GRASS 4.1 User's Reference Manual

Spring, 1993

United States Army Corps of Engineers
Construction Engineering Research Laboratories
Champaign, Illinois

GRASS is a product of the US Army Corps of Engineers Construction Engineering Research Laboratories (USACERL), in Champaign, IL. It is an integrated set of programs designed to provide digitizing, image processing, map production, and geographic information system capabilities to its users.

Other agencies have also written programs for distribution with GRASS. System development continues to be conducted by the Environmental Laboratory of USACERL.

Individual program authors are cited within.

CONTENTS

1 INTRODUCTION Page 3

2 MAIN PROGRAMS and PROGRAMS IN ALPHA TESTING Page 5

3 SHELL SCRIPT PROGRAMS Page 505

4 INDEX of CONTRIBUTED PROGRAMS Page 525

5 DESCRIPTIONS of PROGRAM FORMATS Page 533

6 INDEX Page 551

1 INTRODUCTION

Background

This reference manual details the use of programs distributed with version 4.1 of the Geographical Resources Analysis Support System (GRASS), a public domain, image processing and geographic information system (GIS) originally developed by researchers in the Environmental Division of the US Army Construction Engineering Research Laboratories (USACERL) in Champaign, Illinois. The GRASS system is used to input, manipulate, analyze, and output geographic data by users in both military and nonmilitary, public and private agencies based in North America, Europe, and other parts of the world.

Although most GRASS development has been conducted at USACERL, system integration, development, testing, distribution, training, and support are performed by numerous publicly and privately operated sites throughout the world.

Reference Manual

The manual explains the capabilities, syntax, and use of current GRASS programs. Chapters contain program entries which are ordered alphabetically by program name. Each entry contains:

* the program name
* a brief description
* a synopsis of files associated with the program
* a full description of the program's capabilities and uses
* a summary of the command-line operations associated with the program
* a list of associated or related programs
* a list of program authors

Each entry may optionally include a description of the program's operation in interactive mode, and a discussion of "bugs," program limitations, or special notes of interest.

Throughout the manual, program input to be explicitly entered by the user is printed in bold text. User-specified input (i.e., variables set by the user) is set in italic type. Boxed text shows information output to the user's terminal screen.

GRASS Categories

The GRASS code has been divided into the following categories according to reliability, testing, and application:

* Main and Alpha
* UNIX Shell Scripts
* Contributed
* Garden
* Related

Descriptions of Main and Alpha, Shell Script, and Contributed categories are presented at the introduction

of each appropriate chapter in this manual.

Other software distributed with GRASS but not documented in this manual includes:

(1) Hardware- and software-specific code: SG3d (specific to Silicon Graphics computers) and XGRASS (specific to ***Motif***).

(2) Garden and Related programs contained in the ***src.garden*** and ***src.related*** directories, which the user must compile separately (see the GRASS ***Installation Guide*** for compilation instructions): Any original documentation for these programs is compiled on-line with the directories. Garden includes program interfaces that link GRASS 4.1 with other related software. Related includes programs to which GRASS has been interfaced and that program developers have allowed to be distributed with GRASS.

2 MAIN PROGRAMS and PROGRAMS in ALPHA-TESTING

This chapter describes the syntax and use of GRASS 4.1 programs that have undergone (and survived) alpha- and beta-testing (i.e., initial testing by program developers, and subsequent testing by out-of-house users). These programs constitute the bulk of GRASS and are respectively referred to in the documentation as "Main Programs" and "Alpha Programs." Main programs, which receive both alpha- and beta-testing prior to their release and are located in the ***src*** directory. Alpha programs receive only alpha-testing prior to release and are located in the ***src.alpha*** directory. For the GRASS 4.1 release, both types of programs compile automatically during general GRASS compilation. The GRASS ***Installation Guide*** distributed with GRASS 4.1 software explains the compilation procedure.

NAME

d.3d − Displays three-dimensional images based on raster map layers.
(GRASS Display Program)

SYNOPSIS

d.3d
d.3d help
d.3d [**−l0a**] **map=***name* **elevation=***name* **[from_coordinate=***x,y,z]*
[to_coordinate=*x,y,z]* **[exaggeration=***value]* **[lines=***value]* **[field=***value]*
[color=*name]* **[box=***name]*

DESCRIPTION

d.3d displays three-dimensional graphic images based on GRASS raster map layers. The user identifies the viewing point, the line of sight, a vertical exaggeration factor, the viewing angle (field of view), the frequency and color of vector grid lines to appear in the display, the map to be displayed in 3-d, and the map whose category values are to be used as elevation values in the 3-dimensional image.

The program will be run non-interactively if the user specifies all needed parameter values and flag settings on the command line. Alternately, the user can simply type **d.3d** on the command line; in this case, the program will prompt the user for parameter values and flag settings using the standard interface described in the manual entry for *parser*.

Three-dimensional images can also be generated through the GRASS *d.display* program. enter short description of d.3d here

COMMAND LINE OPTIONS

-l Display *lines* only, without displaying the raster base *map*.

-0 Show zero elevations.

-a Each grid-cell is rendered using four elevation coordinates: one for each corner. By default, each corner takes the value of the grid-cell to the lower right. With the -a option each corner is calculated to be an average of the four bordering cell elevation values.

Parameters:

map=*name* The raster map layer used to generate the color shown in the 3-d output (i.e., the map whose x and y values will be displayed, using the z values present in the *elevation* map layer.)

elevation=*name* The raster map used to generate the texture in the 3-d image (i.e., the map whose category values will become the elevation values in the 3-d map displayed). This need not be an elevation map layer, although elevation is commonly used.
Default: elevation

from_coordinate=*x,y,z*
Coordinates of the viewing point, given as: *northing,easting,elevation*. The default *from_coordinate* value is appropriate for the spearfish sample data set.
Default: Calculated at run-time just south-west of the south-west corner.

to_coordinate=*x,y,z* Coordinates of center of view, given as: *northing,easting,elevation*. The default *to_coordinate* value is appropriate for the spearfish sample data set.
Default: Calculated at run-time as the center of the image

exaggeration=*value* Vertical exaggeration factor. This value is multiplied by the "elevation" values (category values) in the *elevation* map layer. As the vertical coordinates are exaggerated, the elevation (z) value of the center of view (the *to_coordinate*) should be increased.

Default: 2.0

lines=*value* North-south and east-west trending lines can be drawn on the 3-d image output, to enhance the 3-d effect. The *lines* value specifies the number of rows and columns to skip between lines. 0 means "display no lines". Lines will be drawn in the *color* stated. The user can elect to only display these lines, without displaying the *map*, by setting the -l flag.
Default: 1

field=*value* The field of view (viewing angle) in the image, stated in degrees.
Default: 30

color=*name* Color of vector *lines* drawn on the output.
Options: color, white, red, orange, yellow, green, blue, indigo, violet, gray, black
Default: gray

The resulting three-dimensional image is drawn in the active frame on the graphics monitor. The user should select and erase the full graphics monitor frame before running *d.3d* to prepare the screen for graphics. Refer to the GRASS program *3d.view.sh* for a demonstration of the *d.3d* program.

INTERACTIVE MODE

The interactive mode prompts for the raster map to be rendered, the raster map to be queried for elevation information, and an optionally saved set of saved viewing parameters. The user is then presented with a form in which the options can be modified.

It is suggested that the following procedure be used to efficiently find the correct viewing parameters. Start with a gross grid (by default, a resolution is chosen which limits the total number of rows and columns), display lines only, set line colors to "color" (lines take their color from the map), set the box color to something other than "none", and use the default viewing coordinates. Experiment with the viewing coordinates until the map is positioned in a desireable configuration. When the map is lined up correctly, set the display resolution as low as you like for a final image.

BUGS

Several additions to this program have been suggested. These include the following:
1) Border the image with a visually appealing curtain.
2) Add x,y,z coordinates and scale information to the displayed 3-d image.
3) Provide a graphic-oriented user interface for identifying viewing parameters.
4) Display output more quickly by doing mathematics in integers.
5) Add the option of plotting vector map layers.

SEE ALSO

3d.view.sh, d.display, d.erase, d.frame, and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.ask − Prompts the user to select a GRASS data base file from among files displayed in a menu on the graphics monitor.
(GRASS File Management Program)

SYNOPSIS

d.ask help
d.ask element=*name,description* [**prompt="***message*"]

DESCRIPTION

d.ask is designed for shell scripts that need to prompt the user for the name of a data base file in the user's current GRASS mapset search path. After *d.ask* is invoked with needed parameters, the mouse becomes active and a menu containing files of the specified *element* type is displayed on the user's graphics monitor. The user is prompted by the prompt "*message*" to select one of the listed file names with the mouse. Specifically, the query that appears to the user takes the form:

Double click on the *prompt "message"*
Double click here to cancel

(A list of files of the specified *element* type from each of the mapsets
listed in the user's mapset search path is also displayed.)

After the user responds, the mouse is deactivated and the displayed menu is erased from the screen (leaving any underlying materials on display intact). Three lines are written to standard output (the user's terminal screen):

name=*file_name*
mapset=*mapset_name*
fullname=*file_name@mapset_name*

Parameters:

element=*name,description*
Name of a GRASS data base element, followed by a one word description of the element. GRASS data base elements are mapset subdirectories; these include: bdlg, cats, cell, cell_misc, cellhd, colr, colr2, dig, dig_ascii, dig_att, dig_cats, dig_plus, dlg, group, hist, icons, noise, paint, site_lists, and windows. The *description* will be used to display an error message to the screen if no files of the named element type exist in the user's mapset search path. The prompt "*message*" will appear in the pop-up menu displayed on the user's graphics monitor. The files listed in the menu will be of the specified *element* type, and exist in mapsets listed in the user's current mapset search path.

prompt="*message*" A brief message with which the user will be prompted. If this message contains more than one word, it should be enclosed within double quotes (""). It is a non-selectable message displayed in the menu bar of a pop-up menu. Selectable items in the displayed menu include a cancel option and any of the files in the mapset subdirectory *name* in the user's mapset search path.

EXAMPLE

Given the following input,

d.ask element=cell,raster prompt="raster map layer to be used"

d.ask will prompt the user to select a raster (cell) file from among those listed in the "cell" directories of the mapsets listed in his current mapset search path. A pop-up menu will be displayed on the user's graphics monitor, containing the prompt:

Double click on the *raster map layer to be used*
Double click here to cancel

OUTPUT

If the user selects a file name from the displayed menu, ***d.ask*** writes three lines to standard output:

name=*file_name*
mapset=*mapset_name*
fullname=*file_name*@*mapset_name*

The output is in the form of /bin/sh commands to set the variable ***name*** to the file name specified by the user (of the ***element*** type requested by ***d.ask***), ***mapset*** to the GRASS mapset in which this file resides, and ***fullname*** to the name of the file and its mapset.

If the user elects not to select a file but instead chooses the "Double click here to cancel" option, the mouse will be deactivated and no variable assignments will be returned to standard output.

If no files of the specified ***element*** type are found in the user's current mapset search path, the mouse is activated and the following message is displayed on the user's graphics monitor:

No *element description* **files found**
Click here to continue

NOTES

Parameter values (the ***element*** name and description, and ***prompt*** message) cannot be supplied to ***d.ask*** interactively; they must be supplied on the command line or from a file.

SEE ALSO

g.ask, g.filename, g.findfile, g.gisenv, g.mapsets

BUGS

This program calls other GRASS programs which may generate confusing error messages. Example: an error message from the program d.menu will appear if a nonexistent map element is chosen.

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

d.colormode − Allows the user to establish whether a map will be displayed using its own color table or the fixed color table of the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.colormode
d.colormode help
d.colormode mode=*name*

DESCRIPTION

d.colormode establishes what color table will be used to display maps to the graphics monitor.

Parameters:

mode=*name* Options: *fixed* or *float*

Each time a new raster map layer is drawn on the screen, a color table associated with that map is loaded into the graphics display monitor. The command **d.colormode mode=fixed** requests that the monitor's color look-up table be *fixed* (i.e., static). The effect is that colors drawn on the screen by graphics calls will not change when subsequent maps are drawn to the screen. When the *fixed* option is used, the colors in the map's color table are mapped to the nearest colors available in the monitor's fixed color table. When the *float* option is used, the map's color table is loaded directly into the monitor's color look-up table.

There are advantages to using each color display mode. The *d.display* and *d.colors* programs allow the user to interactively modify the color tables of maps displayed to the graphics monitor. This is done by allowing the user to directly modify the monitor's color look-up table. Hence, to accommodate this option in *d.display* and *d.colors*, the *d.colormode mode=float* option is chosen. However, this option has the sometimes undesirable effect of changing the colors in which other maps are subsequently displayed to the graphics monitor (although it does not actually change the *color tables* of these latter maps). The *float* colormode is therefore best used when the user wishes to interactively change a map's color table, or when the user wishes to display one or more maps having the same color table. The *fixed* option allows any number of maps to be displayed to the screen, where each map uses different colors, but all use the same fixed color look-up table. You cannot toggle a map's color table when running in fixed mode.

This program will be run non-interactively if the user specifies the color *mode* on the command line (e.g., by typing **d.colormode mode=fixed** or **d.colormode mode=float**). Alternately, if the user simply types **d.colormode** on the command line, the program will prompt the user for the color mode using the standard GRASS parser interface described in the manual entry for *parser*.

NOTES

Some devices support only a fixed color look-up; e.g., ink-jet printers, plotters, and the AT&T 6300 (running a DEB board). The number of color categories capable of being displayed is also device-dependent; if your colors fall outside this range when in *float* mode, they may not be *displayed* in the colors indicated in the map's color table file. However, the color table files themselves will accurately represent the user's changes.
Color table files associated with raster map layers are stored in the user's current mapset under the *colr* and *colr2* directories.

BUGS

It is strongly recommended that the user erase the graphics monitor screen (e.g., by running *d.erase*) immediately after changing the mode between *fixed* and *float.*

SEE ALSO

d.colors, d.colortable, d.display, d.erase, d.rast and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.colors − Allows the user to interactively change the color table of a raster map layer displayed on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.colors
d.colors help
d.colors map=*name*

DESCRIPTION

A color table file associates specific colors with the categories of a raster map layer. The user can change these map category color assignments (i.e., change the map's color table) interactively, by first displaying the raster map to the graphics monitor and then running the program *d.colors*. If you have first set *d.colormode* to "float" before displaying the map to the graphics monitor, any color changes that you subsequently make while in *d.colors* will immediately (and interactively) appear on the graphics display.

While running *d.colors*, the cursor must be inside the window where the execution takes place. When in "float" colormode, the correct colors appear on the graphics monitor only if the cursor is inside the monitor. To see the correct colors, move the cursor from the execution window to inside the graphics monitor, then move back to the window to continue making changes.

If *d.colormode* is "fixed", any color changes made using *d.colors* will not immediately be shown on the graphics display; however, any color changes saved will still alter the map's color table and will appear next time the raster map layer is redisplayed (see *d.colormode* and *d.colortable*). Note that some graphics display monitors or drivers cannot support interactive color change.

The user must first display the relevant raster map layer to the active frame on the graphics monitor (e.g., using *d.rast* or *d.display*) before running *d.colors*. The user can then either enter the name of the raster map layer whose color table is to be changed on the command line (e.g., by typing: **d.colors map=soils**), or type **d.colors** without program arguments. If the user simply types **d.colors** without program arguments on the command line, *d.colors* will ask the user to enter the name of an existing raster map layer using the standard GRASS interface described in the manual entry for *parser*.

In either case, the user is then presented with the *d.colors* command menu, shown below. This menu is the same as the category and color changing portion of the *d.display* menu. The *d.colors* commands are listed beneath the Category Pointer Movement, Color Modification, Replotting Screen, and Quitting sections below. Commands are invoked by typing in the single-key response shown to the left below. (Longer descriptions of these commands appear to the right.) Results from invoking these commands will be reflected in the Category and Category Number sections of the *d.colors* screen. On the *d.colors* screen menu, commands appear in the right half of the screen, and the current status of categories appears in the left half of the screen.

```
CATEGORIES
  0   No Data
  1   (Category 1 description)
  2   (Category 2 description)
  .   ....
  .   ....
CATEGORY NUMBER:
  RED      0   0%
  GREEN    0   0%
  BLUE     0   0%
Shift Incr:  10   3%
CATEGORY POINTER MOVEMENT
  D/d  down (cats)                    Move pointer to next category
```

U/u up (cats)	Move pointer to previous category
COLOR MODIFICATIONS	
R/r RED	Increase/decrease RED intensity
G/g GREEN	Increase/decrease GREEN intensity
B/b BLUE	Increase/decrease BLUE intensity
I/i increment	Increase/decrease increment (of intensity shift)
h highlight	Highlight current color
+/- shift colors	Shift entire color table (up/down)
c save color	Save color table
t toggle table	Toggle to different color table
REPLOTTING SCREEN	
* Replot screen	Replots the screen
QUITTING	
Q quit	Quits program

Changing categories - The keys **d, D, u,** and **U** are used to move to a different category. The lower case letters move up, **u,** and down, **d,** the category list one category at a time. The upper case letters move 10 categories at a time for fast movement. The cursor does wrap between the first and last categories. The current category is noted on the text screen with an arrow, and is indicated on the graphics screen by a box around the current color.

Changing colors - The color associated with the current category can be changed with the **R, r, G, g, B,** and **b** keys. The upper case letters increase the intensities of red **R,** green **G,** and blue **B** for the current category; the lower case letters decrease the intensities of these same colors for the current category. Video devices make all the colors of the spectrum by mixing red, green, and blue. For those accustomed to red, yellow, and blue being the primary colors, this can be confusing. For starters, yellow is made by mixing red and green. The intensities are listed on the text screen in as percentages.

Keys **I** and **i** increase and decrease the percentage change that each keystroke of one of the color keys (*R, r, G, g, B, b*) causes in its respective color. The increase increment is initially set to 10%. Thus, pressing the *R* key would will increase the red component of the current category by 10%.

Highlight - The **h** key toggles between the current category color and the current highlight color. This color is initially black but can be modified as above while in highlight mode. Blinking can be accomplished by repeatedly striking the **h** key. When changing to different categories using the movement keys as described above, while in highlight mode the category colors will be always left showing their actual colors. Only one category is highlighted at any one time.

Saving the current color table - Pressing the **c** key will save the current color table as you have modified it. This table will then be used next time you display or paint this raster map layer.

Color table toggle - Different types of color tables are suitable for different raster map layers. The key **t** flips between the following color tables: red, green, blue color ramp; gray scale; smooth changing color wave; random colors; and the saved color table.

Color table shift - The entire table is shifted up and down using the **+** and **-** keys.

Quitting the *d.colors* program - Pressing the **Q** key will cause you to quit the *d.colors* program. If colors have been modified but not saved, *d.colors* will ask:
Colors changed
Save the changes? (y/n)
The user should type in **y** to save changes, or **n** to not save changes, before quitting the program. If the user types *n*, the program will ask:

Quit anyway? (y/n)

NOTES

To see map color changes reflected on the display monitor as you make them, you must put the monitor in *float* color mode before running *d.colors*. It is also wise to erase the display screen after changing the color mode.

The map whose color table is to be altered with *d.colors* must already be on display in the active display frame on the graphics monitor before *d.colors* is run. This can be done using the command **d.rast map=***name* (where *name* is a raster map layer whose color table the user wishes to alter).

The user might type the following sequence of commands to interactively change the colors of a raster *soils* map:

d.colormode mode=float
d.erase
d.rast map=soils
d.colors

After the user has saved any color changes made with *d.colors* and exited the program, the user could then redisplay the *soils* map with the new colors by typing:

d.colormode mode=fixed
d.erase
d.rast map=soils

Some color monitors may not support the full range of colors required to display all of the map's categories listed in the map's color table. However, regardless of whether the user can see the color changes he is effecting to a map's color table, any changes to a map's color table made with *d.colors* that are saved will appear in the map's color table.

Some monitors may not support an interactive color change capability.

SEE ALSO

d.colors.boxes, d.colormode, d.colortable, d.display, d.rast, p.colors, r.colors and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.colortable – To display the color table associated with a raster map layer.
(GRASS Display Program)

SYNOPSIS

d.colortable
d.colortable help
d.colortable map=*name* [**color**=*name*] [**lines**=*value*] [**cols**=*value*]

DESCRIPTION

The GRASS program *d.colortable* is used to display the color table associated with a raster map layer in the active frame on the graphics monitor. The map *name* should be an available raster map layer in the user's current mapset search path and location.

Parameters:

map=*name* Name of a raster map layer in the user's current mapset search path whose color table is to be displayed.

color=*name* Color of the lines separating the colors of the color table.
Options: red, orange, yellow, green, blue, indigo, violet, gray, white, and black.
Default: *white*

lines=*value* Number of lines to appear in the color table.
Options: 1 - 1000

cols=*value* Number of columns to appear in the color table.
Options: 1 - 1000

If the *value*s of both *lines* and *cols* are not specified by the user, *d.colortable* divides the active frame equally among the number of categories present in the named raster map layer. If one option is specified, the other is automatically set to accommodate all categories. If both are specified, as many categories as possible are displayed.

The user can specify all needed parameters on the command line using the form:

d.colortable map=*name* [**color**=*name*] [**lines**=*value*] [**cols**=*value*]

If the user specifies the name of a map on the command line but does not specify the values of other parameters, parameter default values will be used. Alternately, if the user types simply **d.colortable** on the command line without any program arguments, the program will prompt the user for parameter settings using the standard GRASS parser interface described in the manual entry for *parser*.

EXAMPLE

The user running the command:

d.colortable map=soils color=red lines=1 cols=3

would see the active graphics frame divided into three columns extending the full frame height. The lines dividing the color table associated with the *soils* map would be displayed in red. The user would see, at most, only three of the colors from the *soils* color table displayed in the active frame (because the user requested that this frame be divided into three sections).

NOTES

If the user wishes to display the entire color table associated with a map, the user should either stipulate a number of lines (rows) and columns (cols) sufficient to accommodate the number of categories in the map's color table, or fail to assign values to one or both of *lines* and/or *cols*. If the user runs *d.colortable* using the default number of lines and columns (the full graphics frame), all categories from the map's color table will be displayed. However, if the user requests that the color table associated with a map which has 10 data categories be displayed in a graphics frame with only 3 lines (rows) and 2 columns (a total of six cells), only six of the ten map categories will be displayed.

The user should run the command **d.colormode mode=float** before running *d.colortable*, for accurate display of the map's color table.

The user should run the GRASS program *d.erase* between runs of *d.colortable* to avoid confusion.

SEE ALSO

d.colormode, d.colors, d.display, d.erase, d.rast and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.display − A menu-driven, highly interactive display program for viewing maps and producing final map products
(GRASS Display Program)

SYNOPSIS

d.display

DESCRIPTION

The GRASS program *d.display* is used to display maps on a graphics monitor and prepare final map products for printing. It is advisable to first run *d.erase* before each run of *d.display* to prepare the screen for graphics.

After invoking the command **d.display**, the DISPLAY MAIN MENU appears on the monitor. Options in this and subsequent *d.display* menus are selected through use of the pointing device (mouse). When the mouse cursor is over the desired menu selection, the user simply presses any button on the mouse.

d.display is a macro that allows the user to access a wide range of GRASS display functions in a menu-driven and highly interactive environment. Most of the functions accessible through *d.display* can also be run independently of one another by running the GRASS display (*d.*) commands listed in the SEE ALSO section, below. However, in many cases, these other *d.* commands give the user greater flexibility and make available more options than are available through the *d.display* menus. For example, the *d.display label file* option runs the *d.label* command using defaults. Users running *d.label* directly from the command line have the choice of stipulating specific label background and text colors, as well as text sizes and font types. Users running the *label file* option through *d.display* cannot specify these options; instead, default values are used (see *d.label* for default values used).

The *d.display* tool is best learned by playing with its options. Refer to the individual *d.* and other commands in the SEE ALSO section for a detailed examination of these functions.

NOTES

The user can also modify the current region settings from the DISPLAY MAIN MENU. Using the mouse, the user can choose to *zoom in* on a portion of the displayed raster map layer to create a smaller geographic region. Once the user has defined the corners of this area using the mouse, the smaller area is enlarged and replotted on the screen. The user can similarly choose to *zoom out*. In this case, the user uses the mouse to encompass that portion of the reduced map that the user wishes to make the current geographic region and display in the center of the screen. The larger map is then replotted and displayed. Finally, the user can enter geographic coordinates for the region's corners from the keyboard using the GRASS command *g.region*. This is done by selecting the *type coordinates* option.

The main menu also offers the user the option of hiding the main menu for ten seconds (*hide menu for 10 seconds*).

The *d.display* command must be run in conjunction with a program which converts the GRASS graphics commands generated by *d.display* to the graphics commands that the current device understands. This device will almost always be a graphics monitor. On most systems the user is required to start up this other program separately from (and prior to) *d.display*. In such cases, a locally-defined program, usually "monitor", is used (see your GRASS manager). (For example, on MASSCOMP machines, the *d.mon* program must be running in the foreground on the graphics monitor. On SUN machines, the *d.mon* program can be started from any terminal and run in the background.)

d.display sends program output to the current graphics frame (set, by default, to the full graphics screen unless modified by the user). You can use *d.frame* to create and choose a smaller display frame on the graphics monitor for use by *d.display*.

See the GRASS macro *3d.view.sh* stored under $GISBASE/scripts for an example of how you can create a *d.display*-like macro using a shell script.

SEE ALSO

See *3d.view.sh* or *slide.show.sh* under $GISBASE/scripts for examples of *d.display*-like macros.

d.3d, d.colormode, d.colors, d.frame, d.label, d.legend, d.measure, d.mon, d.rast, d.scale, d.sites, d.vect, d.zoom, g.region, p.menu, and *v.digit*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.erase – Erases the contents of the active display frame on the user's graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.erase
d.erase help
d.erase [**color**=*name*]

DESCRIPTION

d.erase erases the contents of the active graphics frame, and replaces it with the color black (by default) or by whatever color is specified by the user. *d.erase* will not alter the assignment of the active frame.

Parameter:

color=name	Color with which active frame will be erased. Options: red, orange, yellow, green, blue, yellow, indigo, violet, black, white, gray, brown, and magenta Default: black

SEE ALSO

d.frame, d.mon

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.font – Selects the font in which text will be displayed on the user's graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.font
d.font help
d.font font=*name*

DESCRIPTION

d.font allows the user to select use of a specific text font for display of text on the graphics monitor. The GRASS program *show.fonts.sh* is a UNIX Bourne shell macro which names and displays the fonts that can be selected using *d.font*. If the user does not specify a font when using other GRASS programs that display text, the font type *romans* is used by default.

The user can run this program either non-interactively or interactively. If the user specifies a font type name on the command line the program will run non-interactively. Alternately, the user can simply type **d.font** on the command line; in this case, the program will prompt the user for a display text font type using the standard GRASS interface described in the manual entry for *parser*.

Parameter:

font=*name* Name of a font type, from among the font types italicized below.
Default: *romans*
Options: (italized)

cyrilc	- Cyrillic
gothgbt	- Gothic Great Britain triplex
gothgrt	- Gothic German triplex
gothitt	- Gothic Italian triplex
greekc	- Greek complex
greekcs	- Greek complex script
greekp	- Greek plain
greeks	- Greek simplex
italicc	- Italian complex
italiccs	- Italian complex small
italict	- Italian triplex
romanc	- Roman complex
romancs	- Roman complex small
romand	- Roman duplex
romanp	- Roman plain
romans	- Roman simplex
romant	- Roman triplex
scriptc	- Script complex
scripts	- Script simplex

NOTES

The font type *romans* is the fastest font type to display to the graphics monitor.

SEE ALSO

d.INTRO, d.text, d.title, show.fonts.sh, and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

d.font uses the public domain version of the Hershey Fonts created by Dr. A.V. Hershey while working at the U.S. National Bureau of Standards.

NAME

d.frame − Manages display frames on the user's graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.frame
d.frame help
d.frame [**−cepsD**] [**frame**=*name*] [**at**=*bottom,top,left,right*]

DESCRIPTION

This program manages display frames on the user's graphics monitor. GRASS display programs at run-time connect with graphics rendering programs. While the display programs are identical on every hardware platform, the graphics rendering programs are (essentially the only GRASS programs) designed for individual hardware devices. These rendering programs are managed with the GRASS program *d.mon*. Graphics are displayed in rectangular frames on whatever graphics monitor the user is currently directing GRASS display output to. These frames are created and managed with this program, **Note that GRASS frame contents are not retained when one frame covers another.** You cannot shuffle frames from top to bottom and then back again. They simply define rectangular areas on the screen where subsequent drawing will occur. *d.frame.*

Flags:

-c Creates a new display frame on the graphics monitor.

-e Removes all existing display frames and reinitializes the entire graphics screen (the full-screen display frame).

-p Prints the name of the active frame, in which GRASS display output will appear.

-s Selects a frame for the display of GRASS graphics. This frame is then known as the "active frame".

-D Prints the status of the user's graphics monitor and active display frame to standard output. Information includes the name and the dimensions of the current frame on the graphics monitor, given in the form *bottom top left right*. This function is useful for debugging output, and for determining display screen coordinates.

Parameters:

frame=*name* The name of the display frame to be created/selected.

at=*bottom,top,left,right*
Where to place the frame (implies -c). Frame coordinates are stated in the form: *bottom,top,left,right*. The lower-left corner of the graphics monitor always is at location 0,0 while the monitor's upper-right corner is always at 100,100.

