Exploring Martian Planetary Images
C++ Exercises for CS1

Harriet J. Fell and Viera K. Proulx
Northeastern University
College of Computer Science
Boston, MA 02115, USA

fell@ccs.neu.edu, vkp@ccs.neu.edu

Abstract

We present a series of programming exercises based on photographic images of Mars collected by the NASA Viking Orbiter. Even without the news that there may once have been life on Mars [1], we feel that these exercises provide an exciting platform for exploring machine representation of data, presentation of data, and methods of storing and extracting data from files. All exercises are on the level easily mastered in the first programming course.

Introduction

The files used in these exercises have been downloaded via the Internet from the NASA depository. They contain original unenhanced grayscale image data preceded by a label identifying the image and a histogram of the grayscale values of this image. By working with real data files students appreciate why it is important to understand data formats, file organization, file manipulation, and different conversion methods. At the same time they learn about simple image enhancement, encoding, and image exploration via sound. In addition, two of the exercises include require students to create a written document relevant to the topic.

From a single file that appears to be just a long list of bytes, students can:

- Convert the header bytes into a text file that includes information such as:

```
MAP_PROJECTION_TYPE  = SINUSOIDAL
MAP_RESOLUTION       = 256<PIXEL/DEG>
MAP_SCALE            = 0.231352<KM/PIXEL>
MAXIMUM_LATITUDE     = 67.50000
MINIMUM_LATITUDE     = 62.50000
MAXIMUM_LONGITUDE    = 10.00000
MINIMUM_LONGITUDE    = -0.01627.
```

- Convert 4-byte records representing VAX-style integers into a histogram of grayscale values.
- Convert records of bytes into lines of a grayscale image of a region on Mars!
- Use the histogram information to perform two kinds of image enhancement.
- Write a routine that converts the image data to sound - music from Mars?
- Learn about covert channels and discover a hidden message in the graphic image. (Yes we tamper with the files to make this possible.)

Students can also go straight to the NASA Web site [3] to explore additional images from the Viking Orbiter.

Description of Exercises

The exercises in this suite can be assigned as independent programming projects (open labs) or used in a closed lab setting. They are presented in the order in which they may appear in a first year programming course. For each exercise we identify its prerequisites, the new concepts practiced in this exercise, and computer science ideas and applications that add breadth to the assignment.

Partial support for this work has been provided by the National Science Foundation Leadership in Laboratory Development , award #DUE- 9650552

Our version of these labs uses the Metrowerks CodeWarrior C++ environment for the Macintosh. It is a relatively easy task to adapt them to other platforms. Students need to be provided with software or instructions that allows them to

- set one pixel of a drawing to a specified grayscale shade
- read a file (one byte at a time)
- produce a sound, given the amplitude, frequency, and duration
- manipulate bits

Simple Image Enhancement.

Students display an image of Mars by reading and interpreting an original NASA data file. They then use scaling (a persistent theme in our course) to make the planetary features more distinguishable[2, Tech Note 2].

Prerequisite concepts:

- assignment statements and simple arithmetic
- basic loops and decision statements

Goal:

- practice loops, decision statements, assignment statements
- design and implement simple algorithm (minimum and maximum)
- learn to read data from a file
- learn about images composed of grayscale pixels
- implement and observe the results of a simple image enhancement algorithm

Activities:

Students are asked to write a program that will read the image data and display the image one pixel at a time.

• The first task is to read and discard the label and histogram portion of the file using a simple loop.

• Next task is to read the rest of the file one byte at a time, converting each unsigned byte into an integer that can be used to define the grayscale shade. Alternatively, if students are familiar with arrays or strings, they may process one line at a time. No additional internal storage is needed. Each pixel (or each line) is displayed before the next file read.

• Students then add code to find the maximum and minimum values of the grayscale shade during the first pass through the file (whether or not the data is displayed).

