Gri 2.8 Reference Card

1 What Gri Is

Gri is a language for drawing scientific diagrams such as x-y graphs, contours, vector fields, and images. The Gri language is extensible, well-tested and fully documented. The output is in the PostScript page description language.

2 How to Run Gri

Normally Gri is run non-interactively. At the system prompt, type `gri foo.gri` to run Gri on the file `foo.gri`, creating a PostScript file called `foo.ps`. (If the script name ends in `.gri`, then there is no need to type the suffix.) Several command-line options exist; type `gri -help` to see them, or consult the manual.

Occasionally you might want to run Gri interactively. To do this, type `gri` at the system prompt, and then type Gri commands at the Gri prompt, using `quit` to get out of Gri.

3 Overview of Gri Language

3.1 Syntax

Commands normally appear one per line, although ending a line with back-slash causes Gri to scan the next line also. Comments may be inserted at the end of non-continued lines by preceeding the comment by a hash-code (`#`).

Gri allows the usual suite of programming structures, such as loops and if-statements; additionally, new commands may be added to Gri easily (see section 3.7).

3.2 Built-in Commands

Here are the first words of the built-in Gri commands:

`cd`	`close`	`convert`	`create`	`debug`
`delete`	`differentiate`	`draw`	`expecting`	`filter`
`flip`	`get`	`help`	`if`	`ignore`
`input`	`insert`	`interpolate`	`list`	`ls`
`mask`	`move`	`new`	`open`	`pwd`
`query`	`quit`	`read`	`regress`	`reorder`
`rescale`	`resize`	`return`	`rewind`	`set`
`show`	`skip`	`smooth`	`sprintf`	`superuser`
`system`	`write`			

To get more information on a given command, e.g. the `open` command, type `help open` in an interactive Gri session, or `C-H i gri commands open` in an `emacs` editing session, or `info commands open` at the system level.

3.3 Mathematics

Wherever Gri expects to see a number in a command, one may substitute a mathematical expression written in reverse polish notation (RPN) notation. RPN expressions are enclosed in braces and preceeded by the word `rpn`, e.g.

```
set x size { rpn 5 2.54 * } # Make width be 5 inches
x += { rpn 1 2 /}           # Add 0.5 to x values
y -= { rpn y mean }         # De-mean y column
x -= { rpn x 0 @ }          # Subtract first value
```

3.4 Variables (for Storing Numbers)

User-defined variables have names that begin and end with periods (like `.offset.`); variables defined by gri (which you may alter if you wish) have names that begin and end with two periods (like `..xsize..`). To list the variables use `show variables`. Each of the following commands accomplishes the same thing, making the plot 2 cm wider. (Gri uses `..xsize..` to store the width of the plot.)

```
..xsize.. = { rpn ..xsize.. 2.0 + }
..xsize.. += 2
set x size { rpn ..xsize.. 2.0 + }
set x size bigger 2
```

3.5 Synonyms (for Storing Strings)

Synonyms have names which begin with backslash (like `\name`). To list the synonyms use `show synonyms`. Synonyms can be embedded within strings or used raw, e.g.

```
\dir = "mydir"
query \filename "What's the data file?" ("file.dat")
open \mydir/\filename
read columns x y
draw curve
draw title "Data in \filename in dir \mydir"
```

3.6 Strings and Math Symbols

Strings are enclosed in double quotes. As in TEX, superscripts and subscripts are enclosed in dollar signs. Subscripts are preceeded by underscore, superscripts by carat. Superscripts or subscripts consisting of more than one character are enclosed in braces.