NOTES

If the user has created multiple display frames that overlap one another, whatever the user displays in the active frame will overwrite those portions of the underlying frame where these frames overlap.

SEE ALSO

d.erase, d.mon

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

d.geodesic − Displays a geodesic line, tracing the shortest distance between two geographic points along a great circle, in a longitude/latitude data set.
(GRASS Display Program)

SYNOPSIS

d.geodesic
d.geodesic help
d.geodesic [**coor**=*lon1,lat1,lon2,lat2*] [**lcolor**=*name*] [**tcolor**=*name*]

DESCRIPTION

d.geodesic displays a geodesic line in the active frame on the user's graphics monitor. This line traces the shortest distance between two user-specified points on the curved surface of a longitude/latitude data set. The two coordinate locations named must fall within the boundaries of the user's current geographic region.

COMMAND LINE OPTIONS

This program can be run either interactively or non-interactively. If the user types **d.geodesic** on the command line without other program parameters, the mouse will be activated; the user is asked to use the mouse to indicate the starting and ending points of each geodesic line to be drawn. The default line color (white) and text color (red) will be used.

Alternately, the user can specify the starting and ending coordinates of the geodesic, line color, and text color on the command line, and run the program non-interactively.

Once the user indicates the starting and ending coordinates of the geodesic, the line and its length (in miles) are displayed to the user's graphics monitor.

Parameters:

coor=*lon1,lat1,lon2,lat2*
Starting and ending coordinates, in longitude and latitude values, of the geodesic line to be drawn.

lcolor=*name*
Line color in which geodesic will be displayed.
Options: red, orange, yellow, green, blue, magenta, indigo, violet, gray, white, black
Default: white

tcolor=*name*
Text color in which the length of the geodesic will be displayed.
Options: red, orange, yellow, green, blue, magenta, indigo, violet, gray, white, black.
Default: red

NOTES

This program works only on GRASS data bases using a longitude/latitude coordinate system.

SEE ALSO

d.rhumbline

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

d.graph – Program for generating and displaying simple graphics to the graphics display monitor.
(GRASS Display Program)

SYNOPSIS

d.graph
d.graph help
d.graph [**input**=*name*] [**color**=*name*]

DESCRIPTION

d.graph draws graphics that are described either from standard input (default), or within a file (if an input *file* name is identified on the command line). If graphics commands are entered from standard input, a *ctrl-d* is used to signal the end of input to *d.graph*.

The program can be run interactively or non-interactively. The user can run the program completely non-interactively by specifying the name of a graphics *file* containing *d.graph* graphics commands and the values of all needed parameters on the command line. The user can instead elect to run the program partially interactively, by specifying any/all of the parameters *except* the graphics file=*name* parameter on the command line. In this case, *d.graph* will expect the user to input *d.graph* graphics commands from standard input (i.e., the keyboard) and will (silently) prompt the user for these graphics commands.

Alternately, the user can simply type **d.graph** on the command line, and be prompted for the values of all parameters (the user can still input graphics commands from either an input file or standard input using this form). In this case, the user is presented with the standard GRASS parser interface described in the manual entry for *parser*.

Parameters:

file=*name* Name of file containing graphics commands. This is a UNIX file name; the file must be located in the user's current working directory or be specified by its full path name. If no filename is specified commands are taken from the standard input.

color=*name* Starting color desired for graphics.
Options: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, and black.
Default: *white*

The coordinate system used is 0-100 in x and 0-100 in y, regardless of the graphics monitor display frame size and aspect. The (0,0) location is the lower left corner of the active graphics monitor display frame. All values may be floating point.

The graphics language is simple, and uses the following commands:

*comment*
A line of comment which is ignored in the processing.

move *xpos ypos*
The current location is updated to *xpos ypos*. Values are stated as a percent of the active display frame's horizontal (*xpos*) and vertical (*ypos*) size, and may be floating point values. Values are between 0-100. **Note.** A space must separate *xpos* and *ypos*.

draw *xpos ypos*
A line is drawn in the current color from the current location to the new location *xpos ypos*, which then becomes the current location. Values are stated as a percent of the active display frame's horizontal (*xpos*) and vertical (*ypos*) size, and may be floating point values. Values are between 0-100. **Note.** A space must separate *xpos* and *ypos*.

color *color*
Sets the current color to that stated; subsequent graphics will be drawn in the stated color, until the current color is set to a different color. Options are *red, orange, yellow, green, blue, indigo, violet, brown, magenta, gray, white*, and *black*.

size *xper yper*

Subsequent text will be drawn such that the text is ***xper*** percent of the graphics monitor display frame wide and ***yper*** percent of the frame high. By default, the text size is set to 1 percent of the active frame's width and 1 percent of the frame's height if unspecified by the user (this may be too small to be seen by the user).
Note. A space must separate ***xper*** and ***yper***.

text *line-of-text*

The stated text is drawn at the current location using the current color, and the new current location is then positioned at the end of the text string.

icon *type size x y*

Draws an icon of types o, x, or + with specified size at location x,y. Note: type o designates a square.

polygon
xpos ypos
xpos ypos
.
.
. The coordinates appearing beneath the word ***polygon***, one pair per line, circumscribe a polygon that is to be filled with the current color.

EXAMPLE

For an example use of ***d.graph***, examine the contents of the macro command ***$GISBASE/bin/grass.logo.sh*** located in the GRASS command bin (which draws a GRASS logo by inputting ***d.graph*** graphics commands that are stored in a shell file). Note that the coordinates in the ***grass.logo.sh*** macro were taken directly off an image drawn by hand on graph paper.

NOTES

d.graph remembers the last screen location (xpos ypos) to which the user moved, even after the user erases the display frame. If the user runs ***d.graph*** repeatedly, and wishes to start anew with the default (xpos ypos) screen location, the user should ***clear*** the display frame between runs of ***d.graph***.

LIMITATIONS

There are no automated ways of generating graphic images. It is anticipated that GRASS user sites will write programs to convert output from a resident graphics editor into GRASS ***d.graph*** format.

SEE ALSO

d.INTRO, d.font, d.mapgraph, d.text, grass.logo.sh, and ***parser***

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.grid – Overlays a user-specified grid in the active display frame on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.grid
d.grid help
d.grid size=*value* **[color=***name]* **[origin=***easting,northing]*

DESCRIPTION

d.grid overlays a grid of user-defined size and color in the active display frame on the graphics monitor. The grid will overlay, not overwrite, the contents of the active display frame.

d.grid can be run non-interactively or interactively. If the user specifies the grid ***size*** and (optionally) the grid ***color*** on the command line the program will run non-interactively; if no grid ***color*** is given the default will be used. Alternately, the user may simply type **d.grid** on the command line; in this case, the program will prompt the user for parameter values using the standard GRASS user interface described in the manual entry for *parser*.

Parameters:

size=*value* — Size of grid to be drawn in the active display frame, in current map coordinate system units. It should be noted that, depending on the scale of the map layer displayed in this frame, the grid may fall outside the user's viewing range.
Options: 0-100000

color=*name* — Sets the current grid color to *name*.

origin — Lines of the grid pass through this coordinate. The coordinate need not be within the current frame.
Options: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, and black.
Default: *gray*

NOTES

d.grid will not erase grids already displayed in the active graphics display frame by previous invocations of *d.grid*; multiple invocations of *d.grid* will therefore result in the drawing of multiple grids inside the active graphics frame. (A command like *d.erase*, which erases the entire contents of the active display frame, must be run to erase previously drawn grids from the display frame.)

Currently, the grid size can only be drawn in the units of the map coordinate system used by the current GRASS location.

SEE ALSO

d.display, d.erase, d.frame, d.legend, d.rast, d.scale, and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.his − Produces and displays a raster map layer combining hue, intensity, and saturation (his) values from user-specified input raster map layers.
(GRASS Display Program)

SYNOPSIS

d.his
d.his help
d.his h_map=*name* [**i_map**=*name*] [**s_map**=*name*] [**out**=*name*]

DESCRIPTION

his stands for hue, intensity, and saturation. This program produces a raster map layer providing a visually pleasing combination of hue, intensity, and saturation values from two or three user-specified raster map layers.

The human brain automatically interprets the vast amount of visual information available according to basic rules. Color, or *hue,* is used to categorize objects. Shading, or *intensity,* is interpreted as three-dimensional texturing. Finally, the degree of haziness, or *saturation,* is associated with distance or depth. This program allows data from up to three raster map layers to be combined into one new raster map layer which retains the original information in terms of *hue, intensity,* and *saturation.*

OPTIONS

This program can be run non-interactively or interactively. It will run non-interactively if the user specifies on the command line the name of a map containing hue values (*h_map*), and the name(s) of map(s) containing intensity values (*i_map*) and/or saturation values (*s_map*). If the user also names an output map (*out*) on the command line, the combined hue, saturation, and intensity values will be saved as a raster map layer in the user's current mapset; otherwise, the resulting image will only be displayed in the active display frame on the graphics monitor.

Alternately, the user can run the program interactively by typing **d.his** without naming parameter values on the command line. In this case, the program will prompt the user for parameter values using the standard GRASS parser interface described in the manual entry for *parser*.

Parameters:

h_map=*name*	Name of input raster map layer to be used for HUE values.
i_map=*name*	Name of input raster map layer to be used for INTENSITY values.
s_map=*name*	Name of input raster map layer to be used for SATURATION values.
out=*name*	Name of the output raster map layer combining hue, intensity, and saturation values from input layers.

While any raster map layer can be used to represent the hue information, map layers with a few very distinct colors work best. Only raster map layers representing continuously varying data like elevation, aspect, weights, intensities, or amounts can suitably be used to provide intensity and saturation information.

For example, a visually pleasing raster map layer can be made by using a watershed map for the *hue* factor, an aspect map for the *intensity* factor, and an elevation map for *saturation.* (The user may wish to leave out the elevation information for a first try.) Ideally, the resulting image should resemble the view from an aircraft looking at a terrain on a sunny day with a bit of haze in the valleys.

THE PROCESS

Each map cell is processed individually. First, the working color is set to the color of the corresponding cell in the map layer chosen to represent *HUE.* Second, this color is multiplied by the *red* intensity of that cell in the *INTENSITY* map layer. This map layer should have an appropriate gray-scale color table associated with it. You can ensure this by using the color manipulation capabilities of *d.display* or *d.colors*. Finally, the color is made somewhat gray-based on the *red* intensity of that cell in the *SATURATION* map

layer. Again, this map layer should have a gray-scale color table associated with it.

NOTES

This program produces an image and (optionally) a raster map layer with 1000 colors (10 intensities each of red, green, and blue). The resulting image and raster map layers will not display properly if the graphics display monitor does not have at least 1000 colors while the user is running in *float* colormode. Thus, unless the display device has 1000 colors, it is necessary to run the GRASS command

d.colormode mode=fixed

before running *d.his*. Otherwise, the colors will be incorrectly displayed on the graphics monitor.

Either (but not both) of the intensity or the saturation map layers may be omitted. This means that it is possible to produce output images that represent combinations of *his, hi,* or *hs.*

SEE ALSO

d.3d, d.colormode, d.colors, d.colortable, d.display hsv.rgb.sh, rgb.hsv.sh and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.histogram – Displays a histogram in the form of a pie or bar chart for a user-specified raster file.
(GRASS Display Program)

SYNOPSIS

d.histogram
d.histogram help
d.histogram [**-zq**] **map**=*name* [**color**=*name*] [**style**=*name*]

DESCRIPTION

d.histogram displays the category-value distribution for a user-specified raster map layer, in the form of a bar chart or a pie chart. The display will be displayed in the active display frame on the graphics monitor, using the colors in the raster map layer's color table. The program determines the raster file's category value distribution by counting cells.

OPTIONS

The user can run this program either non-interactively or interactively. The program will be run non-interactively if the user specifies the name of a raster map layer and (optionally) any other desired parameters and flags on the command line, using the form:

d.histogram [**-zq**] **map**=*name* [**color**=*name*] [**style**=*name*]

If a map name is given on the command line, any other parameter values left unspecified on the command line will be set to their default values (see below). Alternately, the user can simply type **d.histogram** on the command line, without program arguments. In this case, the user will be prompted for needed inputs and option choices using the standard GRASS user interface described in the manual entry for *parser*.

Flags:

-z Display zero-data information (cells with category value zero) in the histogram. If the -z flag is set, then cells with category value 0 in the named raster map layer will be included in the pie or bar chart. If this flag is not set, then cells with category value 0 will be excluded from the bar chart.

-q Gather the histogram quietly, without printing messages on program progress to the user's terminal.

Parameters:

map=*name* The name of an existing raster map layer in the user's current mapset search path for which a histogram is to be displayed.

color=*name* The name of the color to be used for the axis, text-labels, and tic-marks on the pie or bar chart.
Options: white, red, orange, yellow, green, blue, indigo, magenta, violet, brown, gray, and black.
Default: **color**=**white**

style=*name* The style of the histogram. If **style**=**bar**, then *d.histogram* will displays the statistics in the form of a bar chart. If **style**=**pie**, then *d.histogram* will display the statistics in the form of a pie chart.
Options: *bar* or *pie*
Default: **style**=**bar**

NOTES

d.histogram uses the current geographic region settings and the current mask (if one exists).

d.histogram uses the colors in the map's color look-up table (i.e., the map's *colr* or *colr2* file). To view results correctly on the display monitor, make sure the color mode is correctly set before running *d.histogram* (i.e., set **d.colormode**=**float**).

d.histogram does not erase the active frame before displaying output.

SEE ALSO

d.colormode, d.colors, d.colortable, d.display, d.erase, g.region, r.mask, r.stats, and *parser*

AUTHOR

Dave Johnson
DBA Systems, Inc.
10560 Arrowhead Drive
Fairfax, Virginia 22030

NAME

d.icons − Displays points, as icons, at user-defined locations in the active display frame on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.icons
d.icons help
d.icons [**-r**] **icon**=*name* [**color**=*name*] [**size**=*value*] [**points**=*name*]

DESCRIPTION

d.icons graphically displays point (site) locations as icons in the active frame on the graphics monitor. Geographic coordinates are read either from standard input or from an input file whose name is stated by the user. At each site location, a user-defined icon is displayed.

OPTIONS

The user must enter at least the name of an *icon* file storing a graphic representation of the icon to be displayed, and the geographic coordinates of the points at which they will appear.

The user can run this program either non-interactively or interactively. The program will be run non-interactively if the user specifies the name of an *icon* file and (optionally) any other desired parameters on the command line, using the form:

d.icons [**-r**] **icon**=*name* [**color**=*name*] [**size**=*value*] [**points**=*name*]

If the user fails to specify the name of a *points* file on the command line, the program will prompt the user to enter geographic coordinates from standard input. If the user specifies at least the name of an *icon* file on the command line, any other parameter values left unspecified by the user will be set to their default values (see below).

Alternately, the user can simply type:

d.icons

on the command line, without program arguments. In this case, the user will be prompted for needed parameter values using the standard GRASS user interface described in the manual entry for *parser*. The geographic coordinates at which icons are to be displayed can still be input from an input file (*points*) in interactive use.

Flag:

-r Coordinates are input in reverse order (i.e., as *northing easting*).

Parameters:

icon=*name* The name of an existing file containing a graphic representation of the icon to be drawn. *icon* files can be created by the user using the *p.icons* command and are stored under the *icons* directory under the user's current mapset.

color=*name* Sets the current icon color to the *name* stated.
Options: red, orange, yellow, green, blue, indigo, violet, gray, white, and black
Default: **color=white**

size=*value* The icon scaling factor.
Options: 1-1000
Default: **size=1**

points=*name* The name of a UNIX file containing the geographic coordinates of sites at which icons are to be drawn. Since this is a UNIX file, the user should specify the name using standard UNIX file naming conventions. If this file is not in the user's current

working directory, its pathname should be specified. The file contents should consist of a series of geographic coordinates that fall within the boundaries of the current geographic region. Each site location should be stated on a separate line as an easting and northing (in that order) separated by a single blank space. If no ***points*** file is specified by the user, input is taken from standard input and should be given in the same form; to end standard input, type ***end*** (or press control-d).

EXAMPLE

An ***icons*** file contains a graphic representation of the icon to be displayed. Here, spaces represent areas of no color, and x's represent areas of color. For example, the user might construct an icon resembling a cross in the following way:

```
  x
  x
xxxxx
  x
  x
  x
```

This icon might be stored in a file called ***cross*** (under the user's $LOCATION/icons directory). If the user specified that this be the ***icon*** file used while running ***d.icons***, this cross would then appear in the ***color*** and ***size*** specified by the user, at each site location named in the ***points*** file.

Note: icons are created with the ***p.icons*** command.

The ***points*** file lists the geographic coordinates of site locations (at which icons will be displayed). This file should take the form:

```
easting northing
easting northing
easting northing
 ...
```

If the user sets the -r flag, the order of these coordinates should be reversed (i.e., coordinates should be given as ***northing easting***).

If the points file is not specified, then the corrdinates are read from the keyboard or across a pipe. This feature allows users to enter the coordinates by hand, or, more usefully, to get then from another program. For example, to display icons at locations specified in a sites list:

```
s.out.ascii name | d.icons icon=cross
```

where ***name*** is the name of a site list, and ***cross*** is the name of an icon.

SEE ALSO

d.points, d.sites, d.where, p.icons, s.db.rim, s.out.ascii, v.db.rim, and ***parser***

AUTHOR

Contributed by:

Dave Johnson
DBA Systems, Inc.
10560 Arrowhead Drive
Fairfax, Virginia 22030

Modified by:

Michael Shapiro
U.S. Army Construction Engineering Research Laboratory

NAME

d.label – Creates and displays text labels in the active display frame on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.label
d.label help
d.label [**size**=*value*] [**backcolor**=*name*] [**textcolor**=*name*] [**font**=*name*]

DESCRIPTION

d.label allows the user to create and display text labels in the active frame on the graphics monitor. It interactively requests that the user type in text and use the pointing device (mouse) to identify where this text is to be placed within the active graphics frame. The program will prompt the user for label size, background color, text color, and text font type, if the user fails to specify these values on the command line. Program prompts use the standard GRASS parser interface described in the manual entry for *parser*.

Parameters:

size=*value* Sets the label text size to the specified number. Values are stated as a percentage of the frame height; e.g., a size of 10 will make each line of text equal to one-tenth the height of the display frame.
Options: 0-1000
Default: 10

backcolor=*name* Sets the color of the label background to the *name* stated.
Options: red, orange, yellow, green, blue, indigo, violet, gray, white, and black.
Default: *black*

textcolor=*name* Sets the color of the label text to the *name* stated.
Options: same as for *backcolor*.
Default: *white*

font=*name* Sets the font type used for the label's text to *name*.
Options: romand, romanp, romant, romans, scriptc, scripts, romancs, italicc, italiccs, gothitt, gothgrt, and gothgbt. The user can view these available fonts by running *show.fonts.sh*.
Default: *romans* (Roman simplex font type), or whatever font the user has set the font type to before entering *d.label*.

NOTES

This program will allow the user to display labels anywhere in the active display frame, even in areas lying outside of the current geographic region.

d.label will only allow the user to type in one line of text. The fully interactive program *d.labels* can be used to create text labels having as many as four lines.

d.label does not create paint labels files displayable by *d.paint.labels* -- *d.labels* does. These programs should be integrated in future GRASS releases.

BUGS

If the user is running GRASS under X Windows, and the user enters a text label longer than 80 characters (i.e., allows text to wrap onto a second line), X Windows will stop the user's GRASS graphics monitor without throwing the user out of GRASS. If this happens, the user should simply restart a graphics monitor, using *d.mon*.

SEE ALSO

d.display, d.font, d.labels, d.legend, d.mon, d.paint.labels, d.rast, d.scale, d.text, d.title, d.where, show.fonts.sh and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.labels – To create/edit GRASS paint label files for display on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.labels

DESCRIPTION

The *d.labels* program allows the user to interactively create and modify paint-labels files for use with the *p.map* and *d.paint.labels* programs. The *paint labels* files created will be stored in the user's current mapset under the */paint/labels* directory.

d.labels is fully interactive, and requires use of a graphics monitor. After the user has typed

d.labels

and has typed in the name of a new or existing paint-labels file, the user is asked to indicate label positions on the display frame using the mouse-pointer, and to edit label parameters that define the label's size, color, and other characteristics through use of a visual-ask (VASK) data entry screen. The mouse can also be used to indicate the geographic coordinates of locations within the current geographic region in the active frame. It is helpful to display a relevant data file in the active display frame before running *d.labels* to provide visual clues for label placement.

USER PROMPTS

You will first be prompted to enter the name of either an old paint-labels file that you wish to modify, or a new paint-labels file that you wish to create. The following is the file name prompt:

```
OPTION: Label text file name

key: name
required: YES
enter option >
```

If you enter the name of a new (non-existent) paint-labels file, you are asked to indicate a location falling within the current geographic region in the active display frame at which the label is to be placed.

```
Mouse Buttons
 Left:   Place Label Here
 Middle: Where Am I ?
 Right:  Quit
```

Once you have indicated a position, you will be presented with a data-entry screen (also known as a VASK screen) so that you can enter the paint-label parameters that define how the label will appear on the display or on the color printer. Once you finish entering the label-parameters, and you hit the <ESC> key to proceed, the label will be displayed on the screen. If you like the appearance of the label, enter "y" to the "Look OK?" prompt; if not, enter "n" and you will again be presented with the data entry screen. Once you approve the label's appearance you will be asked to indicate the position of the next label.

If you choose to display an already-existing paint-labels file, you will be shown the labels. You will then have the option of editing a label or adding more labels.

```
Buttons
 Left:   Edit Existing Labels
 Middle: Quit
 Right:  Add more Labels
```

On-line help is available. If you are puzzled about what should go in a data entry field, just enter "help" in that field, hit the <ESC> key, and you will be presented with a short description of the field and what you may enter.

The following is a list of the different fields in the data entry screen and a short description of each.

TEXT: The user can enter up to four lines of text.

SKIP: yes/no. If *no*, label will be printed. If *yes*, the label will be retained in the file but not printed.

LOCATION: Determines where the text will be located. The user specifies the easting and northing in map units, and (optionally) specifies a vertical and horizontal offset (in printer pixels) from the specified easting/northing. (The vertical (y) offset will shift the location to the south if positive, north if negative. The horizontal (x) offset will shift the location east if positive, west if negative.) These offsets are provided to allow finer placement of labels.

RESET LOCATION: yes/no. If *yes*, the user will be allowed to use the mouse to select a new location for the label. The mouse buttons will function as follows:

Mouse Buttons
Left: Place Label Here
Middle: Where Am I ?
Right: Quit

PLACEMENT: Determines which part of the label to which the location refers. This may be specified as:

lower left	(lower left corner of the text)
lower right	(lower right corner of the text)
lower center	(bottom center of the text)
upper left	(upper left corner of the text)
upper right	(upper right corner of the text)
upper center	(top center of the text)
center	(center of the text)

TEXT SIZE: Determines the size of the lettering used in the label. Text size (height) is stated in units of (ground) meters; e.g., a text size of 500 sets the height of each line of text equal to 500 ground meters. Thus, text will appear to grow or shrink when displayed, depending on the user's current geographic region settings.

TEXT COLOR: This selects the text color. This color can be specified in one of three ways:

1) By color name:
aqua black blue brown cyan gray green grey indigo magenta orange purple red violet white yellow

2) As red, green, blue percentages. For example: .5 .4 .7
(This form is not supported by ***d.paint.labels***, but see ***p.map*** and ***p.labels***.)

3) By printer color number, to get the exact printer color.

(This form is not supported by ***d.paint.labels***, but see ***p.map*** and ***p.labels***.)

Note. If the user sets the text color equal to the background color, the text will not appear on the graphics monitor.

WIDTH: This determines the line thickness of the letters. The normal text width should be set to 1. Larger numbers can be used to simulate bold face. (***d.paint.labels*** ignores this value and always uses 1.)

HIGHLIGHT COLOR: The text can be highlighted in another color so that it appears to be in two colors. The text is drawn first in this color at a wider line width, and then redrawn in the text color at the regular line width. Highlight colors are not shown on the graphics display, they are only shown in the output from ***p.labels*** and ***p.map***.

HIGHLIGHT WIDTH: Specifies how far from the text lines (in pixels) the highlight color should extend. Highlight colors are not shown on the graphics display, they are only shown in the output from ***p.labels*** and ***p.map***.

BACKGROUND COLOR: Text may be boxed in a solid color by specifying a background color (see TEXT COLOR above to specify a color). Specify ***none*** to use no background.

OPAQUE TO VECTORS: ***yes/no***. This field only has meaning if a background color is selected. ***yes*** will prevent vector lines from entering the background. ***no*** will allow vector lines to enter the background.

BORDER COLOR: Select a color for the border around the background. Specify ***none*** to suppress the border.

NOTES

This program is fully interactive and requires no command line arguments.

If the user types the word **help** in a data entry field while entering label information, acceptable field choices will be listed.

Labels must be placed within the current geographic region. If the user attempts to place a label outside of this region, the program will appear to loop endlessly. It is simply failing to display a label positioned beyond the boundaries of the current region; in this event, the user should simply reposition the label.

Labels created by this program are stored in ASCII files under the directory $LOCATION/paint/labels.

SEE ALSO

d.label, d.paint.labels, d.rast, d.vect, d.where, p.labels, p.map

AUTHOR

Dave Johnson
DBA Systems, Inc.
10560 Arrowhead Drive
Fairfax, Virginia 22030

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

d.legend – Displays a legend for a raster map layer in the active frame on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.legend
d.legend help
d.legend map=*name* [**color**=*name*] [**lines**=*value*]

DESCRIPTION

d.legend displays a legend for a user-specified raster map layer in the active frame on the graphics monitor. The legend's size is based on the height of the active frame. The user should therefore take care to create a display frame of suitable dimensions before running *d.legend*. *d.legend* will only obscure those portions of the active frame that directly underlie the legend.

The user can run *d.legend* either non-interactively or interactively. If the user specifies the name of a raster *map* layer on the command line, the program will run non-interactively. Default legend text *color* and number of *lines* will be used unless the user specifies other values on the command line.

Alternately, the user can simply type **d.legend** on the command line; in this case, the program will prompt the user for parameter values using the standard GRASS parser interface described in the manual entry for *parser*.

Parameters:

map=*name* Name of a raster map layer whose legend is to be displayed in the active display frame.

color=*name* Sets the legend text color to the *name* stated.
Options: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, and black.
Default: *white*

lines=*value* Number of lines to appear in the map legend.
The number of lines refers to the maximum number of lines of type that can be displayed given the height of the active display frame. If unspecified by the user, the program will divide the display frame into the number of lines required to display all of the category labels and colors associated with the named *map*. To decrease the size of the displayed text, increase the number of lines.
Options: 1 - 1000
Default: Set *lines* value equal to the number of *map* categories

NOTES

The legend text size is based on the number of *lines* requested (or, by default, on the number of lines needed to display the legend). If the user attempts to display a very long legend in a relatively short display frame, the legend may appear in unreadably small text.

SEE ALSO

d.colormode, d.colors, d.colortable, d.display, d.erase, d.font, d.frame, d.grid, d.label, d.labels, d.rast, d.scale, and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.mapgraph − Generates and displays simple graphics on map layers drawn in the active graphics monitor display frame.
(GRASS Display Program)

SYNOPSIS

d.mapgraph
d.mapgraph help
d.mapgraph [**input**=*name*] [**color**=*name*] [**vsize**=*value*] [**hsize**=*value*]

DESCRIPTION

d.mapgraph draws graphics that are described in standard input (default) or the UNIX input file *name*. If commands are entered via standard input, a *ctrl-d* is used to signal the end of input to *d.mapgraph*. This program performs essentially the same function as *d.graph*; however, point locations are specified to *d.mapgraph* in the geographic coordinate system of the user's current mapset and location (i.e., in map coordinates), rather than in graphics display screen coordinates.

The program can be run interactively or non-interactively. The user can run the program completely non-interactively by specifying the name of a file containing *d.mapgraph* graphics commands and the values of all needed parameters on the command line. The user can instead elect to run the program partially interactively, by specifying any/all of the parameters *except* the graphics input=*name* parameter on the command line. In this case, *d.mapgraph* will expect the user to input *d.mapgraph* graphics commands from standard input (i.e., the keyboard) and will (silently) prompt the user for these graphics commands. Alternately, the user can simply type **d.mapgraph** on the command line, and be prompted for the values of all parameters (the user can still input graphics commands from an input file using this form.) In this case, the user is presented with the standard GRASS parser interface described in the manual entry for *parser*.

Parameters:

input=*name* Name of a UNIX file containing graphics instructions. Specify the full path name of the file if not in the current directory. If no file name is given, commands are taken from standard input.

color=*name* Starting color desired for graphics.
Options: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, and black.
Default: *white*

d.mapgraph is used for drawing simple graphics on top of map layers. The coordinate system used by *d.mapgraph* is the same as that of the map layer displayed in the active display frame on the graphics monitor (or that of the user's current region, if no map is displayed).

