Pgraf: A Portable Graphics System

David J. Raymond

Physics Department and Geophysical Research Center, RDD
New Mexico Institute of Mining and Technology
Socorro, NM 87801

ABSTRACT

This paper provides a brief description of a portable graphics package for the presentation of scientific data. The package is designed to operate on the simplest of graphics devices, with implementation on a new device requiring only the creation of a small set of low level subroutines. The package is written in the C language, with function calls restricted to the standard input-output and math libraries. However, the graphics is accessible from both C and Fortran 77. The functions provided include simple graphing, contouring of two dimensional fields, and perspective viewing of three dimensional wire frame structures.

1. Introduction

Pgraf was created as a result of dual frustrations. First, available public graphics systems have been both primitive and unstable, with "improvements" generated by over-eager systems programmers raising havoc with existing applications. Second, initial attempts to write a private graphics package foundered when too much dependence was placed on the peculiar features of a particular graphics device, resulting in lack of portability. With these experiences in mind, I set out to write a graphics package that would be transportable to any sensible computer and operating system, and would work on the widest possible variety of graphics devices.

The package was initially developed on a PDP-11 running the VENIX operating system. The limited address space of the PDP-11 dictated a modular approach, with application programs calling a set of small subroutines that created plotting instructions to be interpreted by a separate program. This set of plotting instructions, or *metacode*, was written into a file rather than being piped UNIX style to the graphics interpreter so that the package could be transported to non-UNIX systems if necessary.

Pgraf was originally written in Fortran, but was later converted to C in the face of an environment hostile to Fortran. Nevertheless, both Fortran and C versions of the subroutines called by applications programs were retained.

The package demands very little of the graphics device being used. Graphics commands are restricted to drawing vectors and printing text. It is nice if the device supports the drawing of different line types, but this isn't strictly necessary, as a dashed line is just a sequence of solid line segments. Only horizontally oriented text of a single size is used. No interactive capability is provided. Implementation on new hardware is thus quite simple.

The functions supplied by pgraf break naturally into three sub-areas; simple graphing, contouring of two dimensional data and related functions, and three dimensional perspective graphics. Different plots may be made in separate screen areas or *windows* defined by the user. Axes may be plotted in a simple way for each window. Graphs may be made using a variety of line types and symbols.

Contour plots can be highlighted by filling regions that exceed a specified value with hatching. Labelling contours is beyond this programmer, but a grid of numbers representing field values can be superimposed on a contour plot. The contouring algorithm is very simple-minded.

Both simple graphing and the contouring calls assume that data exceeding 999.0 in absolute value represent bad or missing data. These data are not used in generating plots. This might be viewed as a problem for users with valid data that exceeds these bounds. However, limitations in the format of axis labels already precludes the direct plotting of such data, thus forcing the user to scale it down.

The basic function of the three dimensional graphics is to view sets of line segments in three dimensions (*wire frames*) in perspective. The user defines a *focal point*, a *perspective viewing point*, and a magnification (*zoom*). The view point is thus under user control. No attempt is made to implement hidden line removal. However, contours of a function defined on a two dimensional plane of arbitrary orientation may be drawn.

Functions supported by pgraf necessarily arise from the peculiar needs of the author. Other users may desire to add different functions as their personal needs dictate. This is in accord with the idea of having *user based* rather than *system based* graphics. However, a number of users besides myself have found pgraf useful in its present form. (Requests for help in implementing new functions will probably be met with the reply that the author has just left for Acapulco, and won't be back for 6 months!)

The University Corporation for Atmospheric Research (UCAR) Unidata project has specified the GKS graphics standard as a basis for Unidata-sanctioned software. GKS is a low level standard that is intended to provide a stable base for the development of higher level applications. When a GKS package becomes available to the author, a GKS-based version of pgraf will be developed.

The rest of this document describes pgraf in more detail. Pgraf(1) describes the metacode interpreters that function on a Sun Microsystems monochrome workstation, a Hewlett Packard graphics terminal, and an Imagen laser printer. Instructions for implementing pgraf on new devices are also included. Pgraf(3) describes the C and Fortran subroutine calls used to generate pgraf metacode. Pgraf(5) describes the metacode format. Finally, some examples of pgraf use are presented.

In C programs the file pgraf.h should be included. An equivalent include statement is not required for Fortran. To compile and link a C program to generate metacode, include -lpgc -lm on the cc command line. (This assumes that the pgc library has been put in the right place. Procedures will, of course, be different on non-UNIX systems.) To compile and link an equivalent Fortran program, include -lpg77 on the f77 command line.

2. Acknowledgments

Several students played important roles in the development of pgraf. Dale Harris in particular deserves thanks. This work was supported by the National Science Foundation Meteorology and Unidata programs under grants ATM-8311017 and ATM-8605136.