• Finally, students read the file again, but now the value of each grayscale shade is modified using a simple linear transformation. The image features become more visible (Figure 1, 2).

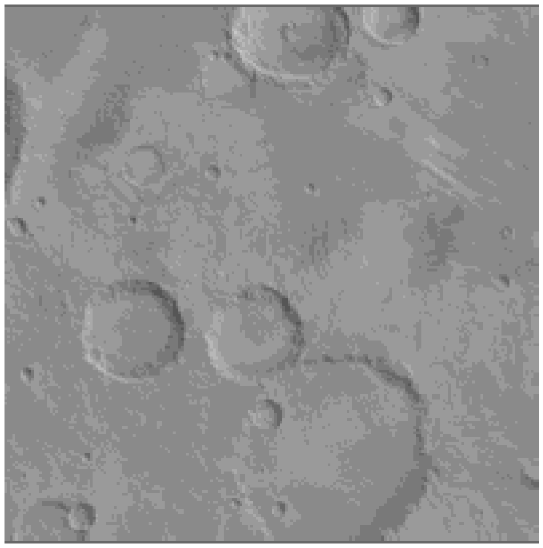

Figure 1. Original Image
showing 200 x 200 pixels

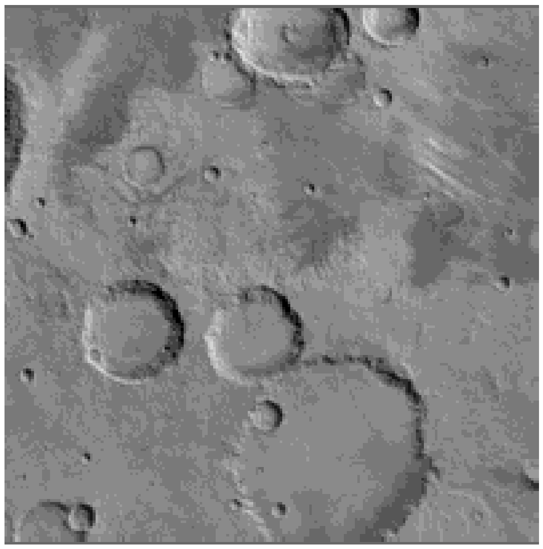

Figure 2. Enhanced Using Transformation
shade = (shade - min) *(256/ (max-min))

Notes:

• Students are given the length of the label field and the width and height of the image in pixels.

• Depending on student's abilities and time constraints, students may be given shell program that contains functions that perform the conversion from an unsigned byte to an integer, or functions that simplify opening and reading the file.

Data Conversion Advanced Image Processing

Students implement a histogram equalization algorithm for image enhancement. They see that by using simple array manipulation, they can produce amazing detailed images (Figure 3) from the original data[2, Tech Note 2].

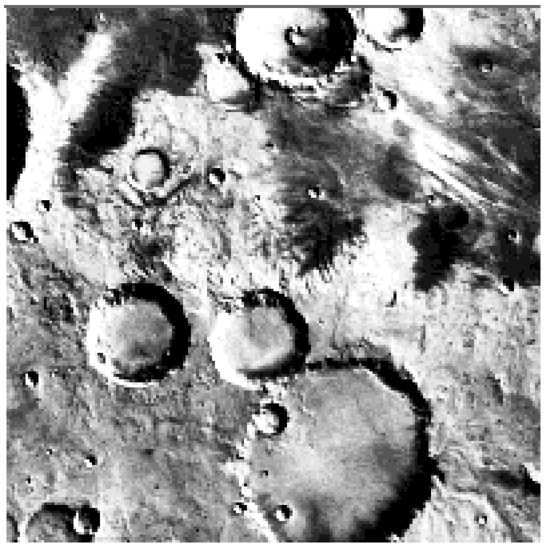

Figure 3. Histogam Equalization Enhancement

Prerequisite concepts:

- arithmetic, loops, decisions
- file reading
- one dimensional arrays

Goals:

- practice array traversals and simple array manipulation
- learn to display array values as a bar graph
- learn about representations of integers in binary and conversions between them
- implement and observe the results of advanced image enhancement algorithm
- practice written presentation techniques

Activities:

• Students are asked to write a program that will first read the histogram data from the file into an array of size 256. For each grayscale shade the histogram records the number of pixels of this shade. This data is stored as a series of VAX longword integers. Depending on the platform used, students may need to write a function that converts these integers into the format used by their computer.