The graphics language is simple and uses the following commands:

*comment*
A line of comment which is ignored in the processing.

move *xpos ypos*
The current location is updated to *xpos ypos* (where these, respectively, are the easting and northing of geographic coordinates stated in the map coordinate system of the user's current GRASS location, falling within the current region and active frame). If unspecified by the user, the current location becomes (0,0). If, as most likely, the point (0,0) falls outside of the user's current region, graphics drawn there will not appear in the graphics frame.
Note: use *g.region* to obtain the coordinates of current location. Use *d.where* to obtain specific map coordinates of various points on the raster map displayed in the active frame.
Note: there must be a space between *xpos* and *ypos*.

draw *xpos ypos*
A line is drawn in the current color from the current location to the new location *xpos ypos*, which then becomes the current location. *xpos* and *ypos* are (respectively) an easting and northing stated in the map coordinate system of the user's current GRASS location, and located within the user's

current geographic region and active frame.
Note: there must be a space between *xpos* and *ypos*.

color *color*

Sets the current color to that stated. Color options are: ***red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white***, and ***black***.

size *xper yper*

Subsequent text will be drawn such that the text is *xper* percent of the display frame's width and *yper* percent of the display frame height. If not specified by the user, the text size becomes 5 percent of the active frame's width and 5 percent of the frame's height. This is equivalent to entering **size 5 5**.

text *line-of-text*

The stated text is drawn at the current location using the current color and the current size.

icon *type size x y*

Draws an icon of types o, x, or + with specified size at location x,y. Note: type o designates a square.

polygon

xpos ypos
xpos ypos
.
.
.

The map coordinates appearing on lines beneath the word *polygon*, one pair per line, circumscribe a polygon which is to be filled with the current color.

NOTES

d.mapgraph is identical to the *d.graph* command, except for the difference in coordinate systems used. *d.mapgraph* will complain if the user enters something to standard input that it does not understand. Blank lines in the input file will result in this error message.

SEE ALSO

d.INTRO, d.frame, d.graph, d.rast, d.zoom, g.region, grass.logo.sh, and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.measure – Measures the lengths and areas of features drawn by the user in the active display frame on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.measure
d.measure help
d.measure [**c1**=*name*] [**c2**=*name*]

DESCRIPTION

d.measure provides the user with an interactive way to measure the lengths and areas of lines and polygons drawn by the user in the active frame on the graphics monitor. Lines and polygons are drawn using a pointing device (mouse). Each line segment is drawn in colors *c1* and *c2*. A mouse button menu indicates that the user can find out the geographic coordinates of the cursor location, draw line segments between user-specified vertices, and quit *d.measure*. Lines and polygons are defined by the series of vertices marked by the user. If more than two successive vertices are drawn, *d.measure* prints the area encompassed (*d.measure* will assume the area is closed even if the user has not joined the first and last vertices). Line lengths are stated in the same units as those of the current LOCATION. Areas are stated in hectares, square miles, and square meters.

Lines and polygons drawn using *d.measure* will overlay (not overwrite) whatever display currently appears in the active frame on the graphics monitor. The user can, for example, run *d.rast* or *d.vect* prior to running *d.measure*, and use raster and/or vector maps as a backdrop.

OPTIONS

The user can specify the colors in which line segments will be drawn by setting the values of *c1* and *c2* on the command line. Default line colors (see below) will be used if the user does not specify the values of *c1* and *c2* on the command line.

Parameters:

c1=*name* — The first color in which each line segment is drawn, while being positioned.
Options: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, and black
Default: *white*

c2=*name* — The second color in which each line segment is drawn, after its vertices are fixed.
Options: Same as *c1*.
Default: *white*

NOTES

d.measure uses all segments drawn by the user when computing area. If the user draws an area within another area, the combined area of both regions will be output.

SEE ALSO

d.display, d.frame, d.graph, d.mapgraph, d.rast, d.vect, d.where

AUTHOR

Jim Westervelt, U.S. Army Construction Engineering Research Laboratory
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory, 4.1 Release

NAME

d.menu – Creates and displays a menu within the active frame on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.menu
d.menu help
d.menu [**bcolor**=*name*] [**tcolor**=*name*] [**dcolor**=*name*] [**size**=*value*]

DESCRIPTION

d.menu allows the user to create a menu containing a title and options, and to display this menu in the active frame on the graphics monitor. After the menu is displayed in the active frame, the pointing device (mouse) is activated and must be used to select one of the menu options. The number associated with the selected menu option is then printed to standard output and the program exits. This program provides GRASS macro writers with a mouse interface for users.

Parameters can be stated on the command line, from within standard input, or (as illustrated in EXAMPLE) from within a script file.

The user can specify the menu's background, text, and line colors (*bcolor*, *tcolor*, and *dcolor*) and the menu size (*size*) on the command line. If the user sets at least one of these values on the command line, any remaining values that are not specified on the command line will be set (automatically) to their default values (see below). Alternately, the user can simply type **d.menu** on the command line; in this case, the program will prompt the user for the menu colors and text size using the standard GRASS parser interface described in the manual entry for *parser*. The user will then be (silently) prompted to enter the menu's location and contents through standard input (see Menu Information, below).

Parameters:

bcolor=*name* Sets the color of the menu background (the rectangle on which the text sits).
Options: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, and black.
Default: *black*

tcolor=*name* Sets the color of the menu text.
Options: Same as those for *bcolor*.
Default: *white*

dcolor=*name* Sets the line color used to divide text lines in the menu.
Options: Same as those for *bcolor*.
Default: *white*

size=*value* Sets the menu's text size to the specified value. Sizes are stated as percentages of the height of the active graphics display frame.
Options: 1 - 100
Default: 3 (i.e., 3%)

Menu Information:

After the user has (optionally) specified menu colors and size, the program expects the user to enter information about the menu's location and content. The menu will be placed in the lower left corner of the active display frame by default if the user does not position it elsewhere using the .T or .L commands.
The user specifies the menu contents by typing in a menu title followed by the option choices to appear in the menu when displayed. The user ***must*** enter a menu title and at least one menu option.

.T *value* Specifies the menu's distance from the active display frame's top edge (as a percentage of the active frame's height).
Note: Not required

.L *value* — Specifies the menu's distance from the active display frame's left edge (as a percentage of the active frame's width).
Note: Not required

menu title — A title that describes the type of options listed in the menu, and that will appear at the top of the menu when it is displayed.

option name(s) — The options that will appear in the menu when displayed. Each menu option should appear on a separate line. The user may enter as many options as desired, but must enter at least one menu option.
Note: The user should choose a menu size and location that will allow all menu options to be displayed in the active frame.

If the user enters the menu title and option(s) from standard input (i.e., at the keyboard rather than from a file), the user should enter *control-d* to end input and display the menu in the active frame on the graphics monitor. (Note: The *d.menu* program can also be incorporated into UNIX Bourne shell script macros. The below example shows how this might be done.)

EXAMPLE

In the following example, the shell script *menu2* calls the shell script *color.select* which contains *d.menu* commands to display a menu in the current frame on the graphics monitor. After the user selects an option from the display menu, the selection number is available for use by *menu2*.

Contents of file *menu2*:

```
#! /bin/csh -f
set option = 0
set colors = (red green blue black white )
@ option = `color.select`

if ($option <= 5) then
        set color = $colors[$option]
        echo $color

endif

exit
```

Contents of file *color.select*:

```
#! /bin/csh -f
d.menu bcolor=red tcolor=green dcolor=yellow size=5 << EOF
.T 25
.L 25
Color Choices
Option 1
Option 2
Option 3
Option 4
Option 5
EOF
```

If the user runs *menu2*, a menu will be displayed on the graphics monitor that has red background, green text, with menu options divided by yellow lines, and a text size of 5% of the active display frame height. The mouse cursor will become active, allowing the user to select (by pointing with the mouse) one of the displayed menu options. Here, these menu options are called *Option 1*, *Option 2*, and *Option 3*, etc. The first line of text (here, the words *Color Choices*) contains the title of the menu; this line is **not** a menu option that can be chosen by the user with the mouse. When the user presses one of the mouse buttons while pointing to the desired menu choice, the number of the option chosen will be available for capture by

the shell script *menu2*. *menu2* is a simple example that takes this information and only echoes it to the screen.

NOTES

Although the user can vary text size, all text within the same menu is displayed in a single text size (and font). If the user specifies that items included in the menu's text be displayed in different sizes, all text will be displayed in the size stated last.

SEE ALSO

d.display, d.font, d.frame, d.grid d.label, d.legend d.paint.labels, d.text, d.title, show.fonts.sh and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.mon – To establish and control use of a graphics display monitor.
(GRASS Display Program)

SYNOPSIS

d.mon
d.mon help
d.mon [**-lLprs**] [**start**=*name*] [**stop**=*name*] [**select**=*name*] [**unlock**=*name*]

DESCRIPTION

The GRASS program ***d.mon*** allows the user to start, select, list, query the status of, release control of, stop, and unlock control of, available graphics monitors. The user can run this program either interactively (through a series of menus), or non-interactively by typing the name of the monitor to start, stop, select, or unlock, and any desired flags on the command line.

Flags:

-l List all monitors

-L List all monitors (with current status)

-p Print name of currently selected monitor

-r Release currently selected monitor

-s Do not automatically select when starting

Parameters:

start=*name* Name of graphics monitor to start

stop=*name* Name of graphics monitor to stop

select=*name* Name of graphics monitor to select

unlock=*name* Name of graphics monitor to unlock

When the user enters the command **d.mon** without specifying parameter values, the below menu appears on the screen:

MONITOR MENU

Making sure that the graphics monitor is running:

1 - Start a graphics monitor
(also automatically selects this monitor)
2 - Stop a graphics monitor

Choosing a graphics monitor for your graphics:

3 - Select a graphics device for output
(currently selected monitor: *name*)
4 - Release control of the graphics driver
(let someone else use it)
(option 4 appears only after selection of a monitor)

RETURN quit

These parameters perform the following functions:

1 **Start a Monitor.**
In order to display on-screen GRASS graphics, the user must ***start*** and ***select*** a graphics monitor. By default, the ***start*** command actually runs two commands, to both start and select whatever monitor is named by the user. (The user can get a list of available monitors by setting the -l or -L flag on the command line.)

When a monitor is ***started***, it is therefore also (automatically) ***selected*** for output, unless the ***-s*** flag is set by the user; the user can also explictly ***select*** a monitor that has been started (see (3) below). After a monitor is started, a blank graphics frame should appear on whatever terminal the user is using to display graphics.

The desired monitor should be started once and need not be restarted unless it is stopped (option *2*) for some reason. A monitor may continue to run for any length of time, even when no GRASS session is being run. The monitor program runs in the background.

2 **Stop a Monitor.**
Sometimes the monitor program needs to be stopped (terminated). Choosing option 2 will terminate a user-specified monitor program.

A graphics monitor has two different types of status: monitor program ***not running***, and monitor ***running***. A monitor that has been started and/or selected will be listed as running; a monitor that has been stopped (or not started) will be listed as not running. The -L (list status) flag will list the status of each monitor connected to the system.

3 **Select a Monitor.**
When the user ***starts*** a monitor, it is also (automatically) ***selected*** for graphics output unless the user sets the ***-s*** flag. In order to use (direct graphics output to) a monitor, the user must ***select*** that monitor for use, either by simply starting the monitor without the -s flag or by explictly selecting the monitor for output using option 3. Only running monitors can be selected for graphics output. Once the user has ***selected*** a monitor for output, no other user can use this monitor for graphics output until the monitor driver is either ***released*** (by the user) or ***unlocked*** (by any user on the system).
The user can run multiple graphics monitors by simply starting each of the graphics monitors drivers he wishes to direct output to.

4 **Release a Monitor.**
Once a user has ***selected*** a monitor for graphics output, it is locked for use by that user until either: (1) the user voluntarily releases control of the monitor for use by another (option 4), or (2) another GRASS user ***unlocks*** the user's control of the monitor. Menu option 4 appears only to the person who has selected the monitor (since only that user can release control of his selected graphics monitor.) If another user wishes to ***unlock*** the user's control of the monitor, that user must run ***d.mon*** from the command line and set the unlock=***name*** parameter.

You may choose multiple options within the ***d.mon*** program. To leave (exit) the ***d.mon*** menu, press the <RETURN> key.

NOTES

The ***d.mon*** program can regulate control of graphics monitors both in systems using multiple monitors and in systems using a single graphics monitor.

SEE ALSO

d.erase, d.frame, monitorcap

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

d.paint.labels – Displays text labels formatted for use with GRASS paint (*p.labels*, *p.map*) output to the active frame on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.paint.labels
d.paint.labels help
d.paint.labels file=*name*

DESCRIPTION

d.paint.labels displays the *paint* label file *name* in the active display frame on the graphics monitor. This file can be generated by the *labels* option in the *p.labels* program, the *d.labels* program, or simply created by the user as an ASCII file (using a text editor like *vi*) and placed in the appropriate */paint/labels* directory under the user's current mapset and location (i.e., under $LOCATION/paint/labels).

OPTIONS

Parameter:

file=*name* Name of ASCII file containing paint labels.

This program can be run either non-interactively or interactively. If the user specifies the name of a paint labels file on the command line, the program will run without interacting with the user. Alternately, the user can type simply **d.paint.labels** on the command line; in this case, the program will prompt the user for the name of a paint labels file using the standard GRASS parser interface described in the manual entry for *parser*.

NOTES

Some of the information stored in the label file is unused by *d.paint.labels*. The unused information found in a label file is printed to standard output. This extra information is used by such programs as *p.labels* and *p.map*.

SEE ALSO

d.display, d.font, d.label, d.labels, d.text, d.title, p.labels, p.map and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.points – Displays point graphics in the active frame on the graphics display monitor.
(GRASS Display Program)

SYNOPSIS

d.points
d.points help
d.points [**color**=*name*] [**size**=*value*] [**type**=*name*] [**file**=*name*]

DESCRIPTION

d.points displays points whose coordinate locations are specified by the user and located within the current geographic region in the active display frame on the graphics monitor. Point coordinates are read either from standard input or from a file stated by the file=*name* option. When entering coordinates through standard input, the user presses *control-d* to signal the end of input to *d.points*.

The program can be run interactively or non-interactively. The user can run the program completely non-interactively by specifying the name of a points *file* (containing the geographic coordinates of icons to be sited) and the values of all needed parameters on the command line. If the name of an input *file* to *d.points* is given on the command line, any other parameter values left unspecified on the command line (e.g., the icon *color*, *size*, and/or *type*) are set to their default values.

The user can instead elect to run the program partially interactively, by specifying any/all of the parameters *except* the points file=*name* parameter on the command line. In this case, *d.points* will expect the user to enter icon locations from standard input (i.e., at the keyboard) and will prompt the user for these locations. Each icon location should be entered as an easting and northing pair separated by a space, followed by a carriage return. Coordinates should be stated in whatever map coordinate system is used by the user's current GRASS location. The user presses *control-d* to signal the end of standard input to *d.points*.

Alternately, the user can simply type **d.points** on the command line, and be prompted for the values of all parameters (the user can still input icon coordinates from an input file using this form). In this case, the user is presented with the standard GRASS parser interface described in the manual entry for *parser*.

Parameters:

color=*name* Sets the current color to the *name* stated.
Options: red, orange, yellow, green, blue, indigo, violet, gray, white, and black
Default: *gray*

size=*value* Size of icon in pixels.
Options: 0-1000 (pixels)
Default: 5 (pixels)

type=*name* Sets the shape (type) of the icon to be drawn to *name*.
Options: x, diamond, box, +
Default: +

file=*name* Take input from the UNIX file *name*. (Note: This file must be in the user's current directory or given by its full path name.)
Default: Standard input.
The file contents should consist of a series of geographic coordinates that fit within the boundaries of the current geographic region falling within the active display frame (refer to *d.where* or *g.region* for these coordinates). Each icon location should be specified by an easting and northing (in that order) separated by a single blank space, and followed by a carriage return. Eastings and northings should be stated in whatever map coordinate system is being used by the user's current GRASS location.

NOTES

d.points will display a point at the location specified in the active frame, even if the point lies outside the boundaries of current geographic region.

SEE ALSO

d.frame, d.where, d.sites, g.region, s.out.ascii, and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.profile − Displays profiles of a user-specified raster map layer.
(GRASS Display Program)

SYNOPSIS

d.profile

DESCRIPTION

This command works only interactively. It clears the entire graphics screen and provides a graphical interaction allowing the selection of transects for which profiles are then generated.

USER PROMPTS

First, you will be presented with a prompt asking you to choose a raster map layer to be profiled. Once you specify a valid raster map layer name, the map layer will be displayed in the left half of the graphics display, and the right half of the dispay will be divided into four display frames. There will also be two frames along the top of the display: a mouse-button menu frame on the left, and a status frame on the right.

The mouse-button menu initially offers you three options:

Mouse Buttons:
Left: Where am I?
Middle: Mark FIRST Point of Profile Line.
Right: QUIT this.

You may query the displayed raster map layer by indicating points with the left mouse-button. The coordinates and category value of each point that you indicate will be displayed on in the status frame. If you mark the first point of the profile line you will be presented with the following mouse-button menu:

Mouse Buttons:
Left: Where am I?
Middle: Mark SECOND Point of Profile Line.
Right: QUIT this.

Once you mark the second point of the profile line, the profile line will be labeled (with a letter from A to D) and displayed in one of the four display frames on the right hand side of the screen. You will then be presented with a third mouse-button menu:

Mouse Buttons:
Left: DO ANOTHER
Middle: CLEAR DISPLAY
Right: QUIT this.

If you would like to view another profile, click on the left mouse-button. If you would like to redisplay the raster map layer and clear out the four profile frames, click on the middle mouse-button. If you would like to quit, then click on the right button.

NOTES

Useful enhancements to *d.profile* would include:

1) Adding an option to display profiles using category colors, like a bar-chart.
2) Allowing profile lines to be defined by a series of points, not just two.
3) Allowing profiles to be saved in a file, for later viewing.
4) Allowing the user to enter profile line points by typing coordinates.

AUTHOR

Dave Johnson
DBA Systems, Inc.

10560 Arrowhead Drive
Fairfax, Virginia 22030

NAME

d.rast – Displays and overlays raster map layers in the active display frame on the graphics monitor. *(GRASS Display Program)*

SYNOPSIS

d.rast
d.rast help
d.rast [**-o**] **map**=*name*

DESCRIPTION

d.rast displays the raster map layer *name* in the active display frame on the graphics monitor.

The program can be run either non-interactively or interactively. If the user specifies the name of a raster map layer (map=*name*) and (optionally) the -o option on the command line, the program will run non-interactively. Alternately, the user can type simply **d.rast** on the command line; in this case, the program will prompt the user for the flag setting and parameter value using the standard GRASS parser interface described in the manual entry for *parser*.

Flag:

-o Overlay the named raster map layer onto whatever is already displayed in the active graphics frame. Any zero category value data areas in the named raster map will seem transparent, and reveal the underlying image previously displayed in the graphics frame. If the -o flag is set, only cells containing non-zero category values will be displayed from the *overlaid* raster map. All other areas (i.e., the portions of the overlaid map that contain category value 0) will leave the underlying display untouched. If the -o flag is not set by the user, *d.rast* will (by default) completely overwrite whatever appears in the active graphics display frame.

Parameter:

map=*name* Name of the raster map to be displayed. If the active graphics frame already contains text or graphics, and the user does not wish to use the -o option, it is wise to first invoke *d.erase* to clear the active graphics frame before running *d.rast*. After running *d.rast*, other *d.* programs like *d.vect* and *d.grid* can be used to enhance the plot.

NOTES

The *d.rast* raster map overlay option (-o) only works when the color look-up table for the graphics monitor is set to **d.colormode fixed**.

SEE ALSO

d.choose, d.colormode, d.colors, d.colortable, d.display, d.erase, d.grid, d.overlay, d.vect, d.what.rast, r.reclass and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.rast.arrow – Draws arrows representing cell aspect direction for a raster map layer.
(GRASS Display Program)

SYNOPSIS

d.rast.arrow
d.rast.arrow help
d.rast.arrow [map=*name*] [**type**=*name*] [**arrow_color**=*name*] [**grid_color**=*name*] [**x_color**=*name*] [**unknown_color**=*name*]

DESCRIPTION

d.rast.arrow is designed to help users better visualize surface water flow direction, as indicated in an aspect raster map layer. There are two ways to specify the aspect layer the program is to use. The first is to display the aspect map layer on the graphics monitor before running *d.rast.arrow*. The second method involves setting the *map* parameter to the name of the desired aspect map. This allows the arrows to be drawn over any other maps already displayed on the graphics monitor.

d.rast.arrow will draw an arrow over each displayed cell to indicate in which direction the cell slopes. An arrow can point in one of eight directions. If the aspect layer has a category value denoting locations of "unknown" aspect, *d.rast.arrow* draws a question mark over the displayed cells of that category. Cells of category 0 (no data) will be marked with an "X".

Note: Arrows are cell size and cannot be seen if the raster map is relatively close in scale. From a command line mode, the user cannot run d.rast.arrow and then run d.rast.zoom and still see the arrows. The zooming action erases the arrows and the user must run .arrow again.

COMMAND LINE OPTIONS

Parameters:

map=*name* Name of an existing raster map layer to be displayed.

type=*name* Type of existing raster data to be displayed.
Options: grass, agnps, answers
Default: grass
Using this information, *d.rast.arrow* uses internal information to convert category values into appropriate arrow directions.

arrow_color=*name* Color in which arrows will be drawn.
Options: white, red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, black
Default: green

grid_color=*name* Color in which grid outlines will be drawn.
Options: white, red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, black
Default: gray

x_color=*name* Color in which x's will be drawn.
Options: white, red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, black
Default: white

unknown_color=*name*
Color in which unknown information will be displayed.
Options: white, red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, black
Default: red

SEE ALSO

d.frame, d.rast, d.rast.edit, d.rast.num, d.rast.zoom, d.zoom, g.region, r.mask, r.slope.aspect, and *parser*

AUTHOR

Chris Rewerts (rewerts@ecn.purdue.edu), Agricultural Engineering, Purdue University.

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

d.rast.edit − Program allowing users to interactively edit the cell category values of raster map layers displayed on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.rast.edit
d.rast.edit help
d.rast.edit [**grid_color**=*name*]

DESCRIPTION

The *d.rast.edit* program allows users to interactively edit cell category values in a raster map layer displayed to the graphics monitor using a mouse cursor. This program determines the name of the raster map layer currently displayed in the active frame on the selected graphics monitor (if none, program will abort). The user is then prompted for the name of a new raster layer to be created in his mapset. *d.rast.edit* does not modify the user's original raster map layer.

OPTIONS

Parameter:

grid_color=*color* Sets the color to be used for a grid drawn during edit mode. A well-selected grid color helps to identify individual cells in the output.
Options: red, orange, yellow, green, blue, indigo, violet, gray, white, and black
Default: black

Geographic Region Concerns

d.rast.edit reads the region definition for the raster map layer being edited from its cell header file. The new, edited copy of the raster layer will be created with the same resolution and region dimensions. If the current region resolution does not match the raster map layer's resolution, the program will abort. The north, south, east, and west geographic region settings of the current region can be set to any view, as long as they fall within the boundaries of the raster map layer described in its cell header. This is important, since the current view must be such that individual cell locations are easily visible and sufficiently large on the graphics monitor to be pointed at with the mouse cursor. Any mask in place will be ignored when writing the new raster map layer.

Mouse Menus

Cell editing is done using a mouse cursor to choose menu selections and indicate areas on the displayed raster map that are to be edited.

Selections available via the *Main Menu* and the subsequent functions or sub-menus are described below:

1. Edit

invokes edit mode, during which no graphic menus are used, so that none of the displayed map is hidden. At the start of the edit mode, a grid is drawn over displayed cells to help distinguish cell boundaries. Users can select the color of this grid by setting the *grid_color* parameter. Interaction with the program during edit mode is done by using the three mouse buttons as follows:

Left Button

what's here. Identify the category value of the cell under the mouse cursor when the button is pressed. The category value of this cell is printed in the text frame.

Middle Button

edit. The current category value of the cell under the mouse cursor is iterated and a prompt for a new cell value appears on the textual command frame. (Remember to move the mouse to the text frame). Enter a new cell category value which is within the range of current category values for the map layer. Note: the edited cell is displayed in the color of the newly-assigned category

value, but is hatched with lines of the grid color to indicate it has been edited, since, if the value of the cell is polled (before leaving edit mode), the old category value will be reported.

Right Button

exit edit mode. Command is returned to the *Main Menu* on the graphics monitor. If cell value changes were made during edit mode, they will be saved upon exit.

2. Redraw

redisplays the raster map layer on the graphics monitor.

3. Zoom

calls the *d.rast.zoom* program, to allow changing the view of the raster layer. **Zoom's** instructions will appear on the text command window. Interaction is conducted using mouse keys.

4. Arrow

This function is available for users editing aspect maps. The *d.rast.arrow* program is called, which draws arrows over the displayed cells to indicate the downslope direction of the cell. After selecting the *arrow* selection from the main menu, sub-menus will appear, allowing input options to be set for the *d.rast.arrow* program. The first sub-menu selects the type of aspect data displayed. (regular "GRASS" format, as produced by *r.slope.aspect*; a format prepared as input to the "AGNPS" program; and a format prepared for the "ANSWERS" program. The *d.rast.arrow* program can accept the name of a layer not drawn on the display for use as input, and an option is given to input the name of that layer, if desired. Otherwise, the currently displayed map layer will be used. Next, the color options of *d.rast.arrow* may be set via separate menus, or a choice for using default settings may be used.

5. Number

calls the *d.rast.num* program, which will print the cell category values over the displayed cells on the graphic monitor.

5. Options

change the program options setting for *grid_color*.

6. Exit

quit the *d.rast.edit* program. If edits have been made, the new raster map layer will be created. Support files are constructed. Category labels and color maps (if any) are copied from the original layer. *d.rast.edit* redisplays the original raster map file on the monitor as it exits.

NOTES

d.rast.edit will not create a new raster map layer if the user makes no cell edits while running the program.

Be careful not to cover the graphics monitor window with another frame during the editing process. In some cases, the *redraw* option will be able to remove traces of other frames; otherwise, the graphics monitor will not be refreshed until after *d.rast.edit* exits. Further note, however, that this has only been tested on *SUN* workstations, using a modified *SUNVIEW* graphics monitor driver.

The primary bane of the *d.rast.edit* program involves large map layers (with lots of rows and columns) and/or slow computers, since the program must read and write raster map layers row by row for the full size of the map layer as dictated by its region size and resolution. (The current region settings of north, south, east, and west will not limit the size of the edited copy of the map layer, since by use of the *d.rast.zoom* program, these values may change several times during the editing session). Their effects could be lessened if the program were to create a table of changes that it could incorporate into the new raster file on a forked process started when the user exits, or otherwise allow the user to issue a "save" command when he has made all desired edits to the raster file. Currently, for instance, if the user needs to use *d.rast.zoom* to

access a different area of the map, he must wait for a read and write for each entrance and egress of the edit mode.

There is no "undo" command or way to exit edit mode without saving changes.

It would be nice to incorporate a scrollable version of ***d.legend*** (such that one could see a legend for files with many categories on a standard size sub-frame). It would be even nicer to be able to select the category values from a graphical legend when editing cell values (thereby saving a trip to the text frame to type in the new value).

Perhaps method(s) for multiple or mass cell edits would be useful. This could be done by providing modes in which the user may: 1) edit a block of cells to a given value by drawing a box; 2) be able to choose a given value which is automatically used as the new value on each cell chosen until a different value is desired.

There is no interrupt handling. This could leave files in .tmp or (rarely) result in half-baked raster files. The original file would survive unscathed by an interrupt at most any point in execution, but the graphics monitor may be left in an indeterminate state (try ***d.erase*** or ***d.mon select=monitor_name*** to bring it back into shape). Beware of exiting the program by means other than using ***exit*** on the ***Main Menu***.

Perhaps a ***grid_color*** option allowing the grid to be turned off is needed, since it may be inappropriately displayed (i.e., not over the cell edges) in some circumstances.

The program has no method to enter new values beyond the current range of categories, but additional programming could make it so.

SEE ALSO

d.erase, d.frame, d.mon, d.rast, d.rast.arrow, d.rast.num, d.rast.zoom, d.zoom, g.region, r.mask, r.slope.aspect,

AUTHOR

Chris Rewerts (rewerts@ecn.purdue.edu), Agricultural Engineering, Purdue University. June 1991.

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

d.rast.num – Overlays cell category values on a raster map layer displayed to the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.rast.num
d.rast.num help
d.rast.num [**map**=*name*] [**grid_color**=*name*]

DESCRIPTION

d.rast.num overlays cell category values onto a raster map layer displayed to the user's graphics monitor. Category values will be displayed in white and/or black, based on the colors in which underlying cells are displayed. A grid outlining each map cell will also be overlain in a user-specified color.

The user should run *d.rast* to display the desired map layer on the graphics monitor before running this program.

d.rast.num can be run non-interactively or interactively. If the user specifies the map whose category values are to be displayed and/or the grid color on the command line, the program will run non-interactively. Alternately, the user may simply type **d.rast.num** on the command line; in this case, the program will prompt the user for parameter values using the standard GRASS user interface described in the manual entry for *parser*. If no grid color is given the default will be used. If no map layer is specified, the program will use whatever raster map layer is currently displayed in the active frame on the graphics monitor.

COMMAND LINE OPTIONS

Parameters:

map=*name*	Name of existing raster map layer whose category values will be displayed. Default: (whatever raster map layer is currently displayed)
grid_color=*name*	Color in which an overlain grid will be displayed. Options: white, red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, black Default: gray

NOTES

The user is advised to set the current region to a relatively small area (i.e., less than 100 rows by 100 columns); otherwise, the individual cells being displayed will be small and the category value associated with each will be difficult to see.

