larry Wall and made available to the internet for free. It has been improved and extended by the suggestions of people like you and me.

3.0 Getting Perl Going on the Macintosh

Perl has been installed on the LRC Macintosh's by David Ritchie and Mr. Skrine. You should find an icon named "MacPerl". It will look like this:



MacPerl

3.1 To Begin

1. Double click on the perl icon. You will launch the perl interpreter. The perl interpreter is the program that reads the perl script that you will write and performs the script commands.
2. When the program launches, nothing much will change on the screen. Look in the upper right hand corner at the multi-finder place. You should see a camel just as in the icon. If you don't, raise your hand and I will come around and see what the problem is.
3. Explore the Menu commands for the perl interpreter. See if you can figure out what they do. We will talk about each one during our session.

3.2 To Continue

Here's how to continue:

1. Begin with the New command under the File menu. This will create a window into which you can type your lines of perl. A line of perl to type which is the traditional first line that anyone learning a new computer language composes is:

```
print ("Hello, World!\n");
```

2. When you have done a few statements, issue the Syntax Check Untitled command under the Script menu. Fix errors until you get a window which says "Syntax Ok".
3. Finally, run your first perl program by issueing the Run Untitled command under the Script menu. Your program generates a window which says:

```
Hello, World!
```

4.0 The Rules of the Language

Every language has rules. English has rules. Spanish has rules. Perl has rules.

The purpose of the rules are to make it easier for the writer to communicate with the reader. If you both use the same set of rules, then it becomes easier to understand the message.

In the case of perl, you as the person programming the computer are the writer. The computer is the reader.

Computers are very picky readers. If you fail to follow the rules, the computer will complain. When it does, it usually says something like "**Syntax Error on Line 10**" or something like that.

After you become more experienced with perl and writing scripts that computers can understand, you will begin to appreciate how easy it is to communicate with people--in some ways, even your brother, sister, or parents are likely to be far easier to communicate with than the computer.

4.1 Sentences

Ah, you thought you'd get out of lessons in proper grammar by coming to the Mars Space Camp! Wrong!

Perl scripts are composed of sentences. You know from Language Arts that a sentence has a subject, a verb, and, usually, a direct object. In addition, it ends with some sort of ending punctuation.

- **The Subject** - for perl, the subject is always "You computer".
- **The Verb** - for perl, the verb is a command that perl understands. An example of a command is "print". What do you think *that* command asks the computer to do?
- **The Direct Object** - for perl, the direct object is the thing that the command is supposed to do something to--usually in perl, the direct object will be something that holds information.
- **The Ending Punctuation** - for perl, the ending punctuation is **ALWAYS** a semi-colon. That is, every sentence must end with a ";".

That's it. Now you know how to write sentences in the language! All you need to know is the allowed verbs and the way to specify direct objects!

4.2 Non-sentences

Well, there is a little more to it. The most important non-sentence (in that it is not something whose subject is "You computer") is a comment. You can think of the subject for the comment as "You reader"--meaning the person reading the script.

A comment is anything from a "pound sign" ("#") to the end of the line. It is ignored by the perl compiler.

5.0 What's Next?

For the purposes of the Mars Space Camp, we are going to teach you just enough "perl" to work with the Mars simulation. There is not really time to do more. If you get really excited about learning "perl," follow up on the references in Appendix A.

5.1 Examples

This document and the others will take you through what you should know--giving you plenty of examples. Some of it will be easy to guess. Here is your first example. What do you think it asks the computer to do?

```
#   Example p1s51e1.pl
#   give welcome
    print ("Welcome to Mars Space Camp!");
```

Notice that the first line contains a comment with the word "Example" and a name that comes after it. That name (p1s51e1.pl) is the name of the file on the floppy disk that you received that contains the example.

Decode the name as follows. It is from Part 1 (p1), which is this document. It is from Section 5.1 (s51). It is example 1 (e1). It is a perl script (.pl). With this information, you should be able to go back and forth between the examples in this document and the floppy disk.

Using MacPerl that you will find on the LRC computers and the information in Section 3, you should be able to try out the example. Try it as is to see it work. Then, make slight changes in it and run it again to explore your understanding of the example.

Appendix A

References

A.0 References

Here are various references to perl documents. **On one hand, don't buy any of these books because you are attending these sessions.** On the other hand, if you are really interested in perl, then these are some of the places to look for further information. Be sure you get your parents' and/or teachers' permission before you investigate any of the web sites.

A.1 Books, etc.

You can get books on "perl" at Anderson's Bookstore and at Books and Bytes. You can also get books on perl at Borders and Barnes and Nobel.

1. *Learning Perl*, Randal L. Schwartz & Tom Christiansen, O'Reilly & Associates, Inc. 1997.
2. *Programming Perl*, Larry Wall, Tom Christiansen & Randal L. Schwartz, O'Reilly & Associates, Inc. 1996.

A.2 Web Pages, etc.

1. <http://www.perl.com/perl/>.