• Next, students display the histogram data as a bar chart.

• Students then implement the algorithm for histogram equalization method of image enhancement. Histogram equalization is simply a transformation of the original distribution of pixels such that the resulting histogram is more evenly distributed from black to white (Figure 3). Students need to compute the scale factor $\mathbf{s_i}$ that is used to scale the shade of each pixel. The shade of each pixel initially at gray level **i** is multiplied by a scale factor $\mathbf{s_i}$ computed as:

$$\mathbf{s_i = (1/n) * sum(n_0, n_1, n_2, ..., n_i)}$$

where **n** is the total number of pixels and $\mathbf{n_i}$ is the number of pixels at gray level **i**.

• To include more array manipulation in this exercise, students may compute their own histogram rather than reading the data from the file. Or they may be asked to compute the histogram of the enhanced image and display it at the end. This helps students to understand the effects of histogram equalization.

• Finally, students are asked to create a document (using word processor or HTML) that explains an illustrates how the different images were created and how the two image enhancement algorithms work.

Text and String Processing

Students read and process information in the label portions of the original data files. They see the need for understanding the details of string manipulation, data representation, and data conversion.

Prerequisite concepts:

- loops
- file reading

Goals:

- learn about ASCII representation of characters
- create and write to a text file
- search a byte stream for a given character string
- convert numbers represented in ASCII characters to integers or floating point numbers
- implement and observe the results of advanced image enhancement algorithm
- practice written presentation techniques

Activities:

• Students are asked to write a program that will read the ASCII label of the data file and write the results to a text file. The bytes of the label may be read, one at a time, into an unsigned char variable and written directly to the output text file*. Students may start with a sample data file with known label size and simply process the necessary number of bytes. They should then read (and possibly print) the file they create with a text editor. They could be asked to report on specific information found in the file. They can use the latitude and longitude information to attach their images to those of other students.

• Once students have succeeded in creating and writing to a text file, they should modify their read loop to terminate when the characters "END_" are encountered. The label part of the files all terminate with this string though the string "END" appears at the end of each block of information. Now they can read arbitrary data files.

• The labels of the files we work with all start like this:

```
CCSD3ZF0000100000001NJPL3IF0PDS200000001
      = SFDU_LABEL
/*         FILE FORMAT AND LENGTH */
RECORD_TYPE          = FIXED_LENGTH
RECORD_BYTES          = 1184
FILE_RECORDS         = 1283
LABEL_RECORDS        = 2
/*            POINTERS TO START RECORDS OF
      OBJECTS IN FILE */
^IMAGE_HISTOGRAM     = 3
^IMAGE               = 4
```

The RECORD_TYPE is always FIXED_LENGTH. To process the images in arbitrary files automatically, the students must retrieve the:

RECORD_BYTES, up to four digits telling the record length in bytes.
FILE_RECORDS, up to four digits telling the total number of records contained in the file.
^IMAGE_HISTOGRAM, up to two digits telling the starting record of the Image Histogram Object.
^IMAGE, up to two digits telling the starting record of the Image Object (data).

Students may be asked to search for keyword strings like "RECORD_BYTES" and then to convert the numeric information following the "=" into an integer. Alternatively, they may use the fact that the label has a standard format and search for the second, third, fifth, and sixth "=" as these are followed by the numbers they need.

• Students should now improve their programs so that they can open arbitrary image data files, automatically find the dimensions and starting place of the image data, and display the graphic image.