SEE ALSO

d.erase, d.frame, d.grid, d.mon, d.rast, d.rast.arrow, d.rast.edit, d.rast.num d.rast.zoom d.zoom, g.region, r.mask, r.slope.aspect

AUTHORS

Raghavan Srinivasan (srin@ecn.purdue.edu), and Chris Rewerts (rewerts@ecn.purdue.edu), Agricultural Engineering, Purdue University

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

d.rast.zoom – To interactively zoom in or zoom out of regions on a raster map displayed in the active frame on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.rast.zoom
d.rast.zoom help

DESCRIPTION

d.rast.zoom allows the user to zoom in or out of a region of interest on the raster map layer currently displayed in the active frame of the graphics monitor. The user zooms in and out interactively, using a mouse to establish the zoom region. To zoom, click the left mouse button to establish a corner, then drag the mouse to estabilsh the second (diagonal) corner and hit the right mouse button to accept the region. "Unzoom" will move back one step (i.e., to the last modified region). If the user tries to unzoom once again, the zoom region will be set equal to the default region settings. The user can zoom or unzoom any number of times until satisfied.

NOTES

d.rast.zoom is sensitive to the current mask (see *r.mask*.

This program runs only interactively.

SEE ALSO

d.frame, d.rast, d.rast.arrow, d.rast.edit, d.rast.num, d.zoom, g.region, r.mask, and *parser*

AUTHOR

Raghavan Srinivasan, Agricultural Engineering, Purdue University

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

d.rgb − Displays three user-specified raster map layers as red, green, and blue overlays in the active graphics frame.
(GRASS Display Program)

SYNOPSIS

d.rgb
d.rgb help
d.rgb [**red**=*name*] [**green**=*name*] [**blue**=*name*] [**out**=*name*]

DESCRIPTION

RGB stands for red, green, and blue. This program visually overlays up to three raster map layers, each displayed in either red, green, or blue. As each overlay is displayed in a single band, the intensity at each point is the average intensity of the red, green, and blue components of the currently active color table. For example, any cell that carries 100% intensity for either red, green, or blue and 0% intensity for the other two colors will be represented at 33% intensity.

WARNING: Maps that are using the "color wave" color table appear to be solid grey when using *d.rgb*. The average intensity of the colors being used is always 33%.

This program sacrifices spatial resolution to provide full color information. Any color that cannot be fully represented at a particular pixel passes the extra color to the next cell right and down. For example, if a cell is to show 50% red, but the closest color available (without showing more than 50%) is 40%, the pixle to the right and the pixle below will be given an extra 5% red each. In this way all the color is provided at the small cost of a slight amount of blurring. The result is very pleasing, especially on high resolution screens.

COMMAND LINE OPTIONS

Parameters:

red=name	Name of raster map layer to be used for RED component.
green=name	Name of raster map layer to be used for GREEN component.
blue=name	Name of raster map layer to be used for BLUE component.
out=name	Name of raster map layer to contain results.

NOTES

It produces an image and (optionally) a raster map layer with 1000 colors (10 intensities each of red, green and blue). The image and raster map layers will not display properly if the graphics device does not have at least 1000 colors while the user is running in *float* colormode. Thus, unless the display device has 1000 colors, it is necessary to run the GRASS program

d.colormode mode=fixed

before running *d.rgb*. Otherwise, the colors will be incorrect.

The intensity or the saturation layers may be left out. This means that it is possible to have *his, hi,* or *hs* images.

SEE ALSO

blend.sh, d.colormode, d.colors, d.colortable, d.his, hsv.rgb.sh, r.mapcalc, rgb.hsv.sh

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.rhumbline − Displays the rhumbline joining two user-specified points, in the active frame on the user's graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.rhumbline
d.rhumbline help
d.rhumbline coor=*lon1,lat1,lon2,lat2* [**lcolor**=*name*]

DESCRIPTION

A rhumbline is a line following a constant angle of the compass (i.e., a line of constant direction). There are 32 points on the compass (points are roughly 11 degrees 15 minutes apart). ***d.rhumbline*** displays the rhumbline joining any two user-specified points in the active frame on the user's graphics monitor. The named coordinate locations must fall within the boundaries of the user's current geographic region.

The user can run this program either interactively or non-interactively. If the user simply types **d.rhumbline** on the command line without specifying parameter values, the mouse will be activated and the user will be asked to use the mouse to indicate the two endpoints of the rhumbline. The rhumbline is then drawn in the default color (white). The program also outputs the coordinate locations of the two endpoints on the user's terminal and the number associated with the mouse button depressed by the user in a format useful for input to other programs.

Alternately, the user can specify the starting and ending longitude/latitude coordinates of the rhumbline and (optionally) the color in which the rhumbline will be displayed; in this case, the program will run non-interactively.

COMMAND LINE OPTIONS

Parameters:

coor=*lon1,lat1,lon2,lat2*
Starting and ending coordinates of the rhumblines to be displayed, given as longitudes and latitudes.

lcolor=*name*
Rhumbline color.
Options: red, orange, yellow, green, blue, magenta, indigo, violet, gray, white, black
Default: white

NOTES

This program works only with longitudelatitude data bases.

SEE ALSO

d.geodesic

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

d.save – Creates a list of commands for recreating screen graphics.
(GRASS Display Program)

SYNOPSIS

d.save
d.save help
d.save [**-ca**] [**frame**=*name*[,*name*,...]]

DESCRIPTION

When GRASS display (d.) commands are used to generate graphical images on the graphics monitor, some commands are noted in lists that the current graphics driver (see manual entry for ***d.mon***) maintains. This command, ***d.save***, uses this information to create a shell script that can be used to recreate graphics in another frame at another time. The shell script is sent to standard output (the user's terminal). If you wish to save the shell script created by ***d.save***, its output should be redirected to a file; for example:

d.save > script.out

The user can run the program non-interactively by specifying flag settings and parameter values on the command line. If the user types **d.save** without specifying these program arguments, the user will be prompted for inputs through the standard user interface described in the manual entry for ***parser***.

COMMAND LINE OPTIONS

Flags:

-c Shell script will contain only information for regenerating graphics from the active (current) frame. Use of this flag overrides the ***frame*** option (below).

-a Save information needed to regenerate all the frames on the graphics monitor. The shell script can be used to completely reconstruct the contents of the graphics screen.

Parameter:

frame=*name*[,*name*,...]
Name(s) of those display frame(s) to be saved.
Options: All available frames
Default: Active frame

NOTES

Display commands used interactively, and those that take their graphic instructions from standard input, will not be saved by ***d.save***.

SEE ALSO

d.frame, d.mon, d.savescreen, g.region, and ***parser***

AUTHOR

David Satnik, Central Washington University

NAME

d.scale − Overlays a bar scale and north arrow for the current geographic region at a user-defined location in the active display frame.
(GRASS Display Program)

SYNOPSIS

d.scale
d.scale help
d.scale [**-m**] [**bcolor**=*name*] [**tcolor**=*name*] [**at**=*x,y*]

DESCRIPTION

d.scale overlays a bar scale and north arrow for the current geographic region at a user-defined location in the active display frame. The scale and north arrow are proportioned to fit in the active frame when placed along the frame's left edge.

The user can specify the location of the scale and north arrow interactively (using a mouse), or provide the geographic coordinates at which they will be placed. The scale and north arrow will only overwrite those portions of the graphics display that lie directly beneath them.

The user can also specify the colors in which the scale and north arrow will be drawn. By default, if unspecified by the user, a white scale and north arrow will be displayed on a black background.

OPTIONS

The user can run this program either non-interactively or interactively. The program will be run non-interactively if the user specifies program parameter values and desired flag settings on the command line, using the form:

d.scale [**-m**] [**bcolor**=*name*] [**tcolor**=*name*] [**at**=*x,y*]

If at least one parameter value is given on the command line and the -m flag is not set, the program will be run non-interactively; any other parameter values left unspecified on the command line will be set equal to their default values (see below). Alternately, the user can simply type **d.scale** on the command line, without program arguments. In this case, the user will be prompted for needed parameter values and the flag setting using the standard GRASS parser interface described in the manual entry for *parser*. If the user sets the -m flag, *d.scale* will expect the user to designate the scale's location interactively, using the mouse.

Flag:

-m Use the mouse to interactively place the location of the scale and north arrow.

Parameters:

bcolor=*name*
Set the background color underlying the scale and north arrow to *name*.
Options: *red, orange, yellow, green, blue, indigo, violet*, *gray, brown, magenta, white*, and *black*
Default: *black*

tcolor=*name*
Set the foreground color in which the text, scale, and north arrow are displayed to *name*.
Options: Same as *bcolor* colors.
Default: *white*

at=*x,y*
The map easting (x) and northing (y) geographic coordinates designating the location at which the upper left corner of the scale is to be placed. The user should choose geographic coordinates located within the current geographic region (see *d.where* and *g.region* for these coordinates).
Default: 0.0,0.0

NOTES

d.scale assumes that layer units are in meters.

The scale that *d.scale* generates is probably not suitable for very tiny maps (in small display frames).

SEE ALSO

d.legend, d.measure, d.what.rast, d.what.vect, d.where, g.region, and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.sites – Displays site markers in the active display frame on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.sites
d.sites help
d.sites sitefile=*name* [**color**=*name*] [**size**=*value*] [**type**=*name*]

DESCRIPTION

d.sites displays a GRASS *site_lists* map using site markers of a color, size, and type specified by the user. Output is displayed in the active frame on the graphics monitor.

This program can be run non-interactively or interactively. If the user gives the name of a GRASS *site_lists* map and (optionally) specifies other parameter values on the command line the program will be run non-interactively; any parameter values left unspecified are set to their default values (see below). Alternately, the user can type simply **d.sites** on the command line; in this case, the program will prompt the user for parameter values using the standard GRASS parser interface described in the manual entry for *parser*.

Parameters:

sitefile=*name* — Name of a GRASS *site_lists* map in the user's current mapset search path.

color=*name* — Sets the current color to *name*.
Options: red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, white, and black
Default: *gray*

size=*value* — Size, in pixels, in which the site icons are to be drawn.
Options: 0-1000
Default: 5

type=*name* — The type of icon to be displayed at site locations.
Options: x, diamond, box, +
Default: x

SEE ALSO

d.display, d.icons, d.points, d.rast, d.sites, d.vect, g.region and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.text – Draws text in the active display frame on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.text
d.text help
d.text [**size**=*value*] [**color**=*name*] [**line**=*value*]

DESCRIPTION

d.text draws text in the active display frame on the graphics monitor. Text can be provided through standard input or redirected from a file (using the UNIX redirection mechanism).

Parameters:

size=*value* Height of letters, stated as a percent of the available display frame height.
Options: 0 - 100
Default: 5

color=*name* Sets display text color to *name*.
Options: red, orange, yellow, green, blue, indigo, violet, gray, white, and black
Default: *gray*

line=*value* The screen line number on which the first line of text will be drawn.
(Line 1 is at the top of the active display frame.)
Options: 1 - 1000
Default: 1

In addition to the options provided on the command line, colors, text size, font type, and boldness, can be adjusted with commands in the standard input (i.e., if the user invokes *d.text* without options on the command line, and then assigns values to these options on lines within the standard input). In this case, the user also sees the standard GRASS parser interface described in the manual entry for *parser*.

Commands:

.C *color*
(where *color* is one of the available colors) causes text appearing on subsequent lines to be drawn in that color.

Text size can be adjusted with the command:
.S *size*
(where *size* is a percentage within the range 0 to 100). Note that a size of 10 would allow 10 lines to be drawn in the active display frame, 5 would allow the drawing of 20 lines, and 50 would allow the drawing of 2 lines.

Font type can be manipulated using the command:
.F *font*
(where *font* is one of the fonts known by the GRASS program *d.font*). Available fonts are listed in the GRASS manual entry for *d.font*. The default font type used (if unspecified by the user) is *romans*. Run the GRASS macro *show.fonts.sh* to see what these fonts look like.

The user can also stipulate that text be printed in **bold** on lines beneath the command:
.B 1
This command means *bold on* .
Similarly, the command:
.B 0
turns *bold off* of all text appearing on lines beneath it. (*Bold off* is used by default, if unspecified by the user.)

EXAMPLE

The following command will print the short phrase "This is a test of d.text" in the active display frame using the color yellow, in bold, and using 4/100'ths (4%) of the active frame's vertical space per line:

```
d.text
.C yellow
.S 4
.B 1
This is a test of d.text
```

The user presses *control-d* (the "ctrl" and "d" keys) to end input to *d.text*.

NOTES

Note that the GRASS command *d.title* creates map titles in a format suitable for input to *d.text*.

d.text needs escape sequences that can be used within lines to change colors, boldness, and perhaps size.

SEE ALSO

d.font, d.title, show.fonts.sh, and *parser*

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.title – Outputs a title for a raster map layer in a form suitable for display by *d.text*.
(GRASS Display Program)

SYNOPSIS

d.title
d.title help
d.title [**-f**] **map**=*name* [**color**=*name*] [**size**=*value*]

DESCRIPTION

d.title generates to standard output a string which can be used by *d.text* to draw a title for the raster map layer *name* in the active display frame on the graphics monitor. Output created by *d.title* can be redirected into a file, or piped directly into *d.text* to display the map title created by *d.title*. The map title created will include the map layer's name, title, MAPSET, LOCATION_NAME, geographic region boundary coordinates, and cell resolution.

The user can state program arguments on the command line, or type simply **d.title** on the command line. In the latter case, the program will prompt the user for the parameter values and flag setting using the standard GRASS parser interface described in the manual entry for *parser*.

Flag:

-f Displays a fancier title.

Parameters:

map=*name* Name of an existing raster map layer in the user's mapset search path.

color=*name* Sets the current color to the *name* stated.
Options: red, orange, green, blue, indigo, violet, black, white, gray, yellow, brown, and magenta
Default: *white*

size=*value* Sets the text size as a percentage of the active display frame's height, to *value*. Floating point values can be used.
Options: 0 - 100
Default: 15.0

EXAMPLE

For example, a user wishing to create a suitable title for the Spearfish, SD *soils* map layer and to display this title in the active display frame on the graphics monitor might type the following:

d.title map=soils color=red size=5 > title.file

d.text < title.file

Alternately, the user might pipe *d.title* output directly into *d.text*:

d.title map=soils color=red size=5 | d.text

A file created by *d.title* can be displayed with *d.text*. Information contained in this file takes precedence over the *color* and *size* parameters for *d.text*.

NOTES

The text created with *d.text* will not necessarily fit within the active display frame on the graphics monitor; the user should choose a text size appropriate to this frame.

SEE ALSO

d.font, d.text and *parser*

AUTHORS

James Westervelt, U.S. Army Construction Engineering Research Laboratory

NAME

d.vect – Displays GRASS vector data in the active frame on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.vect
d.vect help
d.vect map=*name* [**color**=*name*]

DESCRIPTION

d.vect displays the user-named binary vector file in the active display frame on the graphics monitor, in the user-named color. The vector file must already exist under a mapset listed in the user's current mapset search path.

This program can be run either non-interactively or interactively. If the user gives the name of a binary vector map to be displayed and (optionally) specifies a color for vector display, the program will run non-interactively. (If the user gives a vector map name but no vector color on the command line, the named vector map layer will be displayed in the default color *white*.) Alternately, the user can simply type **d.vect** on the command line; in this case, the program will prompt the user for parameter values using the standard GRASS parser interface described in the manual entry for *parser*.

Parameters:

map=*name* — Vector map layer to be displayed in the active display frame.
Must be binary vector file in the user's current mapset search path.

color=*name* — Sets vector display color to *name*.
Options: white, red, orange, yellow, green, blue, indigo, violet, magenta, brown, gray, and black
Default: *white*

NOTES

This program needs to be upgraded with features that would:

1) Allow different line colors, weights, and textures to be assigned to vectors with different category values.
2) Allow different fill patterns and colors to be assigned to areas.
3) Be linked with an associated program identifying these options.

SEE ALSO

d.rast, d.sites, d.vect.dlg, d.what.vect, g.mapsets, and *parser*

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

d.what.rast − Allows the user to interactively query the category contents of multiple raster map layers at user-specified locations within the current geographic region.
(GRASS Display Program)

SYNOPSIS

d.what.rast
d.what.rast help
d.what.rast [**-1t**] [**map**=*name*[,*name*,...]] [**fs**=*name*]

DESCRIPTION

d.what.rast outputs the category values and labels associated with cell(s) at user-specified location(s) on user-named raster map layer(s).

The program will query the contents of raster map layer(s) named by the user on the command line. These map layers must exist in the user's current mapset search path. If the user does not name any raster map layers on the command line, *d.what.rast* will query the category contents of whatever raster map layer is already displayed in the active frame on the graphics monitor.

The program activates the mouse, and expects the user to indicate the cell location(s) to be queried by depressing a mouse button over desired location(s) within the current geographic region in the active display frame on the graphic monitor.

Flags:

-1 Identify and query just one point location. Only one mouse click is executed. This option is provided for shell scripts and programs that want to obtain only one point from the user.

-t Provide only terse output. This option is provided to simplify the parsing of output by other programs.

Parameters:

map=*name* Name of existing raster map layer(s). Limit: 15 maps
Default: Query map currently displayed in the active graphics frame.

fs=*name* Output field separator to be used (in terse mode only).
Default: **:**

d.what.rast output consists of the geographic coordinates of the location pointed to, and, for each map layer, the map layer name, the category value, and category label in the named raster map layers at this location.

EXAMPLE

It is helpful, but not necessary, to first display a map to be used for reference in the active display frame before running *d.what.rast*. For example, the user might type the following series of commands and receive the output below.

d.rast map=soils

To first display the *soils* map in the active frame.

d.what.rast map=soils,aspect

User then moves the mouse to desired location on the displayed *soils* map layer, and presses the left mouse button to query the category contents of the *soils* and *aspect* maps at this geographic location. The program then outputs the below information to the user's terminal.

617112(E) 3732014(N)
soils in PERMANENT (44)Nunn clay loam, NdC
aspect in PERMANENT (20)15 degrees north of west

The first line of output gives the easting (E) and northing (N) coordinates of the geographic location at which the user clicked the mouse. The subsequent two lines give the map name and mapset, map category value (within parentheses), and map category label corresponding to this user-selected map location, for each of the maps queried by the user.

The query may be repeated as often as desired using the left mouse button. The right button on the mouse is used to quit the *d.what.rast* session.

Users can set the -t flag to obtain a terse output from *d.what.rast*. This is useful when the user wishes output to be parsed by another program (like *awk*). If the -t flag is set, users can also select the field separator used (with the fs=*name* option), or elect to use the default **:** field separator. In this case, the command

d.what.rast -t map=soils,aspect

produces output in the form shown below. The first line of output gives the easting, northing, and the mouse button return value (see NOTES, below). Each subsequent line gives the map name and its mapset, the category value, and category label of the point specified on the user-named raster map layers. The default output field separator, a colon, is used since none was specified on the command line.

617112:3732014:1
soils@PERMANENT:44:Nunn clay loam, NdC
aspect@PERMANENT:20:15 degrees north of west

Users can also use this program inside of shell scripts that require as input a map category value and a mouse button depressed. Users can choose an option to run *d.what.rast* only once, and return only the map category value found and the number of the mouse button depressed.

NOTES

Mouse button return values are as follows: 0 indicates no button was pressed, 1 indicates that the left mouse button was pressed, 2 indicates the middle button was pressed, and 3 indicates that the right mouse button was pressed.

d.what.rast will always print its output to the terminal screen. *d.what.rast* output can be redirected into a file; however, if it is, the output will go both to the screen and to the file. For example:

d.what.rast map=soils,aspect > what.out

will both send *d.what.rast* output to the screen and capture its output in the file named *what.out*.

The maximum number of raster map layers that can be queried at one time is 15.

d.what.vect allows the user to interactively query the category contents of multiple vector map layers at user-specified locations.

SEE ALSO

d.rast, d.what.vect, d.where, g.region, and *parser*

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

d.what.vect – Allows the user to interactively query the category contents of a (binary) vector map layer at user-selected locations within the current geographic region.
(GRASS Display Program)

SYNOPSIS

d.what.vect
d.what.vect help
d.what.vect [**-1**] **map**=*name*

DESCRIPTION

d.what.vect outputs the category value(s) associated with user-specified location(s) in a user-specified vector map layer. This program currently returns only category values for line types in the user-specified vector file.

The program will query the contents of the vector map layer named by the user on the command line. This map layer must exist in the user's current mapset search path.

The program activates the mouse, and expects the user to indicate the location(s) to be queried by depressing a mouse button over desired location(s) within the current geographic region in the active display frame on the graphic monitor.

Flag:

-1 Identify and query just one point location. Only one mouse click is executed. This option is provided for shell scripts and programs that want to obtain only one point from the user.

Parameter:

map=*name* Name of an existing binary vector map in the user's mapset search path.

EXAMPLE

A sample *d.what.vect* session is given below. Although it is not necessary that the user first display a vector map to be queried in the active display frame, it is helpful to have a map displayed there for reference.

d.vect map=roads.24000
Displays the 1:24,000 scale *roads* vector map layer on the graphics monitor.

d.what.vect map=roads.24000
After typing this, the user moves the mouse to a desired location on the displayed *roads* map layer, and presses the left mouse button to query the category value of the *roads* vector map at this location. The program then outputs the category value of a line type corresponding to this user-selected map location, for the vector map queried by the user.

The query may be repeated as often as desired using the left mouse button. The right button on the mouse is used to quit the *d.what.vect* session.

Users can also use this program inside of shell scripts that require as input a map category value and a mouse button depressed. Users can set the -1 flag to run *d.what.vect* only once, and return only the map category value found and the number of the mouse button depressed. (Mouse button return values are as follows: 0 indicates no button was pressed, 1 indicates that the left mouse button was pressed, 2 indicates the middle button was pressed, and 3 indicates that the right mouse button was pressed.)

NOTES

Currently, ***d.what.vect*** only outputs category values for lines. It does not output category labels for lines, nor output category values or category labels for areas in a vector file.

d.what.vect will always print its output to the user's terminal screen. ***d.what.vect*** output can be redirected into a file; however, if it is, the output will go both to the screen and to the file. For example:

d.what.vect map=roads > what.out

will both send ***d.what.vect*** output to the screen and capture its output in the file named ***what.out***.

d.what.rast can be used to interactively query the map category contents of multiple raster map layers at user-selected locations.

SEE ALSO

d.rast, d.vect, d.what.rast, g.region, and ***parser***

AUTHOR

Jim Hinthorn, Central Washington University
Upgrades: Dennis Finch, National Park Service

NAME

d.where − Identifies the geographic coordinates associated with point locations in the active frame on the graphics monitor.
(GRASS Display Program)

SYNOPSIS

d.where
d.where help
d.where [**-1**] [**spheroid**=*name*]

DESCRIPTION

d.where is an *interactive* program that allows the user, using the pointing device (mouse), to identify the geographic coordinates associated with point locations within the current geographic region in the active display frame on the graphics monitor.

If the user runs *d.where* without specifying the name of a spheroid on the command line, each mouse click will output the UTM easting and northing of the point currently located beneath the mouse pointer. A mouse-button menu is presented so the user knows which mouse buttons to use. The output is always printed to the terminal screen; if the output is redirected into a file, it will be written to the file as well.

Flag:

-1 Only one mouse click is executed.
This option is provided for shell scripts and programs which want to obtain only one point from the user. The output is only written to stdout, unless redirected into a file. The geographic location and mouse button pressed are output.

Parameter:

spheroid=*name* Name of a spheroid (for latitude/longitude coordinate conversion).
Options: australian, bessel, clark66, clark80, everest, international, wgs72, wgs84

NOTES

This program uses the current geographic region setting and active frame. It is not necessary, although useful, to have displayed a map in the current frame before running *d.where*.

SEE ALSO

d.what.rast, d.what.vect, g.region

AUTHORS

James Westervelt, U.S. Army Construction Engineering Research Laboratory
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

d.zoom – Allows the user to change the current geographic region settings interactively, with a mouse. *(GRASS Display Program)*

SYNOPSIS

d.zoom
d.zoom help
d.zoom [**-q**] [**action**=*name*]

DESCRIPTION

d.zoom allows the user to interactively adjust the settings of the current geographic region using a pointing device such as a mouse. Like *g.region*, *d.zoom* re-defines the settings of the geographic region. However, *d.zoom* allows the user to change the current region settings interactively, by either outlining the new region perimeter with a mouse or "rotating" latitudelongitude data into the current region. The user should run *d.erase* after *d.zoom* is run for the new region settings to affect the graphics display.

Flag:

-q Run quietly, suppressing output of some program messages to standard output.

Parameter:

action=*name* Allows the user to change the geographic region settings by "rotating" the globe to move a different portion of the globe into the stationery boxed region outline (making this new portion of the globe the new current region), rather than by moving the boxed region outline. This option can only be used with latitudelongitude data bases (although *d.zoom* will not complain if the user attempts to set this parameter while running on a non-latitudelongitude data base).
Options: zoom, rotate

After the user types the command **d.zoom** and (optionally) sets the -q flag and type of zoom to be performed, a mouse-button menu will appear, directing the user to: establish the corners of the new geographic region, check its coordinates, and confirm any changes made. When the user accepts new geographic boundaries, *d.zoom* asks,

Accept new region? Y / N >

If the user clicks the mouse over the "Y" (yes), the mouse-drawn geographic region is saved as the user's current geographic region. The user is warned that *d.erase* should be run after *d.zoom* to make new current region settings affect the graphics display.

If the user clicks the mouse over the "N" (no), the first mouse-drawn geographic region is not saved; instead, *d.zoom* asks whether or not the user wishes to

Try Again? Y / N >

The user can then opt to draw a new region with the mouse ("Y"), or opt to exit *d.zoom* ("N") and leave current region settings unchanged.

To zoom-out to a larger area than was windowed-in on with *d.zoom* (i.e., to enlarge the current geographic region), the user may run *g.region*. The user may also alter the current geographic region by running the "region" option of the *d.display* program.

NOTES

Although it is not necessary that the user display a map in the active display frame before running *d.zoom*, it is helpful to do this for reference.

Currently, the rubber-banded boundaries of the zoom region drawn by the user are not clearly visible on the graphics monitor. However, whether or not these boundaries are clearly visible when drawn, the drawn region is still correctly zoomed-in on.

SEE ALSO

d.display, d.erase, d.rast, g.region

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

exit – Exits the user from the current GRASS session.

SYNOPSIS

exit

DESCRIPTION

The *exit* command ends the user's current GRASS session and returns the user to the directory in which he was working prior to entering GRASS. When the user exits GRASS, he is asked whether he wishes to save the files and data stored under his current mapset, and (if maps are present) whether the user wishes to selectively remove map layers, before exiting the system. By default, if the user presses RETURN without responding to these questions, all maps in the user's current mapset are saved. However, because such maps can consume much storage space on the computer, the user should remove any unneeded files before exiting. (The user can also remove data before attempting to *exit* GRASS using the *g.remove* command.)

Before typing *exit*, the polite user should remember to release the graphics display monitor (using the command **d.mon -r**) for use by other GRASS users. Otherwise, the display device may be locked for use by the user, even though the user has exited GRASS, until another user runs *d.mon* and unlocks this monitor for others' use.

Each time the user re-enters GRASS, the variables described in *g.gisenv* are re-set. If the user wishes to change the mapset, location, or data base on which he is working (i.e., those affected by any GRASS programs running during the user's current GRASS session), the user must *exit* GRASS and then re-enter GRASS and specify a different mapset, location, and/or data base location on the GRASS start-up page. When the user re-enters GRASS, these variable settings (the current mapset, location, and data base) are set by default to those used in the user's previous GRASS session, unless changed by the user.

NOTES

This program requires no command line arguments; the user simply types **exit** on the command line to exit GRASS.

SEE ALSO

d.mon, g.gisenv, g.remove

NAME

g.access – Controls user access to the current GRASS mapset.
(GRASS File Management Program)

SYNOPSIS

g.access

DESCRIPTION

This program allows the user to control access to the current mapset. Normally, any user can read data from any GRASS mapset. But sometimes it is desirable to prohibit access to certain sensitive data. The *g.access* command allows a user to restrict read and execute access to the current mapset (see UNIX *chmod* command). *g.access* will not modify write access to the current mapset.

The user may, for example, allow only users in the same UNIX group to read data files in the mapset, or restrict the mapset to personal use only.

After typing

g.access

the user will be presented with a screen page which reflects the current mapset permissions. The user can then change them. The screen page looks like:

```
LOCATION: spearfish                                    MAPSET: demo

This program allows you to control access to your mapset by other users.
Access may be granted/removed for everyone, or for everyone in your group.

Mark an 'x' to allow access;  erase the field to restrict access.

  GROUP: _x_
  OTHER: _x_

            AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
                        (OR <Ctrl-C> TO CANCEL)
```

If you remove the x (using the space bar), access will be denied to that class of user (group or other). If you type an x, access will be granted to that class of user.

NOTES

There is no non-interactive version of *g.access*.

Under GRASS version 4.0, access to the mapset PERMANENT must be open to all users. This is because GRASS looks for the user's default geographic region definition settings and the location title in files that are stored under the PERMANENT mapset directory. The *g.access* command, therefore, will not allow you to restrict access to the PERMANENT mapset.

The *g.mapsets* command isn't smart enough to tell if access to a specified mapset is restricted, and the user is therefore allowed to include the names of restricted mapsets in his search path. However, the data in a restricted mapset is still protected; any attempts to look for or use data in a restricted mapset will fail. The user will simply not see any data listed for a restricted mapset.