Gri handles Greek letters and mathematical symbols as TEX does: they are enclosed in dollar signs and have backslash as the first character. Most Greek letters are available, along with several mathematical symbols, but complicated LaTEX macros (like `\frac{}{}`) are not available. Examples:

```
set x name "x/x$_0$"
draw title "y$_{dim}$ as fcn of $\alpha$"
```

3.7 Exte

You can creat
your ~/.grir
where – they
mands define
The example
by Landscape
with upper ca
commands in
help lines foll
with an open
brace.

```
'Landscape B
Plot in land
{
    set page
    set x ma
    set x si
    set y ma
    set y si
}
open file.da
read columns
close
Landscape Bi
draw curve
quit
```

4 Editi

A `gri-mode` is
It provides co
the gri manua
down menus,
ment placeme
scription. To
~/.emacs file

```
;;; Gri mode
(autoload 'g
(setq auto-m
    (cons '(
```

5 Docu

An `info` man
Emacs. A Pos
book of Gri e
store PostScri
ager to find th
at *http://gri.*
discussion gro

6. Example – Linegraph

Suppose the file `example1.dat` contains data in two columns separated by white space. The following shows how to plot data with lines connecting the points. To get symbols without lines, substitute `draw symbols` for `draw curve`; to get both symbols and lines, use both `draw` commands. If you have several curves which cross, use `draw curve overlying`, which whites out a border below each curve, yielding a visual cue that lets the eye trace the individual curves easily.

```
# Example1.gri -- linegraph using data in a file
open example1.dat      # Open the data file
read columns x y       # Read (x,y)
draw curve             # Draw curve stored in (x,y)
```

You'll notice that there was no need to ask that axes be drawn. They will be automatically determined (based on the data that were read in) and drawn just after the `draw curve` command. (Gri likes to draw axes, and you've got to ask it not to do so, if you don't want them.) Also, note that Gri was not instructed to close the datafile. This is done automatically at termination. One could insert a `close` command after the `read` command, if desired; this is helpful when you wish to work with many data files sequentially.

The axes are labelled `x` and `y`. To change that, and to add a title, do as follows:

```
open example1.dat
read columns x y
set x name "Time, s"
set y name "Distance, m"
draw curve
draw title "Trajectory of fluid motion"
```

To get a thicker curve, say 2 points wide, you could do

```
open example1.dat
read columns x y
set line width 2
draw curve
```

To get a dashed line,

```
open example1.dat
read columns x y
set dash
draw curve
```

To get a red line,

```
open example1.dat
read columns x y
draw axes
set color red
draw curve
```

Note in the above that the axes were drawn before the color was set to red, so they will come out black. Otherwise both the curve and the axes would be red.

7. Example – Contour Graph

Gri can plot contour graphs of either gridded or ungridded data. Several methods are provided for gridding data. The following example shows how to grid randomly distributed (x,y,z) data and plot contours.

```
# Example 5 - Contouring ungridded data, from figure
# 5 of Koch et al., 1983, J. Climate Appl. Met.,
# volume 22, pages 1487-1503.
open example5.dat
read columns x y z
close
set x size 12
set x axis 0 12 2
set y size 10
set y axis 0 10 2
draw axes
set line width symbol 0.2
set symbol size 0.2
draw symbol bullet
set font size 8
draw values
set x grid 0 12 0.25
set y grid 0 10 0.25
convert columns to grid
# Uncomment next line to smooth the grid:
#smooth grid data
set font size 10
draw contour 0 40 2
set font size 12
draw title "Data from Fig 5 Koch et al., 1983"
quit
```

Note that a `quit` command has been included, although it is not required, since Gri quits when it reaches the end of the commandfile anyway.

8. Exa

Gri can draw
shows how to

```
# Example 6
# define cha
\0val
\255val
.r.
.c.
.pixel_width
.km.
# get filena
query \filen
query \maskn
# get data
open \filena
set image ra
read image .
close
open \maskna
read image m
close
# find out w
query \histo
query \Tw
query \Tb
# set up sca
set x size 1
set y size 1
set x name "
set y name "
set x axis 0
set y axis 0
# plot image
if {rpn \his
    set imag
        blac
else
    set imag
end if
draw image
draw image p
draw image h
if {rpn \his
    draw tit
else
    draw tit
end if
```

See also *cmdre*