Note:

• The header files are all in a standard format that is described in the NASA document [4].

Sonification, or Sound Analysis of Data

Sound, as well as graphics, can help in visualizing data [2, Tech Note 10]. Students write a program to produce sound output related to the grayscale of the pixel the mouse is passing over. Though this exercise is unlikely to divulge any new features in the Martian landscape, it does show students how to produce sound and how to work with mouse input.

Prerequisite concepts:
- arithmetic, loops, decisions
- a working program that produces Mars images

Goals:
- learn to read the mouse position
- use the mouse button to control a loop
- learn to retrieve the grayscale (or RGB) value at a pixel
- open one or more sound channels
- produce controlled sound output

Activities:

• Students are asked to write a program that plays "music" created from the image grayscale values. The music is generated as long as the mouse button is down using the grayscale of the pixel under the cursor.

Note::

• Students are supplied with a `SoundChannel` class that includes a constructor, a destructor, and the member function:

`void Play (short amplitude, short duration, long midiNote)` - where
$0 \le$ `amplitude` ≤ 255 controls the volume,
`duration` is in half milliseconds, duration = 2000 lasts one second.
$0 \le$ midiNote ≤ 127 is the MIDI note number, 60 = middle C.

They get to use and see the implementation of a simple class in preparation for writing their own classes.

• They are also given a function to read the mouse position:
void GetMouse(short&x, short& y).

Bit Manipulation, Covert Channel Encryption

Students extract a personal message (from Mars ?) from their own (somewhat corrupted) image data file.

Prerequisite concepts:
- arithmetic, loops, decisions
- file reading and writing
- bit manipulation

Goals:
- practice bit manipulation
- practice file reading and writing
- implement and use covert channel encryption algorithm
- explore social issues in computing related to encryption

Activities:

• Students read an image file that contains a hidden message encoded in the last bits of each image byte data. They write a program that extracts these bits, composing them into eight bit ASCII bytes and stores the results in a text file. Students can now print the encoded message or write a program that reads and displays the file as text.

• Students are asked to do a little research or reading on covert channel encryption and write a short essay about its dangers and advantages.

Note:

• The instructor has to encrypt the message, preferably creating several files so not all students receive the same message. The messages should contain a substantial amount of text so that students need to save results to a file.

• Students may be asked to display the image, to see that the image quality has not been compromised by the encryption and the fact that the image contains covert message is very difficult to detect (Figure 4).

Where to Find the Files

The Viking Orbiter and other planetary data files can be found at
ftp://pdsimage.wr.usgs.gov/cdroms/.
We suggest using the files in vo_2002 that start with "mg". For example:
ftp://pdsimage.wr.usgs.gov/cdroms/vo_2002/mg25sxxx/mg25s022.img.
These files are relatively small, about 100K and contain images that are approximately 300 by 300 pixels. See
ftp://pdsimage.wr.usgs.gov/cdroms/vo_2002/volinfo.txt
for a description of the file format.

Acknowledgments

The inspiration for the first assignment came from reading the book “Visualization of Natural Phenomena” and was motivated by the fact that quite simple algorithms for image enhancement generated images of strikingly better quality. The image data files have been downloaded from the NASA depository (CREDITS). The fact that all data files start with plain text labels describing the file contents motivated the next three exercises. Erich Neuwirth suggested the fourth exercise that illustrates the use of a covert channel for encryption.

References

1. Chandler, David L., "Mars meteorite may hold evidence of microscopic life", *The Boston Globe*, August 7, 1996.
2. Wolff, Robert S. and Yaeger, Larry, *Visualization of Natural Phenomena*, Springer-Verlag, New York, 1993.
3. NASA Image web/ftp Site
ftp://pdsimage.wr.usgs.gov/cdroms/
4. NASA Volume Information:
tp://pdsimage.wr.usgs.gov/cdroms/vo_2002/volinfo.txt