SEE ALSO

UNIX manual entries for *chmod* and *group*
g.mapsets

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

g.ask − Prompts the user for the names of GRASS data base files.
(GRASS File Management Program)

SYNOPSIS

g.ask help
g.ask type=*name* [**prompt**=*"string"*] **element**=*name* [**desc**=*"string"*] **unixfile**=*name*

DESCRIPTION

g.ask is designed for shell scripts that need to prompt the user for the name of a data base file in the user's current GRASS location. After *g.ask* is invoked with needed parameters, it will query the user for a file name of the specified *type* and *element*. After the user responds to this query, the program will write four lines to the UNIX output file specified by *unixfile*.

Parameters:

type=*name* The type of query. Options for *name* are *old*, *new*, *any*, and *mapset*; their functions are given below. "New", "any", and "mapset" only look in the user's current mapset.

- **old** For existing data files anywhere in the user's mapset search path.
- **new** Used to create a new file in the current mapset, which must not already exist there (if a file with this name exists there, it will not be overwritten).
- **any** Creates a file in the current mapset, which may or may not already exist there. If a file with this name exists in the current mapset, it will be overwritten; if not, a new file with this name will be created.
- **mapset** For files that must exist in the current mapset The shell write wants the name of a file which exists in the user's current mapset. This type would used instead of "old" if the file is to be modified.

prompt="*string*" The prompt to be displayed to the user. If more than one word, the prompt should be enclosed within double quotes ("").

element=*name* The name of the GRASS data base element (i.e., directory under a GRASS mapset) whose files are to be queried.

desc="*string*" A short description of the data base element which is meaningful to the user. If description contains more than one word, it should be enclosed within double quotes ("").

unixfile=*name* The name of a UNIX file to store the user's response. See next section for what is written to this file and how it can be used by shell scripts.

OUTPUT

Upon receiving the user's response to its request for a file name, *g.ask* writes four lines to the specified *unixfile*; this output file is placed in the user's current working directory, unless otherwise specified, and contains the following lines:

name='*some_name*'
mapset='*some_mapset*'
fullname='*some_fullname*'
file='*some_fullpath*'

The output is /bin/sh commands to set the variable *name* to the file name specified by the user (of the *element* and *type* requested by *g.ask*), *mapset* to the GRASS mapset in which this file resides (or will be created), *fullname* is the name with the mapset embedded in it, and *file* to the full UNIX path name identifying this file. These variables may be set in the /bin/sh as follows:

. unixfile

The . is a shell command which means read the *unixfile* and execute the commands found there. It is NOT

part of the *unixfile* name and MUST be followed by a space.

NOTES

The user may choose to simply hit the return key and not enter a file name. If this happens the variables will be set as follows:

```
name=
mapset=
fullname=
file=
```

The following is a way to test for this case:

```
if [ ! "$file" ]
then
        exit
fi
```

SEE ALSO

d.ask, g.filename, g.findfile, g.gisenv

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

g.copy – Copies available data files in the user's current mapset search path and location to the appropriate element directories under the user's current mapset.
(GRASS File Management Program)

SYNOPSIS

g.copy
g.copy help
g.copy [**rast**=*from,to*] [**vect**=*from,to*] [**icon**=*from,to*] [**labels**=*from,to*] [**sites**=*from,to*]
[region=*from,to*] [**group**=*from,to*]

DESCRIPTION

A user may access data stored under the other mapsets listed in his mapset search path. However, the user may only modify data stored under his own current mapset. *g.copy* allows the user to copy existing data files *from* other mapsets *to* the user's current mapset ($MAPSET). The files to be copied must exist in the user's current mapset search path and location; output is sent to the relevant data element directory(ies) under the user's current mapset.

The user specifies the type(s) of data files he wishes to copy (raster, vector, etc.), the name of the existing file to be copied (i.e., the *from* file name), and the name of the new file copy to be placed in the user's current mapset (the *to* file name). This information can be given either (non-interactively) on the command line, or entered in response to program prompts given via the standard parser interface described in the manual entry for *parser*.

Information can be entered on the command line in the following format:

> **g.copy** [**rast**=*from,to*] [**vect**=*from,to*] [**icon**=*from,to*] [**labels**=*from,to*] [**sites**=*from,to*]
> **[region**=*from,to*] [**group**=*from,to*]

For example, if the user wished to copy the existing raster file *soils* to a file called *soils.ph* and to copy an existing vector file *roads* to a file called *rds.old*, the user could type:

> **g.copy rast=soils,soils.ph vect=roads,rds.old**

Data files can also be specified by their mapsets. For example, the below command copies the raster file named *soils* from the mapset *wilson* to a new file called *soils* to be placed under the user's current mapset:

> **g.copy rast='soils@wilson',soils**

If no mapset name is specified, *g.copy* searches for the named *from* map in each of the mapset directories listed in the user's current mapset search path in the order in which mapsets are listed there (see *g.mapsets*).

If the user does not enter parameter values but instead types only **g.copy** on the command line the program will prompt the user for a data type, the name of a file of this data type to copy, and the name of a new file to hold the copy. After both file names have been entered, the copy is created and the user is again prompted for a data element to be copied, until the user hits RETURN. When prompted for file names, the user may enter 'list' to see a list of existing files, or hit RETURN to end the file listing.

Parameters:

rast=*from,to* where *from* is an existing raster map layer to be copied, and *to* is the name given to the copy.

vect=*from,to* where *from* is an existing binary vector map layer to be copied, and *to* is the name given to the copy.

icon=*from,to* where *from* is an existing paint icon file to be copied, and *to* is the name given to the copy.

labels=*from,to* where *from* is an existing /paint/labels file to be copied, and *to* is the name given to the copy.

sites=*from,to* where *from* is an existing *site_lists* file to be copied, and *to* is the name given to the copy.

region=*from,to* where *from* is an existing region definition (*windows*) file to be copied, and *to* is the name given to the copy.

group=*from,to* where *from* is an existing imagery group file to be copied, and *to* is the name given to the copy.

NOTE

If a file has support files (e.g., as do raster data files), these support files will also be copied.

SEE ALSO

g.access, g.list, g.mapsets, g.remove, g.rename, and *parser*

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

g.filename − Prints GRASS data base file names.
(GRASS File Management Program)

SYNOPSIS

g.filename
g.filename help
g.filename element=*name* **mapset**=*name* **file**=*name*

DESCRIPTION

g.filename is designed for Bourne shell scripts that need to know the full UNIX file name for raster map layers, vector files, site list files, geographic region definition (windows) files, imagery group files, etc., in the GRASS data base. If the user runs *g.filename* without command line arguments (i.e., simply types **g.filename**), this program will prompt the user for input using the standard parser interface described in the manual entry for *parser*.

Parameters:

element=*name* The name of a GRASS data base element (i.e., directory within the GRASS mapset location).

mapset=*name* The name of a GRASS data base mapset. As a convenience, a single dot (.) can be used to designate the current mapset.

file=*name* The name of a GRASS data base file.

OUTPUT

g.filename writes one line to standard output:

file='*full_file_pathname*'

The output is a /bin/sh command to set the variable specified by the file *name* to the full UNIX path name for the data base file. This variable may be set in the /bin/sh as follows:

**eval `g.filename element**=*name* **mapset**=*name* **file**=*name*`

NOTES

This routine generates the filename, but does not care if the file (or mapset or element) exists or not. This feature allows shell scripts to create new data base files as well as use existing ones.

If the mapset is the current mapset, *g.filename* automatically creates the *element* specified if it doesn't already exist. This makes it easy to add new files to the data base without having to worry about the existence of the required data base directories. (This program will not create a new mapset, however, if that specified does not currently exist.)

The program exits with a 0 if everything is ok; it exits with a non-zero value if there is an error, in which case file='*full_file_pathname*' is not output.

SEE ALSO

g.ask, g.findfile, g.gisenv, and *parser*

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

g.findfile − Searches for GRASS data base files and sets variables for the shell.
(GRASS File Management Program)

SYNOPSIS

g.findfile
g.findfile help
g.findfile element=*name* [**mapset**=*name*] **file**=*name*

DESCRIPTION

g.findfile is designed for Bourne shell scripts that need to search for raster map layer files, vector files, site list files, geographic region definition (windows) files, and imagery group files in the GRASS data base. If the user runs *g.findfile* without command line arguments, he will be prompted for the names of a GRASS element, file, and mapset, through the standard parser interface (see manual entry for *parser*).

Parameters:

element=*name* The data base element (i.e., directory within a GRASS mapset) to be searched.

mapset=*name* The mapset in which to search for the specified file *name*. If not specified, all mapsets in the user's GRASS search path are searched. Otherwise, the specified mapset is searched. As a convenience, if specified as a single dot (.) only the current mapset is searched.

file=*name* The name of a GRASS data file (of the stated *element* type) for which to search.

OUTPUT

g.findfile writes four lines to standard output:

```
name='file_name'
mapset='mapset_name'
file='unix_filename'
fullname='grass_fullname'
```

The output is /bin/sh commands to set the variable *name* to the GRASS data base file name, *mapset* to the mapset in which the file resides, and *file* to the full UNIX path name for the named file. These variables may be set in the /bin/sh as follows:

**eval `g.findfile element**=*name* **mapset**=*name* **name**=*name*`

NOTES

If the specified file does not exist, the variables will be set as follows:

```
name=
mapset=
fullname=
file=
```

The following is a way to test for this case:

```
if [ ! "$file" ]
then
        exit
fi
```

SEE ALSO

g.ask, g.filename, g.gisenv, g.mapsets, and *parser*

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

g.gisenv – Outputs the user's current GRASS variable settings.
(GRASS File Management Program)

SYNOPSIS

g.gisenv
g.gisenv [*variable_name*]

DESCRIPTION

When a user runs GRASS, certain variables are set specifying the GRASS data base, location, mapset, peripheral device drivers, etc., being used in the current GRASS session. These variable name settings are recognized as long as the user is running a GRASS session.

No prompts are given to the user when running *g.gisenv*. If run without arguments, *g.gisenv* lists all of the user's current GRASS variable settings. Results are sent to standard output, and may look like this:

```
GISDBASE=/usr/grass4/data
LOCATION_NAME=spearfish
MAPSET=PERMANENT
```

In this example, the full path name of the user's current location (i.e., $LOCATION_NAME) is /usr/grass4/data/spearfish, and the full path name of the user's current mapset (i.e., $MAPSET) is /usr/grass4/data/spearfish/PERMANENT.

If the user specifies a *variable_name* on the command line (e.g., **g.gisenv MAPSET**), only the value for that particular GRASS variable is output to standard output. Possible variable names depend on the user's system.

While other variables may be associated with each GRASS session (e.g., DIGITIZER, PAINTER, DISPLAY, and other variables), those stated below are essential.

GISDBASE - The $GISDBASE is a directory in which all users' GRASS data are stored. Within the $GISDBASE, data are segregated into subdirectories (called "locations") based on the map coordinate system used and the geographic extent of the data. Each "location" directory itself contains subdirectories called "mapsets"; each "mapset" stores "data base elements" -- the directories (e.g., the *cell*, *cellhd*, *dig*, etc., directories) in which GRASS data files are actually stored.

LOCATION_NAME - The user must choose to work with the data under a single GRASS location within any given GRASS session; this location is then called the *current GRASS location*, and is specified by the variable $LOCATION_NAME. The $LOCATION_NAME is the GRASS data base location whose data will be affected by any GRASS commands issued during the user's current GRASS session, and is a subdirectory of the current $GISDBASE. Each "location" directory can contain multiple "mapset" directories (including the special mapset "PERMANENT"). Maps stored under the same GRASS LOCATION_NAME (and/or within the same MAPSET) must use the same coordinate system and typically fall within the boundaries of the same geographic region (aka, "location").

MAPSET - Each "mapset" contains a set of maps relevant to the LOCATION_NAME directory in which it appears. Each LOCATION_NAME can contain multiple mapsets. (Mapsets which fall under the same LOCATION_NAME all contain data geographically relevant to the LOCATION_NAME, and all store data in the same map coordinate system. Frequently, maps are placed into different mapsets to distinguish file ownership -- e.g., each user might have his own mapset, storing any maps that he has created and/or are relevant to his work.) During each GRASS session, the user must choose one mapset to be the *current mapset*; the current mapset setting is given by

$MAPSET, and is a subdirectory of $LOCATION_NAME. During a single GRASS session, the user can use available data in any of the mapsets stored under the current LOCATION_NAME directory that are in the user's mapset search path and accessible by the user. However, within a single GRASS session, the user only has *write* access to data stored under the *current mapset* (specified by the variable $MAPSET).

Each "mapset" stores GRASS data base elements (i.e., the directories in which GRASS data files are stored). Any maps created or modified by the user in the current GRASS session will be stored here. The MAPSET directory "PERMANENT" is generally reserved for the set of maps that form the base set for all users working under each LOCATION_NAME.

Once within a GRASS session, GRASS users have access only to the data under a single GRASS data base directory (the *current GRASS data base*, specified by the variable $GISDBASE), and to a single GRASS location directory (the *current location*, specified by the variable $LOCATION_NAME). Within a single session, the user may only *modify* the data in the *current mapset* (specified by the variable $MAPSET), but may *use* data available under other mapsets under the same LOCATION_NAME.

All of these names must be legal names on the user's current system. For UNIX users, names less than 14 characters and containing no non-printing or space codes are permissible. Examples of permissible names include: *one, mymap*, *VeGe_map*, and *1_for_me*. The underscore character can safely be used in place of a blank for multiple-word names.

NOTES

The output from *g.gisenv* when invoked without arguments is directly usable by /bin/sh. The following command will cast each variable into the UNIX environment:

eval `g.gisenv`

This works only for /bin/sh. The format of the output is not compatible with other UNIX shells.

SEE ALSO

g.access, g.ask, g.filename, g.findfile, g.mapsets

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

g.help – GRASS help facility.
(GRASS Help Program)

SYNOPSIS

g.help

DESCRIPTION

g.help provides the user with functional information on GRASS programs, a glossary, and access to on-line *User's Reference Manual* entries. The help facility is accessed by simply typing **g.help** on the command line. The user can then wend his way through a series of menus (organized by functional area) and keyword searches.

On-line reference manual entries can also be accessed directly, through the GRASS *g.manual* command.

SEE ALSO

GRASS User's Reference Manual

g.manual

AUTHOR

James Westervelt, U.S. Army Construction Engineering Research Laboratory
Deb Brinegar, U.S. Army Construction Engineering Research Laboratory
Mary Martin, U.S. Army Construction Engineering Research Laboratory

Note. The help facility uses the *hyper* program written by Jim Westervelt.

NAME

g.list − Lists available GRASS data base files of the user-specified data type to standard output.
(GRASS File Management Program)

SYNOPSIS

g.list
g.list help
g.list [**-f**] **type**=*datatype* [**mapset**=*name*]

DESCRIPTION

g.list allows the user to list user-specified, available and accessible files from *mapsets* under the user's current location. When invoked simply as **g.list**, the program prompts the user for the type of data to be listed from all *mapsets* in the user's current mapset search path. The user can list files from a mapset not listed in the current mapset search path by running the program non-interactively, specifying the (optional) flag setting and parameter values on the command line. Program flag and parameters are described below.

Flag:

-f Returns a verbose file listing, that includes map titles.

Parameters:

type=*datatype* The type of data to be listed.
Options: (listed in bold)

rast	Raster files
vect	Binary vector files
icon	Paint icon files
labels	Paint labels files
sites	Site list files
region	Region definition files
group	Imagery group files

mapset=*name* The name of a mapset to be searched for files of the specified *type*. Any mapset name under the current location, whether or not it is listed in the user's current mapset search path, can be specified.
Default: If unspecified, files of the specified *type* from all mapsets in the user's current search path will be listed to standard output.

NOTES

If the user requests that files from a mapset to which access has been restricted (see *g.access* be listed, no files from this mapset will be listed.

SEE ALSO

g.access, g.mapsets

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

g.manual – Accesses GRASS User's Reference Manual entries.
(GRASS File Management Program)

SYNOPSIS

g.manual
g.manual help
g.manual [**-1aefs**] [**entries**=*name*[,*name*,...]]

DESCRIPTION

The *g.manual* command provides user access to the on-line **GRASS User's Reference Manual** entries. The user may request a list of available manual entries, and may get a manual entry printed to the terminal screen and/or to the line printer.

g.manual can be run either interactively or non-interactively. If the user types

g.manual

on the command line without program arguments, the program will prompt the user for a manual entry to display. The user may enter "list" to get a section-by-section listing of the manual entries available. Once the user has viewed the desired information it may be printed by responding to the questions appropriately.

The user can run the program non-interactively, by specifying the appropriate options and/or the *name*(s) of the manual entries to displayed.

OPTIONS

-1 This option will list all manual entries, one per line.

-a This option will list all manual entries in a more appealing format. The manual page list will be separated by manual section.

-e This option tells *g.manual* to ignore empty manual sections when printing the listings from the *-1* or *-a* options.

-f This option will add formfeeds to output listing when using the *-a* option.

-s This option will cause *g.manual* to run silently. Instead of displaying the manual page it will simply set the exit status to:
0 if entry exists, or
1 if it does not exist.

These entries may also be accessed through the *g.help* command.

SEE ALSO

GRASS User's Reference Manual

g.help

AUTHOR

Kurt Buehler, U.S. Army Construction Engineering Research Laboratory

NAME

g.mapsets – Modifies the user's current mapset search path, affecting the user's access to data existing under the other GRASS mapsets in the current location.
(GRASS File Management Program)

SYNOPSIS

g.mapsets
g.mapsets help
g.mapsets [**-lp**] [**mapset**=*name*[,*name*,...]]

DESCRIPTION

A *mapset* holds a distinct set of data layers, each relevant to the same (or a subset of the same) geographic region, and each drawn in the same map coordinate system. At the outset of every GRASS session, the user identifies a GRASS data base, location, and mapset that are to be the user's *current data base*, *current location*, and *current mapset* for the duration of the session; any maps created by the user during the session will be stored under the *current mapset* ($MAPSET) set at the session's outset.

The user can add, modify, and delete data layers that exist under his *current mapset*. Although the user can also *access* (i.e., use) data that are stored under *other* mapsets in the same GRASS location, the user can only make permanent changes (create or modify data) located in the *current mapset*. The user's *mapset search path* lists the order in which other mapsets in the same GRASS location can be searched and their data accessed by the user. The user can modify the listing and order in which these mapsets are accessed by modifying the mapset search path; this can be done using the *g.mapsets* command. This program allows the user to use other's relevant map data without altering the original data layer, and without taking up disk space with a copy of the original map.

g.mapsets shows the user available mapsets under the current GRASS location, lists mapsets to which the user currently has access, and lists the order in which accessible mapsets will be accessed by GRASS programs searching for data files. The user is then given the opportunity to add or delete mapset names from his search path, or modify the order in which mapsets will be accessed.

When the user specifies the name of a data base element file (e.g., a particular vector file, raster file, imagery group file, etc.) to a GRASS program, the program searches for the named file under each of the mapsets listed in the user's mapset search path in the order listed there until the program finds a file of the given name. (Users can also specify a file by its mapset, to make explicit the mapset from which the file is to be drawn; e.g., the command:

g.copy rast='soils.file@PERMANENT',my.soils

ensures that a new file named *my.soils* is to be a copy of the file *soils.file* from the mapset PERMANENT.)

It is common for a user to have the special mapset **PERMANENT** included in his mapset search path, as this mapset typically contains finished base maps relevant to many applications. Often, other mapsets which contain sets of interpreted map layers will be likewise included in the user's mapset search path. Suppose, for example, that the mapset *Soil_Maps* contains interpreted soils map layers to which the user wants access. The mapset *Soil_Maps* should then be included in the user's *search path* variable.

The *mapset search path* is saved as part of the current mapset. When the user works with that mapset in subsequent GRASS sessions, the previously saved mapset search path will be used (and will continue to be used until it is modified by the user with *g.mapsets*).

Flags:

-l List all available mapsets under the user's current location.

-p Print the user's current mapset search path to standard output.

Parameter:

mapset=*name*[,*name*,...] Name(s) of existing GRASS mapset(s) under the current location.

g.mapsets sets the current *mapset search path* to the *mapsets* named on the command line. If **g.mapsets** is typed but no *mapset* names are specified by the user on the command line, the program will print the user's current mapset search path, list available mapsets, and prompt the user for a new mapset search path listing.

NOTES

Users can restrict others' access to their mapset files through use of the GRASS program *g.access*. Mapsets to which access is restricted can still be listed in another's mapset search path; however, access to these mapsets will remain restricted.

SEE ALSO

g.access, g.copy, g.gisenv, g.list

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

Greg Koerper, ManTech Environmental Technology, Inc.

NAME

g.region – Program to manage the boundary definitions for the geographic region.
(GRASS Region Management Program)

SYNOPSIS

g.region
g.region help
g.region [-dgpu] [region=*name*] **[raster**=*name*] **[vector**=*name*] **[sites**=*name*] **[3dview**=*name*] **[n**=*value*] **[s**=*value*] **[e**=*value*] **[w**=*value*] **[res**=*value*] **[nsres**=*value*] **[ewres**=*value*] **[zoom**=*name*] **[align**=*name*] **[save**=*name*]

DESCRIPTION

The *g.region* program allows the user to manage the settings of the current geographic region. These regional boundaries can be set by the user directly and/or set from a region definition file (stored under the *windows* directory in the user's current mapset). The user can create, modify, and store as many geographic region definitions as desired for any given mapset. However, only one of these geographic region definitions will be current at any given moment, for a specified mapset; i.e., GRASS programs that respect the geographic region settings will use the current geographic region settings.

INTERACTIVE PROGRAM USE: MAIN MENU

The main menu consists of an information section listing the current GRASS data base LOCATION, MAPSET, and CURRENT REGION, followed by user options:

```
------------------------------------------------------------
|                    REGION FACILITY                       |
| LOCATION: sample                    MAPSET: grass        |
|                                                          |
| CURRENT REGION: N=5167600  S=5156755  RES=50  ROWS=216   |
|                 E=729314   W=705924   RES=50  COLS=467   |
| PROJECTION: 1 (UTM)                                      |
| ZONE: 13                                                 |
|                                                          |
|                                                          |
| Please select one of the following options               |
|                                                          |
|    Current Region              Region Database           |
|                                                          |
| 1 Modify current region      6 Save current region in    |
|   directly                     the database              |
|                                                          |
| 2 Set from default region    7 Create a new region       |
|                                                          |
| 3 Set from a database        8 Modify an existing region |
|   region                                                 |
|                                                          |
| 4 Set from a raster map                                  |
|                                                          |
| 5 Set from a vector map                                  |
|                                                          |
| RETURN to quit                                           |
------------------------------------------------------------
```

DEFINITIONS

Region:

Here, a *region* refers to a geographic area with some defined boundaries, based on a specific map coordinate system and map projection. Each region also has associated with it the specific east-

west and north-south resolutions of its smallest units (rectangular units called "cells").

The region's boundaries are given as the northernmost, southernmost, easternmost, and westernmost points that define its extent. The north and south boundaries are commonly called *northings*, while the east and west boundaries are called *eastings*.

The region's cell resolution defines the size of the smallest piece of data recognized (imported, analyzed, displayed, stored, etc.) by GRASS programs affected by the current region settings. The north-south and east-west cell resolutions need not be the same, thus allowing non-square data cells to exist.

Default Region:
Each GRASS LOCATION_NAME has a fixed geographic region, called the default geographic region (stored in the region file DEFAULT_WIND under the special mapset PERMANENT), that defines the extent of the data base. While this provides a starting point for defining new geographic regions, user-defined geographic regions need not fall within this geographic region.

Current Region:
Each mapset has a current geographic region. This region defines the geographic area in which all GRASS displays and analyses will be done. Data will be resampled, if necessary, to meet the cell resolutions of the current geographic region setting.

Region Data Base:
Each GRASS MAPSET may contain any number of pre-defined, and named, geographic regions. These region definitions are stored in the user's current mapset location under the *windows* directory (also referred to as the user's data base of region definitions). Any of these pre-defined geographic regions may be selected, by name, to become the current geographic region. Users may also access saved region definitions stored under other mapsets in the current location, if these mapsets are included in the user's mapset search path.

REGION EDIT PROMPT

Most of the options will require the user to edit a geographic region, be it the current geographic region or one stored in the user's data base of region definitions (the *windows* directory). A standard prompt is used to perform this edit. An example is shown below:

```
------------------------------------------------------------
|                     IDENTIFY REGION                      |
|                                                          |
|          ========= DEFAULT REGION =========             |
|          |   Default North: 3402025.00     |             |
|          |                                 |             |
|          |          ==YOUR REGION==        |             |
|          |          | NORTH EDGE |         |             |
|          |          | 3402025.00_|         |             |
|          |          |            |         |             |
| Def West: |WEST EDGE |            |EAST EDGE| Def East: |
| 233975.00 |233975.00_|            |236025.00_| 236025.00 |
|          |          | SOUTH EDGE |         |             |
|          |          | 3399975.00_|         |             |
|          |          ============           |             |
|          |                                 |             |
|          |   Default South: 3399975.00     |             |
|          ===================================             |
|                                                          |
|            Default   GRID RESOLUTION  Region             |
|             50.00   --- East-West ---  50.00_           |
|             50.00   -- North-South --  50.00_           |
|                                                          |
|                                                          |
|    AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE   |
------------------------------------------------------------
```

The fields NORTH EDGE, SOUTH EDGE, WEST EDGE and EAST EDGE, are the boundaries of the geographic region that the user can change. The fields Default North, Default South, Def West and Def East are the boundaries of the default geographic region that are displayed for reference and *cannot* be changed. The two GRID RESOLUTION Region fields (east-west, and north-south) are the geographic region's cell resolutions that the user can change. The two GRID RESOLUTION Default fields list the resolutions of the default geographic region; these are displayed for reference and cannot be changed here by the user.

REGION MANAGEMENT MENU OPTIONS

1 Modify the current geographic region directly

Allows the user to edit the current region.

2 Set current geographic region from default region

Copies the default region to the current geographic region, and then lets the user edit the current geographic region.

3 Set current geographic region from a data base geographic region

Allows the user to select a geographic region by name from the data base of geographic regions to become the current geographic region, and then lets the user edit the current geographic region.

Note: geographic region definition files may be selected from other mapsets as well, if accessible and in the user's mapset search path.

4 Set current geographic region from a raster (cell) map layer

Allows the user to select a raster map layer, copies the cell header for this map layer to the current geographic region, and then lets the user edit the current geographic region. This option is useful when subsequent GRASS operations will be used to produce a raster map layer from one input raster map layer and it is necessary that the result coincide with the input raster map layer.

5 Save the current geographic region (window) in the data base

Allows the user to save the current geographic region settings in the user's data base of such settings. These files are stored in the ***windows*** directory under the user's current mapset. This option is useful when the current geographic region is set directly using option 2, or even by another GRASS program (e.g., ***d.display***). This option installs an otherwise temporary geographic region setting into the geographic region definition data base for recall when needed.

6 Create a new data base geographic region setting

Creates a new geographic region definition in the user's data base of such settings in the ***windows*** directory under the current mapset, using the geographic region edit prompt described above. After the geographic region definition is created, the user is asked if this geographic region setting should also be used as the current geographic region.

7 Modify a data base geographic region setting

Modifies a geographic region setting (in the data base of such settings in the ***windows*** directory of the current mapset), using the geographic region edit prompt. After the changes have been made, the user is asked if this geographic region setting should also be used as the current geographic region.

NON-INTERACTIVE PROGRAM USE

Alternately, the user can modify the settings of the current geographic region by specifying all needed parameters on the command line. The user enters the command **g.region** *parms*, where *parms* are the following parameters and/or flags:

Flags:

-d	Set current region settings equal to default region settings.
-g	Print the current region settings (shell script style) in a format that can be given back to ***g.region*** on its command line.
-p	Print the current region settings.
-u	Do not update the current region file settings. Allows the user to temporarily use a different region setting, without saving this setting.

Parameters:

region=*name*	Make current region settings same as the named region file settings
raster=*name*	Make current region settings same as those in the named raster map's cell header. But see **zoom**=*name* option, below.
vector=*name*	Make the current region settings the same as those of the named vector map.
sites=*name*	Set the current region to the smallest region encompassing all coordinates in the named site_lists file, aligned with the current region.
3dview=*name*	Make current region settings same as those in the named 3dview file, which holds the region that was current when the 3dview was saved.
n=*value*	Set map coordinate value for the region's northern edge to x
s=*value*	Set map coordinate value for the region's southern edge to x
e=*value*	Set map coordinate value for the region's eastern edge to x
w=*value*	Set map coordinate value for the region's western edge to x
res=*value*	Set grid resolution (both north-south and east-west) to x
nsres=*value*	Set north-south grid resolution value to x
ewres=*value*	Set east-west grid resolution value to x
zoom=*name*	Set current region settings to the smallest region encompassing all non-zero data in the named raster map layer that fall inside the user's current region.

If the user also includes the **raster**=*name* option on the command line, **zoom**=*name* will set the current region settings to the smallest region encompassing all non-zero data in the named **zoom** map that fall inside the region stated in the cell header for the named **raster** map.

align=*name* Set the current resolution equal to that of the named raster map, and align the current region to a row and column edge in the named map. Alignment only moves the existing region edges outward to the edges of the next nearest cell in the named raster map -- not to the named map's edges. To perform the latter function, use the **raster**=*name* option.

save=*name* Save current region settings in the named region file

EXAMPLES

g.region n=7360100 e=699000

will reset the northing and easting for the current region, but leave the south edge, west edge, and the region cell resolutions unchanged.

g.region –dp s=698000

will set the current region from the default region for the GRASS data base location, reset the south edge to 698000, and then print the result.

g.region n=n+1000 w=w500

The n=*value* may also be specified as a function of its current value: n=n+*value* increases the current northing, while n=n*value* decreases it. This is also true for s=*value*, e=*value*, and w=*value*. In this example the current region's northern boundary is extended by 1000 units and the current region's western boundary is decreased by 500 units.

g.region n=s+1000 e=w+1000

This form allows the user to set the region boundary values relative to one another. Here, the northern boundary coordinate is set equal to 1000 units larger than the southern boundary's coordinate value, and the eastern boundary's coordinate value is set equal to 1000 units larger than the western boundary's coordinate value. The corresponding forms s=n-*value* and w=e-*value* may be used to set the values of the region's southern and western boundaries, relative to the northern and eastern boundary values.

g.region raster=soils

This form will make the current region settings exactly the same as those given in the cell header file for the raster map layer *soils*.

g.region raster=soils zoom=soils

This form will first look up the cell header file for the raster map layer *soils*, use this as the current region setting, and then shrink the region down to the smallest region which still encompasses all non-zero data in the map layer *soils*. Note that if the parameter *raster=soils* were not specified, the zoom would move to encompass all non-zero data values in the soils map that were located within the current region setting.

g.region –up raster=soils

The –u option suppresses the re-setting of the current region definition. This can be useful when it is desired to only extract region information. In this case, the cell header file for the soils map layer is printed without changing the current region settings.

g.region –u raster=soils zoom=soils save=soils

This will zoom into the smallest region which encompasses all non-zero soils data values, and save the new region settings in a file to be called *soils* and stored under the *windows* directory in the

user's current mapset. The current region settings are not changed.

g.region −p

This will print the current region in the format:

```
projection:   1 (UTM)
zone:         15
north:        4294050.00
south:        4249950.00
east:         526050.00
west:         500950.00
nsres:        100.00
ewres:        100.00
rows:         441
cols:         251
```

g.region −g

The −g option prints the region in the following format:

```
n=4294050.00
s=4249950.00
e=526050.00
w=500950.00
nsres=100.00
ewres=100.00
```

This format does not have the rows and columns, but can be fed back into *g.region* on its command line.

The −p (or −g) option is recognized last. This means that all changes are applied to the region settings before printing occurs.

NOTE

After all updates have been applied, the current region's southern and western boundaries are (silently) adjusted so that the north/south distance is a multiple of the north/south resolution and that the east/west distance is a multiple of the east/west resolution.

SEE ALSO

d.display, d.zoom, g.access, g.mapsets

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

g.remove – Removes data base element files from the user's current mapset.
(GRASS File Management Program)

SYNOPSIS

g.remove
g.remove help
g.remove [**rast**=*name*[,*name*,...]] [**vect**=*name*[,*name*,...]] [**icon**=*name*[,*name*,...]] [**labels**=*name*[,*name*,...]] [**sites**=*name*[,*name*,...]] [**region**=*name*[,*name*,...]] [**group**=*name*[,*name*,...]]

DESCRIPTION

g.remove allows the user to remove specified data base element files from the current mapset. If **g.remove** is invoked without arguments on the command line, a menu will appear listing possible data element types, as below:

1 raster maps
2 vector maps
3 paint icon files
4 paint label files
5 site list files
6 region files
7 imagery group files

RETURN to exit

Once the element type is selected, the user is prompted to name a specific file of this element type for removal. (This list will vary, depending on what files currently exist in the user's mapset.) The specified file is removed, and the user is again prompted for the name of a file of this element type to be removed. When prompted for a file name, the user may enter **list** to see a list of existing files of this element type, or hit RETURN to get back to the above menu.

Alternately, the user can specify the data base element type and file(s) to be removed on the command line. Data base element types are specified by the names to the left, below.

Parameters:

rast=*name*[,*name*,...]	Name(s) of raster file(s) to be removed.
vect=*name*[,*name*,...]	Name(s) of vector file(s) to be removed.
icon=*name*[,*name*,...]	Name(s) of paint icon file(s) to be removed.
labels=*name*[,*name*,...]	Name(s) of paint labels file(s) to be removed.
sites=*name*[,*name*,...]	Name(s) of site list file(s) to be removed.
region=*name*[,*name*,...]	Name(s) of region file(s) to be removed.
group=*name*[,*name*,...]	Name(s) of imagery group file(s) to be removed.

The data base element file(s) named by the user on the command line are subsequently removed from the user's current mapset.

EXAMPLE

For example, the below command will cause the raster files named ***soils***, ***slope***, and ***temp***, the vector files named ***roads*** and ***rail***, and the imagery group files named ***nhap.1*** and ***nhap.2***, and these files' associated support files (e.g., cell header files, category files, etc.), to be removed from the user's current mapset.

g.remove rast=soils,slope,temp vect=roads,rail group=nhap.1,nhap.2

NOTE

If a particular data base element file has support files associated with it (e.g., as is commonly the case with raster files), *g.remove* will remove these support files along with the data base element file specified.

The user can only use *g.remove* to remove data files existing under the user's *current mapset*.

FILES

$GISBASE/etc/element_list lists the element types whose files can be removed by the user.

SEE ALSO

g.copy, g.list, g.rename

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

g.rename – To rename data base element files in the user's current mapset.
(GRASS File Management Program)

SYNOPSIS

g.rename
g.rename help
g.rename [rast=*old,new*] **[vect**=*old,new*] **[icon**=*old,new*] **[labels**=*old,new*] **[sites**=*old,new*]
[region=*old,new*] **[group**=*old,new*]

DESCRIPTION

g.rename allows the user to rename data base element files in the user's current mapset. The user can specify all necessary information to *g.rename* on the command line, by specifying: the type of data base element to be renamed (one or more of: **rast, vect, icon, labels, sites, region,** and **group**); the specific file element in the current mapset to be renamed (*old*); and the new name to be assigned to this file element (*new*) in the current mapset. The file element *old* is then renamed to *new*.

Users can also simply type **g.rename** without arguments on the command line, to receive a menu of existing data base element types and files from which to choose for possible renaming:

1	raster maps
2	binary vector maps
3	paint icon files
4	paint label files
5	site list files
6	region definition files
7	imagery group files

RETURN to exit

NOTE

If a data base element has support files (e.g., as is commonly the case with raster files), these support files also are renamed.

If the user attempts to rename a file to itself by setting the *new* file name equal to the *old* file name (e.g., **g.rename rast=soils,soils**), *g.rename* will not execute the rename, but instead state that no rename is needed. However, *g.rename* will allow the user to overwrite other existing files in the current mapset by making the *new* file name that of an already existing file.

SEE ALSO

g.copy, g.list, g.remove

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

g.setproj − Allows the user to create the PROJ_INFO and the PROJ_UNITS files to record the projection information associated with a current location.

SYNOPSIS

g.setproj

DESCRIPTION

Allows a user to create a PROJ_INFO file in the PERMANENT mapset of the current location. PROJ_INFO file is used to record the projection information associated with the specified mapset.

Notes:

User running *g.setproj* must own PERMANENT mapset. It is highly recommended to run *g.setproj* after creating a new location so that conversion programs (such as *v.proj can be run.*
The current location must not contain a PROJ_INFO or PROJ_UNITS file.
The user will be prompted for the projection name.
The specification of any projection other than ll will generate a request to the user for a name of a standard ellipse.
The projections of aea, lcc, merc, and tmerc will generate a request to the user for the prime meridian and standard parallel for the output map.
The projection of stp will generate a request to the user for the state abbreviation and choice of zone for the output map.
The user will be prompted for the spheroid and zone of the UTM projection.

SEE ALSO

v.proj m.proj

AUTHOR

Irina Kosinovsky, US Army CERL, Champaign, Il

NAME

g.tempfile – Creates a temporary file and prints the file name
(GRASS File Management Program)

SYNOPSIS

g.tempfile help
g.tempfile pidfR=*value*

DESCRIPTION

g.tempfile is designed for shell scripts that need to use large temporary files. GRASS provides a mechanism for temporary files that does not depend on /tmp. GRASS temporary files are created in the data base with the assumption that there will be enough space under the data base for large files. GRASS periodically removes temporary files that have been left behind by programs that failed to remove them before terminating.

g.tempfile creates an unique file and prints the name. The user is required to provide a process-id which will be used as part of the name of the file. Most Unix shells provide a way to get the process id of the current shell. For /bin/sh and /bin/csh this is $$. It is recommended that $$ be specified as the process-id for *g.tempfile.*

EXAMPLE

For /bin/sh scripts the following syntax should be used:

```
temp1=`g.tempfile pid=$$`
temp2=`g.tempfile pid=$$`
```

For /bin/csh scripts, the following can be used:

```
set temp1=`g.tempfile pid=$$`
set temp2=`g.tempfile pid=$$`
```

NOTES

Each call to *g.tempfile* creates a different (i.e. unique) name.

Although GRASS does eventually get around to removing tempfiles that have been left behind, the programmer should make every effort to remove these files. They often get large and take up disk space. If you write /bin/sh scripts, learn to use the /bin/sh *trap* command. If you write /bin/csh scripts, learn to use the /bin/csh *onintr* command.

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

g.version – Outputs the GRASS version number and date.
(GRASS File Management Program)

SYNOPSIS

g.version
g.version help

DESCRIPTION

g.version prints to standard output the GRASS version number and date, in the form:

GRASS 4.0 (Summer 1991)

NOTES

This program requires no command line arguments; the user simply types **g.version** on the command line to see the version number and date of the GRASS software currently being run by the user.

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

i.cca – Canonical components analysis (cca) program for image processing.
(GRASS Image Processsing Program)

SYNOPSIS

i.cca
i.cca help
i.cca group=*name* **subgroup**=*name* **signature**=*name* **output**=*name*

DESCRIPTION

i.cca is an image processing program that takes from two to eight (raster) band files and a signature file, and outputs the same number of raster band files transformed to provide maximum separability of the categories indicated by the signatures. This implementation of the canonical components transformation is based on the algorithm contained in the LAS image processing system.

Typically the user will use the *i.class* program to collect a set of signatures and then pass those signatures along with the raster band files to *i.cca*. The raster band file names are specified on the command line by giving the group and subgroup that were used to collect the signatures.

The output raster map names are built by appending a ".1", ".2", etc. to the output raster map name specified on the command line.

Parameters:

group=*name*	Name of the imagery group to which the 2 to 8 raster band files used belong.
subgroup=*name*	Name of the imagery subgroup to which the 2 to 8 raster band files used belong.
signature=*name*	Name of an ASCII file containing spectral signatures.
output=*name*	Output raster file prefix name. The output raster map layer names are built by appending a ".1", ".2", etc. onto the *output* name specified by the user.

NOTES

i.cca respects the current geographic region definition and the current mask setting while performing the transformation.

SEE ALSO

Schowengerdt, Robert A. **Techniques for Image Processing and Classification in Remote Sensing**, Academic Press, 1983.

i.class, i.pca, r.covar, r.mapcalc

AUTHORS

David Satnik, GIS Laboratory, Central Washington University
Ali R. Vali, Space Research Center, University of Texas, Austin

NAME

i.class − An imagery function that generates spectral signatures for an image by allowing the user to outline regions of interest. The resulting signature file can be used as input for ***i.maxlik*** or as a seed signature file for ***i.cluster***.
(GRASS Image Processing Program)

SYNOPSIS

i.class

DESCRIPTION

i.class performs the first pass in the GRASS two-pass supervised image classification process; the GRASS program ***i.maxlik*** executes the second pass. Both programs must be run to generate a classified map in GRASS raster format.

i.class is an interactive program that allows the user to outline a region on the screen and calculate the spectral signature based on the cells that are within that region. During this process the user will be shown a histogram of the region for each image band. The user can also display the cells of the image bands which fall within a user-specified number of standard deviations from the means in the spectral signature. By doing this, the user can see how much of the image is likely to be put into the class associated with the current signature.

The spectral signatures that result are composed of region means and covariance matrices. These region means and covariance matrices are used in the second pass (***i.maxlik***) to classify the image.

Alternatively, the spectral signatures generated by ***i.class*** can be used for seed means for the clusters in the ***i.cluster*** program.

USER INPUTS

The first screen in the program ***i.class*** asks the user for the imagery ***group*** and ***subgroup*** to be analyzed:

LOCATION: *location* **SUPERVISED CLASSIFIER** **MAPSET:** *demo*

Please select the group and subgroup to be analyzed

GROUP: *spot*_____ **(list will show available groups)**
SUBGROUP: *123*_____ **(list will show available subgroups)**

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

The ***group*** should contain the imagery bands that the user wishes to classify. The ***subgroup*** is a subset of this group. The user must create a group and a subgroup by running the GRASS program ***i.group*** before running ***i.class***. The subgroup should contain only the image bands that the user wishes to classify. Note that this subgroup must contain more than one band.

After the first screen, the program asks the user for the name of the resulting signature file. The signature file is both the output file for ***i.class*** and the required input file for the GRASS program ***i.maxlik***. It contains the region means and covariance matrices that are used to classify an image in ***i.maxlik***.

After entering the resulting signature file name, the user is asked to enter the name of a seed signature file. This is optional. A "seed" signature file is a previously created signature file. Such a seed signature file may be the result of an earlier run of ***i.class***. The seed signature file is copied into the new resulting

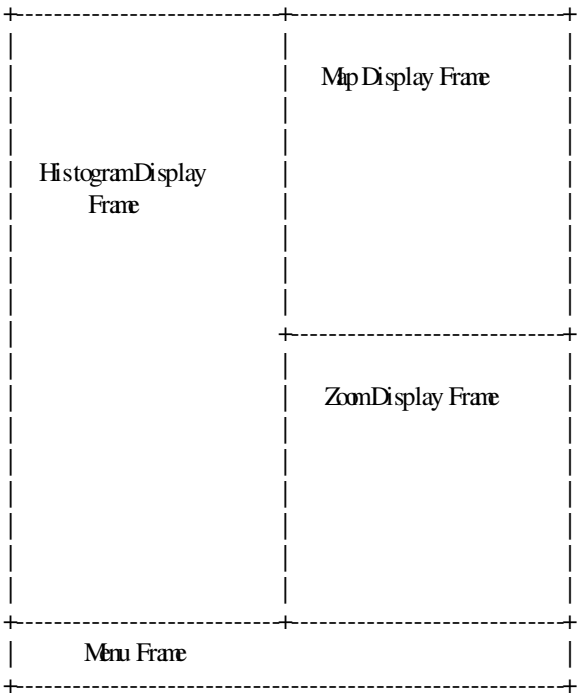
signature file before any new signatures are added by ***i.class***. In this way, you can collect the work from several sessions with ***i.class*** into one signature file.

Next, the user is asked to enter the name of the raster map to be displayed during the process of outlining regions. Typically, the user will want to enter the name of a color composite previously created by ***i.composite***. However, the user can enter the name of any existing raster map. This leaves the potential for using a raster map not directly derived from the image as a backdrop on which the user can outline the classes of interest.

At this point the ***i.class*** graphics screen will be drawn on the graphics monitor and the user will be directed to use the mouse. From this point on the user will primarily work with the mouse, selecting options from the menus and outlining regions on the screen. The only time that the user will need to return to the text terminal is to enter names for the signatures created.

THE DISPLAY FRAMES

The display frame layout that ***i.class*** uses is represented below for reference.



THE MENUS

All of the menus in the ***i.class*** program are displayed across the bottom of the graphics monitor in the Menu Frame. To select an option from one of these menus, simply place the cursor over your selection and press any button on the mouse. Each of the menus is discussed in the following paragraphs.

The Command Menu

The Command Menu includes the following selections:

Zoom - This command allows the user to outline a rectangular region in either the Map or Zoom Display Frames and the region is displayed, magnified, to fit in the Zoom Display Frame. A red rectangle is drawn in the Map Display Frame, indicating what area the Zoom Display Frame shows.

To outline the rectangular region simply use any mouse button to anchor the first corner of the border and then use any button to choose the other corner.

Define region - This selection takes the user to the Region Menu. This menu includes the options that allow the user to outline a region of interest on the displayed raster map.

Redisplay map - This selection takes the user to the Redisplay Menu. The Redisplay Menu allows the user to redraw map display frames.

Analyze region - This selection starts the process of analyzing the currently defined region. A histogram of the defined region will be displayed for each band. On the histogram for each band, the mean, standard deviation, minimum cell value and maximum cell value are marked. The histograms are automatically scaled in an attempt to fit the data into the space available, but it is possible that all of the data will not fit. In this case, as much of the data as possible, centered around the mean, will be displayed. After the histograms are displayed, the user will be given the Signature Menu.

Quit - The user should make this selection to end the session with ***i.class***.

The Region Menu

The Region Menu contains the following selections:

Erase region - This selection erases any currently defined region.

Draw region - This selection allows the user to use the mouse to draw a region on either the Map or Zoom Display Frame. An explanation of which mouse buttons to use is displayed in the Menu Frame. The user does not need to try to complete the region boundary. The last line of the region will be added when the user selects the Complete region option on the Region Menu.

Restore last region - This selection restores the last region that was drawn. After a region is completed, it will be saved to be restored later. Only one previous region is saved.

Complete region - This selection completes the region that is currently being drawn. As noted above, it saves the complete region to be restored later, if needed. Once the user has made a complete region, it can be analyzed with the Analyze Region selection on the Command Menu.

Done - Use this selection to return to the Command Menu.

The Redisplay Map Menu

The Redisplay Map Menu has the following selections, which are useful to redraw the raster maps displayed in the Map and Zoom Display Frames.

Map geographic region - This selection causes the raster map in the Map Display Frame to be redrawn.

Zoom region - This selection causes the Zoom Display Frame to be redrawn.

Both - This selection causes both the Map and Zoom Display Frames to be redrawn.

Cancel - Use this selection if you do not want to redisplay either of the above regions. The user will be returned to the Command Menu.

The Analyze Region Menu

The Analyze Region Menu contains the Signature Menu, which allows the user to set the number of standard deviations and the display color, and then to display (as an overlay) the cells that match the signature within the number of standard deviations specified. Note that once the matching cells are displayed, the Map Display Frame must be redisplayed to see only the original raster map again. The following selections are available on the Signature Menu:

Set std dev's - This selection allows the user to set the number of standard deviations from the mean for the maximum and minimum range. The maximum and minimum range is used when finding the cells that "match" the signature. The user is presented with a menu of typical choices and an "Other" option. If the "Other" option is selected, enter the number of standard deviations from the keyboard on the text terminal.

Otherwise, the selected option will be used. When the number of standard deviations is set, the histograms for each band will be redrawn with the maximum and minimum range marked.

Note that the number in parentheses on this selection is the current number of standard deviations.

Set color - This selection allows the user to set the color for the display of cells that "match" the current signature. The user is presented with a menu of color choices. The color selected will be used when the Display Matches Menu selection is made.

Note that the color in parentheses on this selection is the current color for display.

Display matches - This selection displays the cells that "match" the current signature in the current color. A cell "matches" the current signature if the cell value in each band is between the minimum range and maximum range for that band defined by the number of standard deviations currently set.

Done - When this selection is chosen, the user will be asked whether or not he/she would like to save the current signature. If the user answers with the "Yes" selection, he/she will be asked to enter a description for the resultant signature file on the text terminal keyboard. The saved signature file description will be used by ***i.maxlik*** to name the category that is created from the current signature. After either a "No" answer or the signature description is entered, the user is returned to the Command Menu.

NOTES

i.class uses the current MASK to generate the overlay for cells that match a signature. As a result, if a MASK already exists it will be removed during the execution of this program.

The cell values in the image bands cannot fall outside of the range of 0 to 255. ***i.class*** will report an error if they do.

i.class, like some of the other imagery programs, does not use the standard GRASS display frames. After running ***i.class***, you will need to create a display frame (e.g., using ***d.frame*** or ***d.erase***) before you can use most of the GRASS display (d.) commands.

i.group must be run before ***i.class*** to create an imagery group and a subgroup containing the image bands to be classified.

The user can perform a supervised image classification by running ***i.class*** followed by ***i.maxlik***. The user can perform an unsupervised classification by running ***i.cluster*** followed by ***i.maxlik***.

i.class is interactive and requires no command line arguments. The user must be running a graphics display monitor (see ***d.mon***) to run this program.

SEE ALSO

GRASS Tutorial: Image Processing

d.frame, d.mon, g.region, i.cca, i.cluster, i.composite, i.group, i.maxlik, r.mapcalc, r.mask

AUTHOR

David Satnik, Central Washington University GIS Laboratory

NAME

i.cluster − An imagery function that generates spectral signatures for land cover types in an image using a clustering algorithm. The resulting signature file is used as input for *i.maxlik*, to generate an unsupervised image classification.
(GRASS Image Processing Program)

SYNOPSIS

i.cluster
i.cluster help
i.cluster [**-q**] **group**=*name* **subgroup**=*name* **sigfile**=*name* **classes**=*value* [**seed**=*name*] [**sample**=*row_interval,col_interval*] [**iterations**=*value*] [**convergence**=*value*] [**separation**=*value*] [**min_size**=*value*] [**reportfile**=*name*]

DESCRIPTION

i.cluster performs the first pass in the *GRASS* two-pass unsupervised classification of imagery, while the GRASS program *i.maxlik* executes the second pass. Both programs must be run to complete the unsupervised classification.

i.cluster is a clustering algorithm that reads through the (raster) imagery data and builds pixel clusters based on the spectral reflectances of the pixels. The pixel clusters are imagery categories that can be related to land cover types on the ground. The spectral distributions of the clusters (which will be the land cover spectral signatures) are influenced by six parameters set by the user. The first parameter set by the user is the initial number of clusters to be discriminated. *i.cluster* starts by generating spectral signatures for this number of clusters and "attempts" to end up with this number of clusters during the clustering process. The resulting number of clusters and their spectral distributions, however, are also influenced by the range of the spectral values (category values) in the image files and the other parameters set by the user. These parameters are: the minimum cluster size, minimum cluster separation, the percent convergence, the maximum number of iterations, and the row and column sampling intervals.

The cluster spectral signatures that result are composed of cluster means and covariance matrices. These cluster means and covariance matrices are used in the second pass (*i.maxlik*) to classify the image. The clusters or spectral classes result can be related to land cover types on the ground.

OPTIONS

The program can be run either non-interactively or interactively. It will be run non-interactively if the user specifies the name of group file, the name of subgroup file, the name of a file to contain result signatures, the initial number of clusters to be discriminated, and optionally other parameters (see below) on the command line using the form:

> **i.cluster** [**-q**] **group**=*name* **subgroup**=*name* **sigfile**=**name** **classes**=*value* [**seed**=*name*] [**sample**=*row_interval,col_interval*] [**iterations**=*value*] [**convergence**=*value*] [**separation**=*value*] [**min_size**=*value*] [**reportfile**=*name*]

where the *group* should contain the imagery files that the user wishes to classify. The *subgroup* is a subset of this group. The user must create a group and subgroup by running the *GRASS* program *i.group* before running *i.cluster*. The subgroup should contain only the imagery band files that the user wishes to classify. Note that this subgroup must contain more than one band file. The purpose of the group and subgroup is to collect map layers for classification or analysis. The *sigfile* is the file to contain result signatures which can be used as input for *i.maxlik*. The classes value is the initial number of clusters to be discriminated; any parameter values left unspecified are set to their default values. Alternatively, the program will be run interactively if the user types only *i.cluster*; in this case the program will prompt the user for parameter values using the standard *GRASS* parser interface described in the manual entry for parser.

Flags:

-q Run quietly. Suppresses output of program percent-complete messages and the time elapsed from the beginning of the program. If this flag is not used, these messages are printed out.

Parameters:

group=*name* The name of the group file which contains the imagery files that the user wishes to classify.

subgroup=*name* The name of the subset of the group specified in group option, which must contain only imagery band files and more than one band file. The user must create a group and a subgroup by running the *GRASS* program *i.group* before running *i.cluster*.

sigfile=*name* The name assigned to output signature file which contains signatures of classes and can be used as the input file for the *GRASS* program *i.maxlik* for an unsupervised classification.

classes=*value* The number of clusters that will initially be identified in the clustering process before the iterations begin.

seed=*name* The name of a seed signature file is optional. The seed signatures are signatures that contain cluster means and covariance matrices which were calculated prior to the current run of *i.cluster*. They may be acquired from a previously run of *i.cluster* or from a supervised classification signature training site section (e.g., using the signature file output by *i.class*). The purpose of seed signatures is to optimize the cluster decision boundaries (means) for the number of clusters specified.

sample=*row_interval,col_interval*
These numbers are optional with default values based on the size of the data set such that the total pixels to be processed is approximately 10,000 (consider round up).

iterations=*value* This parameter determines the maximum number of iterations which is greater than the number of iterations predicted to achieve the optimum percent convergence. The default value is 30. If the number of iterations reaches the maximum designated by the user; the user may want to rerun *i.cluster* with a higher number of iterations (see *reportfile*).
Default: 30

convergence=*value* A high percent convergence is the point at which cluster means become stable during the iteration process. The default value is 98.0 percent. When clusters are being created, their means constantly change as pixels are assigned to them and the means are recalculated to include the new pixel. After all clusters have been created, *i.cluster* begins iterations that change cluster means by maximizing the distances between them. As these means shift, a higher and higher convergence is approached. Because means will never become totally static, a percent convergence and a maximum number of iterations are supplied to stop the iterative process. The percent convergence should be reached before the maximum number of iterations. If the maximum number of iterations is reached, it is probable that the desired percent convergence was not reached. The number of iterations is reported in the cluster statistics in the report file (see *reportfile*).
Default: 98.0

separation=*value* This is the minimum separation below which clusters will be merged in the iteration process. The default value is 0.0. This is an image-specific number (a "magic" number) that depends on the image data being classified and the number of final clusters that are acceptable. Its determination requires experimentation. Note that as the minimum class (or cluster) separation is increased, the maximum number of iterations should also be increased to achieve this separation with a high percentage of convergence (see *convergence*).
Default: 0.0

min_size=*value* This is the minimum number of pixels that will be used to define a cluster, and is therefore the minimum number of pixels for which means and covariance matrices

will be calculated.
Default: 17

reportfile=*name* The reportfile is an optional parameter which contains the result, i.e., the statistics for each cluster. Also included are the resulting percent convergence for the clusters, the number of iterations that was required to achieve the convergence, and the separability matrix.

NOTES

Running in command line mode, ***i.cluster*** will overwrite the output signature file and reportfile (if required by the user) without prompting if the files existed.

SEE ALSO

GRASS Tutorial: Image Processing

i.class, i.group, i.gensig, i.maxlik

AUTHORS

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory
Tao Wen, University of Illinois at Urbana-Champaign, Illinois

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

i.colors – An imagery function that creates colors for imagery groups.
(GRASS Image Processing Program)

SYNOPSIS

i.colors

DESCRIPTION

i.colors allows the user to interactively assign red, green, and blue colors to the band files in an imagery group while viewing the display of the combined bands on the graphics monitor.

The user is asked to select an imagery group. Band files in the group that are currently assigned to the red, green, and blue color bands are displayed along the left edge of the user's graphics display frame, and a composite image of these bands is displayed in the right portion of the display frame. A small matrix appears in the lower left portion of the display frame, and reflects current red, green, and blue band file color assignments. Red, green, and blue color boxes appear next to the names of the image band files to which they are currently assigned. The user can change current color assignments by clicking the mouse cursor beside desired color assignments.

A menu along the bottom of the graphics display frame indicates that the user can also replot and enlarge the composite image on display, or quit the program, by clicking the mouse cursor on the desired option.

The GRASS program *i.group* can also be used to assign red, green, and blue colors to imagery group band files.

NOTES

This program is not yet complete. To assign colors to an imagery group use the GRASS program *i.group*.

This program is interactive and requires no command line arguments. The user must be running a graphics monitor to use this program.

SEE ALSO

GRASS Tutorial: Image Processing

d.mon, hsv.rgb.sh, i.composite, i.grey.scale, i.group, i.his.rgb, i.rgb.his, r.mapcalc, rgb.hsv.sh

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

i.composite − An imagery function that creates a color composite image from three imagery band files specified by the user.
(GRASS Image Processing Program)

SYNOPSIS

i.composite

DESCRIPTION

i.composite creates a color composite image from three band files specified by the user. The user specifies the bands to be used by assigning a red, blue, and/or green color to each band. The resulting image is a raster map layer of raw spectral data composed of the three bands chosen by the user. The color composite can then be displayed, painted, or manipulated as would any raster map layer in GRASS.

The first prompt asks the user for the imagery group whose files are to be used.

The following menu is then displayed:

Please indicate which files to use for red, green, and blue colors. You may leave any color out. You may specify more than one color per file. However, each color may only be specified once. For example, to get a full color image, specify r,g,b for 3 different files. To get a grey scale image, specify rgb for a single file.

b__ spot.1
g__ spot.2
r__ spot.3
___ spotclass
___ spotreject

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

The user is then allowed to check the choice of bands:

Colors assigned as follows:

RED: spot.3@mapsetname
GREEN: spot.2@mapsetname
BLUE: spot.1@mapsetname

Look ok? (y/n) [y]

The color table that is created has 1000 colors (10 saturation levels (or shades) per primary color (blue, green, red)). The number of colors that can be displayed at one time on a color graphics monitor will depend on the graphics monitor being used. For example, if the graphics monitor can only display 512 colors at one time, then the user must run the GRASS command **d.colormode mode=fixed** before displaying the raster map layer. The colors that cannot be displayed will be assigned to the nearest displayable color, and the raster map layer will retain its relative color accuracy. If the user is in *float* colormode, however, the raster map layer displayed on the graphics monitor will not accurately reflect the map's real color assignments.

The user is then asked to name the composite image raster map layer. The percentage completed is echoed to the screen and *r.support* files are created automatically.

NOTES

The user should always check the geographic region settings before running most imagery commands. It is very easy for the boundaries of the geographic region to be completely off the image. Before running *i.composite*, or other imagery commands, the user should probably set the geographic region to match that of the raster map layers to be read. This can be accomplished using option 4 of the *g.region* command.

This program is interactive and requires no command line arguments.

SEE ALSO

GRASS Tutorial: Image Processing

d.colormode, d.his, d.rast, g.region, i.colors, i.grey.scale, i.group, r.support

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

i.fft – Fast Fourier Transform (FFT) for image processing.
(GRASS Image Processing Program)

SYNOPSIS

i.fft
i.fft help
i.fft input_image=*name* **real_image**=*name* **imaginary_image**=*name* [**range**=*value*]

DESCRIPTION

i.fft is an image processing program based on the algorithm given by Press (1988), with enhancements by Vali (1990), that processes a single input raster map layer (***input_image***) and constructs the real and imaginary Fourier components in frequency space.

Parameters:

input_image=*name*	Input raster map layer on which the fast Fourier transform is run.
real_image=*name*	Output real part arrays stored as raster map layer.
imaginary_image=*name*	Output imaginary part arrays stored as raster map layer.
range=*value*	Range of values used during fast Fourier transformation.

The real and imaginary components are stored as arrays of doubles in the ***cell_misc*** directory (for use in the inverse transform program, ***i.ifft***), and are also scaled and formatted into the ***real_image*** and ***imaginary_image*** raster map layers for inspection, masking, etc. In these raster map layers the low frequency components are in the center and the high frequency components are toward the edges. The ***input_image*** need not be square; before processing, the X and Y dimensions of the ***input_image*** are padded with zeroes to the next highest power of two in extent (i.e., 256 x 256 is processed at that size, but 200 x 400 is padded to 256 x 512). The cell category values for viewing, etc., are calculated by taking the natural log of the actual values then rescaling to 255, or whatever optional range is given on the command line, as suggested by Richards (1986). A color table is assigned to the resultant map layer.

The current geographic region and mask settings are respected when reading the input file. The presence of a mask will, in general, make the resulting fast Fourier transform invalid, or at least difficult to interpret.

SEE ALSO

Numerical Recipes in C, by William H. Press, Cambridge University Press, 1988.

Remote Sensing Digital Image Analysis, by John A. Richards, Springer-Verlag, 1986.

Personnal communication, between progam author and Ali R. Vali, Space Research Center, University of Texas, Austin, 1990.

i.cca, i.class, i.ifft, i.pca

AUTHOR

David Satnik, GIS Laboratory, Central Washington University

NAME

i.gensig – Generates statistics for *i.maxlik* from raster map layer.
(GRASS Imagery Program)

SYNOPSIS

i.gensig
i.gensig help
i.gensig trainingmap=*name* **group=***name* **subgroup=***name* **signaturefile=***name*

DESCRIPTION

i.gensig is a non-interactive method for generating input into *i.maxlik.* It can be used as the first pass in the GRASS two-pass classification process (instead of i.cluster or i.class). It reads a raster map layer, called the training map, which has some of the pixels or regions already classified. *i.gensig* will then extract spectral signatures from an image based on the classification of the pixels in the training map and make these signatures available to *i.maxlik*.

The user would then execute the GRASS program *i.maxlik* to actually create the final classified map.

COMMAND LINE OPTIONS

Parameters

trainingmap
ground truth training map

This map must be prepared by the user in advance. Programs like *v.digit* or *r.digit* can be used to define representative areas of the classes the user defines to be in the image. Of course other methods could be devised by the user for creating this training map - *i.gensig* makes no assumption about the origin of this map layer. It simply creates signatures for the classes defined in the training map for the image to be classified (the image is specified in other options - see below).

group imagery group

This is the name of the group that contains the band files which comprise the image to be analyzed. The *i.group* command is used to construct groups of raster layers which comprise an image.

subgroup
subgroup containing image files

This names the subgroup within the group that selects a subset of the bands to be analyzed. The *i.group* command is also used to prepare this subgroup. The subgroup mechanism allows the user to select a subset of all the band files that form an image.

signaturefile
resultant signature file

This is the resultant signature file (containing the means and covariance matrices) for each class in the training map that is associated with the band files in the subgroup select (see above).

INTERACTIVE MODE

If none of the arguments are specified on the command line, *i.gensig* will interactively prompt for the names of these maps and files.

It should be noted that interactive mode here only means interactive prompting for maps and files. It does not mean visualization of the signatures that result from the process.

SEE ALSO

i.group for creating groups and subgroups.

v.digit, r.digit for interactively creating the training map.

i.cluster for unsupervised clustering as an alternative to i.gensig to create signatures.

i.class for a graphic/interactive as an alternative to i.gensig to create signatures.

AUTHOR

Michael Shapiro, U.S. Construction Engineering Research Laboratory

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

i.gensigset – generate statistics for ***i.smap*** from raster map layer.
(GRASS Imagery Program)

SYNOPSIS

i.gensigset
i.gensigset help
i.gensigset trainingmap=*name* **group=***name* **subgroup=***name* **signaturefile=***name* **[maxsig=***value]*

DESCRIPTION

i.gensigset is a non-interactive method for generating input into ***i.smap.*** It is used as the first pass in the a two-pass classification process. It reads a raster map layer, called the training map, which has some of the pixels or regions already classified. ***i.gensigset*** will then extract spectral signatures from an image based on the classification of the pixels in the training map and make these signatures available to ***i.smap.***

The user would then execute the GRASS program ***i.smap*** to create the final classified map.

COMMAND LINE OPTIONS

Parameters

trainingmap
ground truth training map

This raster layer, supplied as input by the user, has some of its pixels already classified, and the rest (probably most) of the pixels unclassified. Classified means that the pixel has a non-zero value and unclassified means that the pixel has a zero value.

This map must be prepared by the user in advance. The user must use ***r.digit,*** a combination of ***v.digit*** and ***v.to.rast,*** or some other import/developement process (e.g., ***v.in.transects)*** to define the areas representative of the classes in the image.

At present, there is no fully-interactive tool specifically designed for producing this layer.

group imagery group

This is the name of the group that contains the band files which comprise the image to be analyzed. The ***i.group*** command is used to construct groups of raster layers which comprise an image.

subgroup
subgroup containing image files

This names the subgroup within the group that selects a subset of the bands to be analyzed. The ***i.group*** command is also used to prepare this subgroup. The subgroup mechanism allows the user to select a subset of all the band files that form an image.

signaturefile
resultant signature file

This is the resultant signature file (containing the means and covariance matrices) for each class in the training map that is associated with the band files in the subgroup selected.

maxsig maximum number of sub-signatures in any class
default: 10

The spectral signatures which are produced by this program are "mixed" signatures (see NOTES). Each signature contains one or more subsignatures (represeting subclasses). The algorithm in this program starts with a maximum number of subclasses and reduces this number to a minimal number of subclasses which are spectrally distinct. The user has the option to set this starting value with this option.

INTERACTIVE MODE

If none of the arguments are specified on the command line, ***i.gensigset*** will interactively prompt for the names of these maps and files.

It should be noted that interactive mode here only means interactive prompting for maps and files. It does not mean visualization of the signatures that result from the process.

NOTES

The algorithm in ***i.gensigset*** determines the parameters of a spectral class model known as a Gaussian mixture distribution. The parameters are estimated using multispectral image data and a training map which labels the class of a subset of the image pixels. The mixture class parameters are stored as a class signature which can be used for subsequent segmentation (i.e., classification) of the multispectral image.

The Gaussian mixture class is a useful model because it can be used to describe the behavior of an information class which contains pixels with a variety of distinct spectral characteristics. For example, forest, grasslands or urban areas are examples of information classes that a user may wish to separate in an image. However, each of these information classes may contain subclasses each with its own distinctive spectral characteristic. For example, a forest may contain a variety of different tree species each with its own spectral behavior.

The objective of mixture classes is to improve segmentation performance by modeling each information class as a probabilistic mixture with a variety of subclasses. The mixture class model also removes the need to perform an initial unsupervised segmentation for the purposes of identifying these subclasses. However, if misclassified samples are used in the training process, these erroneous samples may be grouped as a separate undesired subclass. Therefore, care should be taken to provided accurate training data.

This clustering algorithm estimates both the number of distinct subclasses in each class, and the spectral mean and covariance for each subclass. The number of subclasses is estimated using Rissanen's minimum description length (MDL) criteria [1]. This criteria attempts to determine the number of subclasses which "best" describe the data. The approximate maximum likelihood estimates of the mean and covariance of the subclasses are computed using the expectation maximization (EM) algorithm [2,3].

REFERENCES

[1] J. Rissanen, "A Universal Prior for Integers and Estimation by Minimum Description Length," *Annals of Statistics,* vol. 11, no. 2, pp. 417-431, 1983.

[2] A. Dempster, N. Laird and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," ***J. Roy. Statist. Soc. B,*** vol. 39, no. 1, pp. 1-38, 1977.

[3] E. Redner and H. Walker, "Mixture Densities, Maximum Likelihood and the EM Algorithm," *SIAM Review,* vol. 26, no. 2, April 1984.

SEE ALSO

i.group for creating groups and subgroups

v.digit, r.digit for interactively creating the training map.

i.smap for creating a final classification layer from the signatures generated by *i.gensigset.*

AUTHOR

Charles Bouman, School of Electrical Engineering, Purdue University
Michael Shapiro, U.S. Construction Engineering Research Laboratory

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

i.grey.scale − An interactive imagery function that assigns a histogram contrast stretch grey scale color table to a raster map layer.
(GRASS Image Processing Program)

SYNOPSIS

i.grey.scale

DESCRIPTION

i.grey.scale is an interactive imagery function that assigns a histogram contrast stretch grey scale color table to a raster map layer. The histogram contrast stretch expands the original range of digital (category) values to utilize the full range of the user's graphics monitor.

The user is asked for the name of the raster map layer that needs a grey scale. When the raster map layer is displayed it will be displayed with a grey scale color scheme.

NOTES

This program is interactive and requires no command line arguments.

SEE ALSO

GRASS Tutorial: Image Processing

d.colormode, d.colors, d.colortable, i.colors, i.composite

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

i.group – An imagery function that creates and edits groups and subgroups of (raster) imagery files.
(GRASS Image Processing Program)

SYNOPSIS

i.group

DESCRIPTION

i.group allows the user to collect raster map layers in an imagery group by assigning them to user-named subgroups or other groups. This enables the user to run analyses on any combination of the raster map layers in a group. The user creates the groups and subgroups and selects the raster map layers that are to reside in them. Imagery analysis programs like *i.points*, *i.rectify*, *i.ortho.rectify* and others ask the user for the name of an imagery group whose data are to be analyzed. Imagery analysis programs like *i.cluster* and *i.maxlik* ask the user for the imagery group and imagery subgroup whose data are to be analyzed.

The first menu in the *i.group* program asks the user to select a *group*. If the group does not exist, the user will be asked if he or she would like to create a new group.

This program edits imagery groups. You may add raster map layers to, or remove such layers from, an imagery group. You may also create new groups.
Please enter the group to be created/modified

GROUP: _____ (enter 'list' for a list of groups)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO EXIT)

If the word *list* is entered, groups that have already been created in the user's current LOCATION_NAME and MAPSET(S) will be listed. The second menu in *i.group* provides the user with the following options:

1. Select a different group
2. Edit group title
3. Include new raster (cell) files in the group
or remove raster (cell) files from the group
4. Assign colors to the group
5. Create a new subgroup within the group

RETURN to exit

The options are described as follows:

Select a different group

If option number 1 is chosen, the following menu is displayed:

Please enter the group to be created/modified

GROUP: _____ (enter 'list' for a list of groups)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO EXIT)

If the word *list* is entered, groups that have already been created in the current LOCATION_NAME and MAPSET(S) will be displayed.

Edit group title

If option number 2 is selected, an entry space is provided to type in the group title. This title is useful in identifying each group:

TITLE_____

This option offers an opportunity to go back and change the entry if it is not correct by asking: **Look ok? (y/n)**.

Include new raster (cell) files in the group or remove raster (cell) files from the group

When choosing option number 3, the following menu is displayed:

LOCATION: *location* **GROUP:** *spot* **MAPSET:** *demo*

If you wish to delete a file from group [*spot***], remove the 'x' from in front of the file name.**

x_ spot.1 in demo
x_ spot.2 in demo
x_ spot.3 in demo

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR Ctrl-C> TO CANCEL)

Next,a menu listing all the other raster map layers present in the current MAPSET(S) will be displayed:

LOCATION: *location* **GROUP:** *spot* **MAPSET:** *demo*

Please mark an 'x' by the files to be added in group [*spot***]**

MAPSET: *demo*

x_ composite1
x_ spotclass1
__ spotclass2

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR Ctrl-C> TO CANCEL)

If more than one MAPSET is selected, menus for those mapsets will also be displayed. All raster map layers selected with an 'x' will be included in the group being updated.

The user will then have the opportunity to check the contents of the group that was just modified:

Group [*spot***] references the following raster files**

spot.1	**in demo**
spot.2	**in demo**
spot.3	**in demo**
composite1	**in demo**
spotclass1	**in demo**

Look ok? (y/n)

If the user responds with the letter *y* then the following sentence is displayed on the screen:

Group [*spot***] updated!**

And the main menu for *i.group* returns.

If the user responds *n*, the menu containing the group files after it was modified will be displayed and the user will be asked to place an *x* in front of those raster map layers that are to be removed from the group. Then, a menu listing all of the other raster map layers in the current MAPSET will be displayed again, and the user will be again asked to place an *x* in front of raster map layers to be included in the group. This gives the user the opportunity to correct mistakes or make changes in the choice of raster map layers to be selected in a group without exiting *i.group*.

Assign colors to the group

Option number 4 provides the following menu:

Please indicate which files to use for red, green, and blue colors. You may leave any color out. You may specify more than one color per file. However, each color may only be specified once. For example, to get a full color image, specify r,g,b for 3 different files. To get a grey scale image, specify rgb for a single file.

b__ spot.1
g__ spot.2
r__ spot.3
___ composite1
___ spotclass1

<<< r,g,b can only be specified once >>>

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

This menu allows you to select a color for each imagery band or for each file for display. Note, however, that composite images and classified images are already assigned colors during their creation.

An opportunity to change the choice of colors is offered after escaping the menu by:

Look ok? (y/n)

Create a new subgroup within the group

The following menu enables the user to create a subgroup out of any combination of raster map layers in the group. Any number of subgroups may be created by repeating the option.

LOCATION: *location* **MAPSET:** *spot*

GROUP: *spot1*
SUBGROUP: _____ ('list' will show available subgroups)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

After selecting or creating a subgroup, this menu is displayed:

Mark an 'x' by the files to form subgroup [*123*]

x_ spot.1
x_ spot.2
x_ spot.3
__ composite1
__ spotclass1

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

The user is then given the opportunity to check the contents of the subgroup:

Subgroup [*123*] references the following raster (cell) files

spot.1	**in demo**
spot.2	**in demo**
spot.3	**in demo**

Look ok? (y/n)

If the user responds with the letter ***n***, the group menu will appear again enabling the user to select raster map layers to form the subgroup.

NOTES

The ***i.group*** options are only available for imagery map layers in the current LOCATION_NAME.

Subgroup names may not contain more than 12 characters.

This program is interactive and requires no command line arguments.

SEE ALSO

GRASS Tutorial: Image Processing

i.cluster, i.maxlik, i.points, i.rectify

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

i.his.rgb − Hue-intensity-saturation (his) to red-green-blue (rgb) raster map color transformation function.
(GRASS Image Processing Program)

SYNOPSIS

i.his.rgb
i.his.rgb help
i.his.rgb hue_input=*name* **intensity_input**=*name* **saturation_input**=*name*
red_output=*name* **green_output**=*name* **blue_output**=*name*

DESCRIPTION

i.his.rgb is an image processing program that processes three input raster map layers as hue, intensity and saturation components and produces three output raster map layers representing the red, green and blue components of this data. The output raster map layers are created by a standard hue-intensity-saturation (his) to red-green-blue (rgb) color transformation. Each output raster map layer is given a linear gray scale color table. The current geographic region and mask settings are respected.

Parameters:

hue_input=*name*	Name of input raster map layer representing ***hue***.
intensity_input=*name*	Name of input raster map layer representing ***intensity***.
saturation_input=*name*	Name of input raster map layer representing ***saturation***.
red_output=*name*	Output raster map layer representing the ***red*** component in the data.
green_output=*name*	Output raster map layer representing the ***green*** component in the data.
blue_output=*name*	Output raster map layer representing the ***blue*** component in the data.

NOTES

It is not possible to process three bands with ***i.his.rgb*** and then exactly recover the original bands with ***i.rgb.his***. This is due to loss of precision because of integer computations and rounding. Tests have shown that more than 70% of the original cell values will be reproduced exactly after transformation in both directions and that 99% will be within plus or minus 1. A few cell values may differ significantly from their original values.

SEE ALSO

hsv.rgb.sh, i.colors, i.grey.scale, i.rgb.his, rgb.hsv.sh

AUTHOR

David Satnik, GIS Laboratory, Central Washington University

with acknowledgements to Ali Vali, Univ. of Texas Space Research Center, for the core routine.

NAME

i.ifft – Inverse Fast Fourier Transform (ifft) for image processing.
(GRASS Image Processing Program)

SYNOPSIS

i.ifft
i.ifft help
i.ifft real_image=*name* **imaginary_image**=*name* **output_image**=*name*

DESCRIPTION

i.ifft is an image processing program based on the algorithm given by Press (1988), and modified by Vali (1990), that converts real and imaginary frequency space images (produced by *i.fft*) into a normal image.

Parameters:

real_image=*name*	Input raster map layer for inversion fast Fourier transform, real part.
imaginary_image=*name*	Input raster map layer for inversion fast Fourier transform, imaginary.
output_image=*name*	Output inversion raster map layer after fast Fourier transformation.

The current mask is respected when reading the real and imaginary component files; thus, *r.mask* become a primary program for selecting the portion of the frequency space data to be included in the inverse transform. The GRASS program *d.digit* can be used to create masks while viewing the real or imaginary component image. When *i.ifft* is executed, it (automatically) uses the same GRASS region definition setting that was used during the original transformation done with *i.fft*.

The real and imaginary components are read from arrays of doubles in the *cell_misc* directory (produced by the forward transform program, *i.fft*), and the reconstructed image will preserve the cell value scaling of the original image processed by *i.fft*. No color table is assigned to the output map; one should be created before viewing the *output_image*.

SEE ALSO

Numerical Recipes in C, by William H. Press, Cambridge University Press, 1988.

Remote Sensing Digital Image Analysis, by John A. Richards, Springer-Verlag, 1986.

Personal communication, between program author and Ali R. Vali, Space Research Center, University of Texas, Austin, 1990.

i.cca, i.class, i.fft, i.pca

AUTHOR

David Satnik, GIS Laboratory, Central Washington University

NAME

i.in.erdas –Creates raster files from ERDAS files.
(GRASS Raster Program)

SYNOPSIS

i.in.erdas
i.in.erdas help
i.in.erdas [**–l**] **input=***name* **output=***name* **[trailer=***name*] **[bands=***value[,value,...*]] **[srow=***value*] **[scol=***value*] **[rows=***value*] **[cols=***value*]

DESCRIPTION

This program creates raster map files from ERDAS files. It creates one raster file for each selected band, up to a maximum of seven bands. GRASS color and category support files are created if an ERDAS trailer file is specified.

Only ERDAS version 7.4 or later files in 4-bit, 8-bit, or 16-bit formats are supported.
GRASS raster files will be named *prefix.band*

Note:

Remember that it is necessary to run: *r.support* to create the histogram or change the color table. *i.group* to associate the individual raster files as an image group.

COMMAND LINE OPTIONS

Flags:

-l List the ERDAS header only (no raster files created)

Parameters:

input ERDAS input file name

output output prefix of the GRASS raster files to be created.

trailer ERDAS trailer input file name

bands Selected bands to extract. (defaults to all bands)

srow Starting row. (defaults to first row in file)

scol Starting column. (defaults to first column in file)

rows Number of rows to extract. (defaults to all rows)

cols Number of columns to extract. (defaults to all columns)

SEE ALSO

r.support,
i.group

AUTHOR

M.L. Holko, USDA, SCS, NHQ-CGIS
Paul H. Fukuhara, USDA, SCS, NCG-GSS

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

i.maxlik − An imagery function that classifies the cell spectral reflectances in imagery data based on the spectral signature information generated by either ***i.cluster***, ***i.class***, or fIi.gensig.
(GRASS Image Processing Program)

SYNOPSIS

i.maxlik
i.maxlik help
i.maxlik [**-q**] **group**=*name* **subgroup**=*name* **sigfile**=*name* **class**=*name* [**reject**=*name*]

DESCRIPTION

i.maxlik is a maximum-likelihood discriminant analysis classifier. It can be used to perform the second step in either an unsupervised or a supervised image classification.

Either image classification methods are performed in two steps. The first step in an unsupervised image classification is performed by ***i.cluster***; the first step in a supervised classification is executed by the ***GRASS*** program ***i.class***. In both cases, the second step in the image classification procedure is performed by *i.maxlik*.

In an unsupervised classification, the maximum-likelihood classifier uses the cluster means and covariance matrices from the ***i.cluster*** signature file to determine to which category (spectral class) each cell in the image has the highest probability of belonging. In a supervised image classification, the maximum-likelihood classifier uses the region means and covariance matrices from the spectral signature file generated by ***i.class***, based on regions (groups of image pixels) chosen by the user, to determine to which category each cell in the image has the highest probability of belonging.

In either case, the raster map layer output by ***i.maxlik*** is a classified image in which each cell has been assigned to a spectral class (i.e., a category). The spectral classes (categories) can be related to specific land cover types on the ground.

The program will run non-interactively if the user specifies the names of raster map layers, i.e., group and subgroup names, seed signature file name, result classification file name, and any combination of non-required options in the command line, using the form

i.maxlik[**-q**] **group**=*name* **subgroup**=*name* **sigfile**=*name* **class**=*name* [**reject**=*name*]

where each flag and options have the meanings stated below.

Alternatively, the user can simply type ***i.maxlik*** in the command line without program arguments. In this case the user will be prompted for the program parameter settings; the program will run foreground.

OPTIONS

Flags:

-q Run quietly, without printing program messages to standard output.

Parameters:

group=*name* The imagery group contains the subgroup to be classified.

subgroup=*name* The subgroup contains image files, which were used to create the signature file in the program ***i.cluster***, ***i.class***, or ***i.gensig*** to be classified.

sigfile=*name* The name of the signatures to be used for the classification. The signature file contains the cluster and covariance matrices that were calculated by the ***GRASS*** program ***i.cluster*** (or the region means and covariance matrices generated by ***i.class***, if the user runs a supervised classification). These spectral signatures are what determine the categories (classes) to which image pixels will be assigned during the classification process.

class=*name* The name of a raster map holds the classification results. This new raster map layer

will contain categories that can be related to land cover categories on the ground.

reject=*name* The optional name of a raster map holds the reject threshold results. This is the result of a chi square test on each discriminant result at various threshold levels of confidence to determine at what confidence level each cell classified (categorized). It is the reject threshold map layer, and contains one calculated confidence level for each classified cell in the classified image. One of the possible uses for this map layer is as a mask, to identify cells in the classified image that have the lowest probability of being assigned to the correct class.

NOTES

The maximum-likelihood classifier assumes that the spectral signatures for each class (category) in each band file are normally distributed (i.e., Gaussian in nature). Algorithms, such as ***i.cluster, i.class,*** or ***i.gensig***, however, can create signatures that are not valid distributed (more likely with ***i.class).*** If this occurs, ***i.maxlik*** will reject them and display a warning message.

This program runs interactively if the user types ***i.maxlik*** only. If the user types ***i.maxlik*** along with all required options, it will overwrite the classified raster map without prompting if this map existed.

SEE ALSO

Grass Tutorial: Image Processing

i.class, i.cluster, i.gensig i.group, r.mask

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory
Tao Wen, University of Illinois at Urbana-Champaign, Illinois

NAME

i.ortho.photo - An interactive imagery function for the ortho-rectification of imagery group files.
(GRASS Image Processing Program)

SYNOPSIS

i.ortho.photo

DESCRIPTION

i.ortho.photo allows the user to ortho-rectify imagery group files. An imagery group consists of several scanned aerial photographs (raster files) of a common area. Imagery groups can be created or modified using the GRASS Image Processing Program *i.group*, or using the first menu option described below. *i.ortho.photo* guides the user through the steps required to ortho-rectify the raster files in a single imagery group.

The first menu in *i.ortho.photo* provides the user with the following options:

Initialization Options
1) Select/Modify imagery group
2) Select/Modify imagery group target
3) Select/Modify target elevation model
4) Select/Modify imagery group camera

Transformation Parameter Computation
5) Compute image-to-photo transformation parameters
6) Compute photo-to-target transformation parameters
7) Initialize exposure station parameters

Ortho-rectification Option
8) Ortho-rectify imagery group raster files

RETURN to exit
>

The options are described as follows:

Select/Modify imagery group

The current imagery group is display at the top of the previous menu. You may select another (new or existing) imagery group for the ortho-rectification program using option (1). After choosing option (1) you will be prompted for the name of a new or existing imagery group. Option (1) using the GRASS Image Processing Program *i.group*, for creation or modification of imagery groups. For more information on imagery group creation or modification please the the GRASS manual page for *i.group*.

Select/Modify imagery group target

The target location and mapset may be selected or modified using option (2). After choosing option (2) you will be prompted for the names of the target location and mapset, where the ortho-rectified raster files will reside. The target location is also the location from which the elevation model (raster file) will be selected -- see option (3). Option (2) uses the GRASS Image Processing Program *i.target*, for selection or modification of the imagery group target location and mapset. For more information on imagery group selection or modification please the the GRASS manual page for *i.target*.

Select/Modify imagery group

Option (3) allows you to select the raster file from the target location to be used as the elevation model. The elevation model is required for both the computation of photo-to-target parameters (option 6) and for the ortho-rectification of the imagery group files (option 8). The raster file select for the elevation model should cover the entire area of the image group to be ortho-rectified. Currently, the elevation model raster file is expected to be in units of meters. DTED and DEM files are suitable for use as the elevation model in the ortho-rectification program. After selection option (3) you will be prompted for the name of the raster file in the target location that you want to use as the elevation model.

Select/Modify imagery group camera

Using option (4) you may select or create an camera reference file that will be used with the current imagery group. A camera reference file contains information on the internal characteristics of the aerial camera, as well as the geometry of the fiducial or reseau marks. The most important characteristic of the camera is its focal length. Fiducial or reseau marks locations are required to compute the scanned image to photo-coordinate transformation parameter (option 5). For a more detailed description of option (4) please see the GRASS manual page for *photo.camera*.

Compute image-to-photo transformation parameters

The scanned image to photo-coordinate transformation parameters are computed using option (5). In this interactive option you associate scanned reference points (fiducials, reseau marks, etc.) with their known photo coordinates from the camera reference file. Complete documentation for this option is available under the manual entry *photo.2image*.

Compute photo-to-target transformation parameters

The photo to target transformation parameters are compute using option(6). Here control points are marked on one or more imagery group files and associated with there known UTM and elevation coordinates. Complete documentation for this option is available under the manual entry *photo.2target*.

Select initial exposure station parameters

If option (7) is selected, initial camera exposure station parameters and initial variances may be selected or modified. Complete documentation for this option is available under the manual entry *photo.init*.

Ortho-photo imagery group files

Option (8) is used to perform the actual image ortho-rectification after all of the transformation parameters have been computed. Ortho-rectified raster files will be created in the target location for each selected imagery group file. You may select either the current window in the target location or the minimal bounding window for the ortho-rectified image. Complete documentation for this option is available under the manual entry *photo.rectify*.

NOTES

i.ortho.photo currently requires the elevation model to be in meters, and the target location to be a UTM coordinate system.

SEE ALSO

photo.camera[2], *photo.2image[2]*, *photo.2target[2]*, *photo.init[2]*, *photo.rectify[2]*

AUTHOR

Mike Baba DBA Systems, Inc.

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

i.pca – Principal components analysis (pca) program for image processing.
(GRASS Image Processing Program)

SYNOPSIS

i.pca
i.pca help
i.pca input=*name,name*[*,name,name,...*] **output**=*name* **rescale**=*min,max*

DESCRIPTION

i.pca is an image processing program based on the algorithm provided by Vali (1990), that processes n (2 >= n) input raster map layers and produces n output raster map layers containing the principal components of the input data in decreasing order of variance ("contrast"). The output raster map layers are assigned names with .1, .2, ...n suffixes. The current geographic region definition and mask settings are respected when reading the input raster map layers. When the rescale option is used, the output files are rescaled to fit the min,max range.

Parameters:

input=*name,name*[*,name,name,...*]
Name of two or more input raster map layers.

output=*name* The output raster map layer name to which suffixes are added. Each output raster map layer is assigned this user-specified ***name*** with a numerical (.1, .2, ...n) suffix.

rescale=*min,max* The optional output category range. (Default: 1,255) If rescale=0,0, no rescaling is performed on output files.

Richards (l986) gives a good example of the application of principal components analysis (pca) to a time series of LANDSAT images of a burned region in Australia.

SEE ALSO

Richards, John A., **Remote Sensing Digital Image Analysis**, Springer-Verlag, 1986.

Vali, Ali R., Personal communication, Space Research Center, University of Texas, Austin, 1990.

i.cca, i.class, i.fft, i.ifft, m.eigensystem, r.covar, r.mapcalc

AUTHOR

David Satnik, GIS Laboratory, Central Washington University
Major modifications for GRASS 4.1 were made by Olga Waupotitsch and Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

i.points − An imagery function that enables the user to mark coordinate system points on an image to be rectified and then input the coordinates of each point for creation of a coordinate transformation matrix. The transformation matrix is needed as input for the GRASS program ***i.rectify***.
(GRASS Image Processing Program)

SYNOPSIS

i.points

DESCRIPTION

i.points is an imagery function that enables the user to mark points on a (raster) image to be rectified and then input the geographic coordinates of each point for calculation of a coordinate transformation matrix. ***i.points*** must be followed by use of the GRASS program ***i.rectify***, which rectifies the image using the transformation matrix coefficients calculated by ***i.points***.

Rectification is the mapping (transformation) of an image from one coordinate system to another. The geometry of an image extracted into a GRASS LOCATION having an x,y coordinate system is not planimetric. To create a planimetric image, that is, to convert the x,y coordinate system into a standard coordinate system (for example, the UTM coordinate system or the State Plane coordinate system), points from a map having the standard coordinates must be associated with the same points on the image to be rectified. ***i.points*** enables the user to mark points on an image and input the standard coordinates for that point. ***i.points*** then calculates a least squares regression using the two coordinate systems (x,y and standard) for the marked points. A matrix containing transformation coefficients is the output file for ***i.points***.

During the process of marking points and entering map coordinates, the user can compute the RMS (root mean square) error for each point entered. ***i.points*** does this by calculating the transformation equation (the same one that is calculated in the GRASS program ***i.rectify***), and then plugging these results into an equation for RMS error.

i.points offers a zoom option to locate precisely the point to be marked on an image. This program also offers the user the option of acquiring standard coordinates for a marked point from a map layer in the target data base.

i.target must be run before running ***i.points*** to enable the PLOT RASTER option to be used and to identify a target data base LOCATION_NAME and MAPSET for the rectified image. To run ***i.points***, a graphics monitor is required.

The procedure for marking points, entering coordinates, and calculating RMS error is described below.

The first prompt in the program asks the user for the imagery group to be registered. Note that if ***i.target*** is not run before ***i.points***, the ***i.points*** program will display the following error message:

ERROR: Target information for group [*spot*] missing
Please run *i.target* for group [*spot*]

After entering the group to be registered the terminal screen displays the message:

Use mouse now...

The graphics monitor displays the following screen:

imagery filename (mag)	*target filename (mag)*
QUIT ZOOM PLOT RASTER ANALYZE	

A pop-down menu like that shown below will be superimposed on the left half of the screen:

Double click on raster map layer to be plotted Double click here to cancel

Mapset *demo*	
spotclass	**spot.1**
composite	**spot.2**
spot.3	

Any single raster map layer in the imagery group may be used on which to mark points, and the user can mark points on more than one raster map layer in the imagery group to accumulate the suggested minimum number of 12 points. Any raster map layer in the imagery group can be rectified (using ***i.rectify***) based on the transformation matrix computed from these points.

The imagery file chosen by the user is displayed in the upper left quadrant of the screen.

ZOOM

To magnify the displayed file, the user must place the mouse cross hairs on the word ZOOM. The following menu will then be displayed at the bottom of the screen:

Cancel	**Box**	**Point**	**Select type of ZOOM**

The user has the option of identifying the zoom region either by using the mouse to make a box, or by using the mouse to mark the two diagonal points of the desired region. The terminal screen will display a mouse

button menu to guide the user in identifying the corner points of the region.

MARKING POINTS

To mark the points on the image that correspond to the points on a standard coordinate system map, the user must place the mouse cross hairs on the corresponding location on the image to be marked and press the left hand button on the mouse. A diamond shaped symbol will be marked on the image. The user's terminal will display the following menu:

Point 1 marked on the image at **East: 1023.77** **North: -164.41**	
Enter coordinates as east north:	

The user then enters the easting and northing (separated by a space) for the point marked on the image. If the user wishes not to enter a coordinate, he or she may simply hit RETURN to return control to the mouse; the marked point then disappears.

PLOT RASTER

In addition to acquiring reference points from a standard map, the user has the option of acquiring the reference points from a raster map layer in the target data base LOCATION_NAME. The data base raster map layer is displayed by placing the mouse cross hairs on the words PLOT RASTER. The following line is then displayed at the bottom of the graphics monitor:

Cancel	**Indicate which side should be plotted**

Which side of the graphics monitor is to be plotted is indicated by placing the mouse cross hairs on the half of the graphics monitor screen that the user would like to use, and pressing the left mouse button. The following pop-down menu will be superimposed on the half of the screen that was chosen:

Double click on raster (cell) map layer to be plotted **Double click here to cancel**

Mapset *demo*	
tm.rectified	
tm.classified	
Mapset PERMANENT	
elevation	**geology**
slope	**soils**
aspect	
roads	
streams	
airfields	

After the raster map layer is displayed the following message appears at the bottom of the graphics monitor:

input method -->	**keyboard**	**screen**

If the user wishes to use the plotted raster map layer only as a comparative reference, then the keyboard can be chosen as the means to input coordinates corresponding to the marked points on the image. This is done by placing the mouse cross hairs on the word KEYBOARD and pressing the left button on the mouse.

If the user selects the SCREEN option, then points marked on the image will automatically be associated with the coordinates from the corresponding points on the target data base map layer. In this option, when the user marks a point on the image, the following menu is displayed at the terminal:

Point 5 marked on the image at **East: 1023.77** **North: -164.41** **Point located at** **East: 679132.57** **North: 4351080.67**	
use mouse now...	

The user then uses the mouse to mark a corresponding point on the displayed image from the target data base. The coordinates for the target data base map layer are automatically saved as the coordinates corresponding to the marked point on the image.

ANALYZE

After a number of points have been marked (4 to 7), the user can check the RMS error of the points marked on the image. This is done by placing the mouse cross hairs on the word ANALYZE at the bottom of the graphics monitor. An error report resembling that shown below is superimposed on the graphics monitor:

		error		image	target		
#	**row**	**col**	**target**	**east**	**north**	**east**	**north**
1	**0.0**	**-0.9**	**1.0**	**1048.5**	**-144.8**	**679132.5**	**4351080.6**
2	**0.4**	**1.0**	**1.3**	**2153.1**	**-567.2**	**684314.7**	**4399001.4**
3	**-1.2**	**-0.5**	**.6**	**1452.8**	**-476.5**	**567841.4**	**3457682.8**
4	**1.1**	**0.5**	**1.3**	**1034.0**	**-109.2**	**677573.8**	**4352626.4**
5	*-2.7*	*14.0*	*14.2*	*1048.6*	*-144.9*	*679132.6*	*4351080.7*
	overall	**rms**	**error:**	**4.46**			

The following menu then appears at the bottom of the graphics monitor:

DONE	**PRINT FILE**	**Double click on point to be included/excluded**

The RMS error for the image is given under the column titled "error" and subtitled "row" and "col". In the above report, point number 1 is 0.0 rows and -0.9 columns from the predicted location calculated from the transformation equation. The RMS error for the target raster map layer is listed under the heading "target". This is the RMS error for the east and north coordinates of the target map layer, but it is presented in the table using one general value. The overall RMS error is displayed at the bottom of the screen in meters. Points that create high RMS error are displayed in red on the graphics monitor (represented here in italics).

The location of the point marked on the imagery group file is given under the heading "image" and the subheadings "east" and "north". The location of the point in the target data base is given under the heading "target" and the subheadings "east" and "north". If the user would like to exclude or include a point, this can be accomplished by placing the mouse cross hairs on the point number to be included (if the point is absent) or excluded (if the point is displayed) and pressing the left button on the mouse twice. When a point is excluded, it is not afterwards included in the calculation of the RMS error, or included in the final transformation matrix. However, it can be retrieved within *i.points* at any time by double clicking with the mouse as described above.

QUIT

To end the *i.points* program place the mouse cross hairs on the word QUIT; the marked points (including coordinates) will be saved.

NOTES

A good rule of thumb is to mark at least 12 to 15 points which are evenly distributed over the entire imagery group file in order to obtain an accurate transformation equation for the rectification process. The RMS error may increase with more points added, but the transformation equation will be more accurate.

An RMS error of less than or equal to approximately one resolution unit (pixel or cell) for the image being rectified is generally considered acceptable.

In order to use a digitizer with *i.points*, at least one digitizer driver besides "none" (the on-screen digitizer) must be available in the digitcap file.

This program is interactive and requires no command line arguments.

SEE ALSO

GRASS Tutorial: Image Processing

g.mapsets, i.group, i.rectify, i.target

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

i.quantize − An interactive imagery function that creates a raster map layer whose color table is based on the red, green, and blue color values present in existing, user-specified imagery group files.
(GRASS Image Processing Program)

SYNOPSIS

i.quantize

DESCRIPTION

i.quantize is an interactive imagery function that allows the user to create a new raster map layer whose color table values are based on the red, green, and blue color values present in (as many as) three raster map layers in an imagery group.

The user is first asked to enter the name of the imagery group from whose map layers red, green, and blue color values are to be extracted. The user is then shown a listing of imagery files in the specified group, and is asked to indicate which files in the group are to be used for red, green, and blue colors in the new map layer. Each color (red, green, and blue) must be specified once and only once.

The user is also asked to assign a name to the new raster map layer output.

NOTES

This program is interactive and requires no command line arguments.
The input files must ALL be within the range of 0-255.

SEE ALSO

GRASS Tutorial: Image Processing

d.colormode, d.colors, d.colortable, hsv.rgb.sh, i.colors, i.composite, i.grey.scale, i.group, i.his.rgb, i.rgb.his, r.mapcalc, rgb.hsb.sh, r.colors

AUTHOR

David Gerdes, U.S. Army Construction Engineering Research Laboratory
Ref. Paul Heckbert "Image Quantization" Siggraph Proceedings 1982

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

i.rectify − An imagery function that rectifies an image by computing a coordinate transformation for each cell (pixel) in the image using the transformation coefficient matrix created by the GRASS program *i.points*.
(GRASS Image Processing Program)

SYNOPSIS

i.rectify

DESCRIPTION

i.rectify rectifies an image by using the transformation coefficient matrix created by *i.points*. Rectification is the process by which the geometry of an image is made planimetric. This is accomplished by mapping (transforming) an image from one coordinate system to another. In *i.rectify*, the coefficients computed by *i.points* are used in an equation to convert x,y coordinates to standard map coordinates for each cell in the image. The result is an image with a standard map coordinate system. Upon completion of the program the rectified image is deposited in a previously targeted GRASS LOCATION_NAME and MAPSET.

The first prompt in the program asks the user for the name of the group containing the raster map layers to be rectified.

The user is then asked to select the map layer(s) within the group to be rectified:

Please select the file(s) to rectify by naming an output file

spot.1 in demo	**.....**
spot.2 in demo	**.....**
spot.3 in demo	**.....**
spotclass in demo	*spotrectify***..**
spotreject in demo	**.....**

(Enter list by any name to get a list of existing raster files)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

More than one raster map layer may be rectified at a time. Each raster map layer should have a unique output file name.

Next the user is asked to select one of two geographic region settings:

Please select one of the following options
1. Use the current region in the target location
2. Determine the smallest region which covers the image

i.rectify will only rectify that portion of the image that occurs within the chosen geographic region setting. Only that portion will be relocated in the target database. It is therefore important to check the current geographic region settings in the target location if choice number one is selected.

NOTES

i.rectify uses nearest neighbor resampling during the transformation.

i.rectify uses a linear affine transformation:
x' = ax + by +c
y' = Ax + Bt +C

The a,b,c,A,B,C are determined by least squares regression based on the control points entered. This transformation applies scaling, translation and rotation. It is NOT a general purpose rubber-sheeting, nor is it ortho-photo rectification using a DEM, not second order polynomial, etc. It can be used if (1) you have geometrically correct images, and (2) the terrain or camera distortion effect can be ignored.

Use *i.ortho.photo* for photos for which you have camera information and a DEM, or *i.rectify2* if you want a 2nd or 3rd order polynomial transformation.

i.rectify will run in the background and notify the user by mail when it is finished. The process may take an hour or more depending on the size of the image, the number of files, and the size of the geographic region definition.

The rectified (raster) image will be located in the target LOCATION when the program is completed. The original unrectified raster map layers are not modified or removed.

This program is interactive and requires no command line arguments.

SEE ALSO

GRASS Tutorial: Image Processing

i.group, i.points, i.target

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

i.rectify2 –

The i.rectify2 routine is the imagery function that rectifies an image by computing a coordinate transformation for each pixel in the image based on the control points created by the GRASS program i.points or i.vpoints. The i.rectify2 program uses the control points identified in i.points or i.vpoints to calculate a transformation matrix based on a first, second , or third order polynomial and then converts x,y cell coordinates to standard map coordinates for each pixel in the image. The result is a planimetric image with a transformed coordinate system (i.e., a different coordinate system than before it was rectified).

The i.vpoints or i.points program must be run before i.rectify2, and both programs are required to rectify an image. An image must be rectified before it can reside in a standard coordinate LOCATION, and therefore be analyzed with the other map layers in the standard coordinate LOCATION. Upon completion of i.rectify2, the rectified image is deposited in the target standard coordinate LOCATION. This LOCATION is selected using the i.target program of GRASS.

(GRASS Image Processing Program)

SYNOPSIS

i.rectify2

DESCRIPTION

Program Prompts

The first prompt in the program asks for the name of the group containing the files to be rectified.

Enter the group containing files to be rectified
Enter 'list' for a list of existing imagery groups
Enter 'list -f' for a verbose listing
Hit RETURN to cancel request
>

This is the same imagery group that was selected in i.points or i.vpoints and the group that contains the cell files with the marked points and their associated map coordinates. You are then asked to select the cell file(s) within the group to be rectified:

Please select the file(s) to rectify by naming an output file

spot1.1 in mapsetname .....
spot1.2 in mapsetname .....
spot1.3 in mapsetname .....
spotclass1 in mapsetname spotrectify1..

spotreject1 in mapsetname .....

(enter list by any name to get a list of existing cell files)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE (OR<Ctrl-C> TO CANCEL

More than one cell file may be rectified at a time. Each cell file should be given a unique output file name.

Next, you are asked to select one of two windows regions:

Please select one of the following options

1. Use the current window in the target location
2. Determine the smallest window which covers the image

>

The i.rectify2 program will only rectify that portion of the image or cell file that occurs within the chosen window region, and only that portion of the cell file will be relocated in the target database. It is important therefore, to check the current mapset window in the target LOCATION if choice number one is selected.

If you are rectifying a file with plans to patch it to another file using the GRASS program patch[1] or patch[2], choose option number one, the current window in the target location. This window, however, must be the default window for the target LOCATION. When a file being rectified is smaller than the default window in which it is being rectified, zeros are added to the rectified file. Patching files of the same size that contain 0/non-zero data, eliminates the possibility of a no-data line the patched result. This is because, when the images are patched, the zeros in the image are "covered" with non-zero pixel values. When rectifying files that are going to be patched, rectify all of the files using the same default window.

By selecting the 1st ORDER option, the user may select the order of transformation desired:

Select order of transformation --> 1st Order 2nd Order 3rd Order

The program will immediately recalculate the RMSE and the number of points required.

Polynomial Transformation Matrix

The ANALYZE function has been changed to support calculating the registration coefficients using a first, second, or third order transformation matrix. The number of control points required for a selected order of transformation (represented by þnþ) is ((n + 1) * (n + 2) / 2) or 3, 6, and 10 respectively. It is strongly recommended that one or more additional points be identified to allow for an overly- determined transformation calculation which will generate the Root Mean Square (RMS) error values for each included point. The RMS error values for all the included control points are immediately recalculated when the user selects a different transformation order from the menu bar. The polynomial equations in the new i.rectify2 program are performed using a modified Gaussian elimination method instead of the Cramerþs rule method used in the i.rectify routine.

Program Execution

i.rectify2 will run in the background and notify you by mail when it is finished.

Note: The rectified image or rectified cell files will be located in the target LOCATION when the program is completed. The original unrectified files are not modified or removed.

This program is interactive and requires no command line arguments.

SEE ALSO

GRASS Tutorial: Image Processing

i.group, i.points, i.vpoints, i.rectify, i.target

AUTHOR

William R. Enslin, Michigan State University Center for Remote Sensing

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of

GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

i.rgb.his – Red-green-blue (rgb) to hue-intensity-saturation (his) function for image processing.
(GRASS Image Processing Program)

SYNOPSIS

i.rgb.his
i.rgb.his help
i.rgb.his red_input=*name* **green_input**=*name* **blue_input**=*name*
hue_output=*name* **intensity_output**=*name* **saturation_output**=*name*

DESCRIPTION

i.rgb.his is an image processing program that processes three input raster map layers as red, green, and blue components and produces three output raster map layers representing the hue, intensity, and saturation of the data. The output raster map layers are created by a standard red-green-blue (rgb) to hue-intensity-saturation (his) color transformation. Each output raster map layer is given a linear gray scale color table. The current geographic region definition and mask settings are respected.

Parameters:

red_input=*name*	Input raster map layer representing the red component.
green_input=*name*	Input raster map layer representing the green component.
blue_input=*name*	Input raster map layer representing the blue component.
hue_output=*name*	Output raster map layer representing hue.
intensity_output=*name*	Output raster map layer representing intensity.
saturation_output=*name*	Output raster map layer representing saturation.

SEE ALSO

hsv.rgb.sh, i.his.rgb, rgb.hsv.sh

AUTHOR

David Satnik, GIS Laboratory, Central Washington University,

with acknowledgements to Ali Vali, Univ. of Texas Space Research Center, for the core routine.

NAME

i.smap − An imagery function that performs contextual image classification using sequential maximum a posteriori (SMAP) estimation.
(GRASS Imagery Program)

SYNOPSIS

i.smap
i.smap help
i.smap [**−mq**] **group=***name* **subgroup=***name* **signaturefile=***name* **[blocksize=***value]* **output=***name*

DESCRIPTION

The *i.smap* program is used to segment multispectral images using a spectral class model known as a Gaussian mixture distribution. Since Gaussian mixture distributions include conventional multivariate Gaussian distributions, this program may also be used to segment multispectral images based on simple spectral mean and covariance parameters.

i.smap has two modes of operation. The first mode is the sequential maximum a posteriori (SMAP) mode [1,2]. The SMAP segmentation algorithm attempts to improve segmentation accuracy by segmenting the image into regions rather than segmenting each pixel separately (see NOTES).

The second mode is the more conventional maximum likelihood (ML) classification which classifies each pixel separately, but requires somewhat less computation. This mode is selected with the -m flag (see below).

COMMAND LINE OPTIONS

Flags

-m Use maximum likelihood estimation (instead of smap). Normal operation is to use SMAP estimation (see NOTES).

-q Run quietly, without printing messages about program progress. Without this flag, messages will be printed (to stderr) as the program progresses.

Parameters

group imagery group

The imagery group that defines the image to be classified.

subgroup
imagery subgroup

The subgroup within the group specified that specifies the subset of the band files that are to be used as image data to be classified.

signaturefile
imagery signaturefile

The signature file that contains the spectral signatures (i.e., the statistics) for the classes to be identified in the image. This signature file is produced by the program *i.gensigset* (see NOTES).

blocksize
size of submatrix to process at one time
default: 128

This option specifies the size of the "window" to be used when reading the image data.

This program was written to be nice about memory usage without influencing the resultant

classification. This option allows the user to control how much memory is used. More memory may mean faster (or slower) operation depending on how much real memory your machine has and how much virtual memory the program uses.

The size of the submatrix used in segmenting the image has a principle function of controlling memory usage; however, it also can have a subtle effect on the quality of the segmentation in the smap mode. The smoothing parameters for the smap segmentation are estimated separately for each submatrix. Therefore, if the image has regions with qualitatively different behavior, (e.g., natural woodlands and man-made agricultural fields) it may be useful to use a submatrix small enough so that different smoothing parameters may be used for each distinctive region of the image.

The submatrix size has no effect on the performance of the ML segmentation method.

output output raster map.

The name of a raster map that will contain the classification results. This new raster map layer will contain categories that can be related to landcover categories on the ground.

INTERACTIVE MODE

If none of the arguments are specified on the command line, ***i.smap*** will interactively prompt for the names of the maps and files.

NOTES

The SMAP algorithm exploits the fact that nearby pixels in an image are likely to have the same class. It works by segmenting the image at various scales or resolutions and using the course scale segmentations to guide the finer scale segmentations. In addition to reducing the number of misclassifications, the SMAP algorithm generally produces segmentations with larger connected regions of a fixed class which may be useful in some applications.

The amount of smoothing that is performed in the segmentation is dependent of the behavior of the data in the image. If the data suggests that the nearby pixels often change class, then the algorithm will adaptively reduce the amount of smoothing. This ensures that excessively large regions are not formed.

REFERENCES

[1] C. Bouman and M. Shapiro," "Multispectral Image Segmentation using a Multiscale Image Model," ***Proc. of IEEE Int'l Conf. on Acoust., Speech and Sig. Proc.,*** pp. III-565 - III-568, San Francisco, California, March 23-26, 1992.

[2] C. Bouman and M. Shapiro, "A Multiscale Random Field Model for Bayesian Image Segmentation," submitted to the ***IEEE Trans. on Image Processing.***

SEE ALSO

i.group for creating groups and subgroups

i.gensigset to generate the signature file required by this program

AUTHOR

Charles Bouman, School of Electrical Engineering, Purdue University
Michael Shapiro, U.S. Construction Engineering Research Laboratory

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

i.tape.mss − An imagery function that extracts Multispectral Scanner (MSS) imagery data from half-inch tape.
(GRASS Image Processing Program)

SYNOPSIS

i.tape.mss

DESCRIPTION

i.tape.mss is a program that extracts Multispectral Scanner (MSS) imagery data from tape.

This program must be run in a LOCATION_NAME with a x,y coordinate system (i.e., a coordinate system with projection 0). For further information regarding this LOCATION_NAME refer to the manual entry for *imagery*.

The first prompt in *i.tape.mss* asks the user for the tape device name. This is sometimes /dev/rmt0 (for a half-inch tape with a tape density of 1600), but this varies with each machine.

The next prompt is:

Please mount and load tape, then hit RETURN -->

IMAGE IDENTIFICATION MENU

The first menu in the program asks the user for information about the data.

please enter the following information

Tape Identification: _____

Image Description: _____

Title for the Extracted Raster (Cell) Files: _____

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

This program automatically enters the scene ID number and the date of the image into the field for Tape Identification. The sun angles are automatically entered into the field for Image Description.

The second menu is:

MSS TAPE EXTRACTION
please select the desired tape window (geographic region definition) to extract

first row: _____(1-2984)
last row: _____(1-2984)

first col: _____(1-3548)
last col: _____(1-3548)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

The numbers in parentheses are the total number of rows and columns on the tape including filler (zeros). This information and additional information can also be obtained by running the GRASS program *i.tape.mss.h* which reads the header information on an MSS tape. Any subset of the image on tape may be extracted. For a discussion of row and column extraction see the subheading titled ROW AND COLUMN EXTRACTION below.

The next menu is:

please make an x by the bands you want extracted

_____ 1
_____ 2
_____ 3
_____ 4

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

MSS imagery has 4 bands, but the user may want to extract only a subset of these bands. See the subheading in this entry titled ROW AND COLUMN EXTRACTION.

The user then is asked to enter the prefix/group for the band files to be created. This name will precede each band file extracted into GRASS. For example, if three bands are extracted the following three (raster) band files will result:

prefixname.1
prefixname.2
prefixname.3

Whatever *prefixname* is specified will also automatically become the name for the imagery group file being created. Each image (i.e., each run of ***i.tape.mss***) should be given a unique prefix/group name.

The extraction process will begin by first skipping the number of specified files, advancing to the starting row, and then reading the tape. The percent completion of the extraction is displayed on the screen. If more than one tape is required to store the image, the program will pause and inform the user to mount the next

tape.

The extracted (raster) band files will be listed as raster map layers available in the current MAPSET and may be displayed using the GRASS commands *d.display*, *d.rast* or *i.points*.

NOTES

After extracting an image from tape, the geographic region definition in the x,y coordinate LOCATION_NAME will be set based upon the extracted rows and columns from the tape. The relationship between the image rows and columns and the geographic coordinates of the region is discussed in the manual entry for *imagery*.

This program is interactive and requires no command line arguments.

ROW AND COLUMN EXTRACTION

The display options in GRASS allow the user to locate rows and columns on the digital image. If enough disk space is available, one band of an entire image, or one band of a portion of an image known to contain the area of interest, can be extracted and displayed. The *measurements* option in *d.display*, or *d.where* (following the use of *d.rast*) will echo x and y coordinates to the screen. (These coordinates will display negative numbers in the north-south direction, but ignoring the negative sign will yield the row number.) See the *imagery* manual entry for further explanation.

If a photograph of the digital image is available, the rows and columns to be extracted can be determined from it by associating inches with the total number of known rows and columns in the scene. For example, if the total length of the photograph is 12 inches, the total number of rows on the tape is 2000, and the northwest corner of the area of interest begins 2 inches from the top of the photo, then:

$$12" / 2000 \text{ rows} = 2" / x \text{ rows}$$
$$x = 333.333$$

The northwest corner of the area of interest starts at row 333. The starting row, ending row, starting column, and ending column can be calculated in this manner.

SEE ALSO

GRASS Tutorial: Image Processing

d.display, d.rast, d.where, i.group, i.points, i.tape.mss.h, i.tape.other, i.tape.tm, imagery

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

i.tape.mss.h − An imagery function that extracts header information from LANDSAT Multispectral Scanner (MSS) imagery data stored on half-inch tape.
(GRASS Image Processing Program)

SYNOPSIS

i.tape.mss.h
i.tape.mss.h help
i.tape.mss.h *tape_drive_name*

DESCRIPTION

i.tape.mss.h reads the header information on a Multispectral Scanner (MSS) tape. This program reads the specified input file (the computer-compatible tape *tape_drive_name*), and by default displays the output to the user's terminal. The user may redirect output to a file by using the UNIX redirection mechanism. For example:

i.tape.mss.h /dev/rmt0 > *h.out*

The name of the tape drive depends on the computer being used.

This program can be run either non-interactively or interactively. The user can run the program by specifying program arguments on the command line. Alternately, the user can simply type **i.tape.mss.h** on the command line, without program arguments. In this event, the program will prompt the user to enter a tape device name using the standard user interface described in the manual entry for *parser*.

SEE ALSO

GRASS Tutorial: Image Processing

i.tape.mss and *parser*

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

i.tape.other − An imagery function that extracts scanned aerial imagery (NHAP, etc.) and satellite imagery (TM, SPOT, etc) from half-inch or 8mm tape.
(GRASS Image Processing Program)

SYNOPSIS

i.tape.other

DESCRIPTION

i.tape.other is a generic program that extracts imagery from tape using the tape description that is input by the user.

This program must be run in a LOCATION_NAME with a x,y coordinate system (i.e., a coordinate system with projection 0). For further information regarding this LOCATION_NAME refer to the manual entry for *imagery*.

The first prompt in *i.tape.other* asks the user for the tape device name. This is sometimes /dev/rmt0 (for a density of 1600), but this varies with each machine.

The next prompt is:

Please mount and load tape, then hit RETURN -->

IMAGE IDENTIFICATION MENU

The first menu in the program asks the user for information about the data.

please enter the following information

Tape Identification: _____

Image Description: _____

Title for the Extracted Raster (Cell) Files: _____

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

TAPE LAYOUT MENU

The next menu asks for the physical layout of the tape.

GENERIC TAPE EXTRACTION

tape layout
0__ number of tape files to be skipped
0__ number of records in the remaining files to be skipped
band files
0__ number of bands on the tape
data format
___ band sequential (BSQ) | mark one with an x
___ band interleaved (BIL) |
0__ if you select BSQ format and all the bands are in a single file, enter the total number of records in the file. Otherwise enter 0

0__ length (in bytes) of the longest record on the tape
1__ blocking factor of data in the file

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE (OR <Ctrl-C> TO CANCEL)

number of tape files to be skipped

If there are files at the beginning of the tape which are not image data, they can be skipped. Sometimes information that comes with a tape will indicate the number of header files or records on the tape. The GRASS utility *m.examine.tape* will also provide this information. The *record length* is the number of columns in the image, while the *number of records* is the number of rows in the image (not always correct, see *blocking factor of data in the file* below). NHAP imagery and usually most scanned aerial imagery do not have tape header files, but this should be checked. TM imagery has one header file that contains imagery format of data files and parameters of data aquisition. SPOT imagery has two files that should be skipped on the first tape, and one file to be skipped on the second tape (of a two-tape set).

number of records in the remaining files to be skipped

If the files which contain the image begin with non-image data, these records can also be skipped. This is usually 0 for most data types. SPOT imagery stored in 1600bpi has one header record in the image file on each tape that should be skipped.

number of bands on the tape

Most aerial imagery have 3 bands, but satellite simulator data may have more. TM data has seven bands and SPOT has three bands as a standard, respectively. The total number of bands on the tape should be specified here, not just the number that will be extracted.

data format

The two formats that imagery data are most commonly stored in are called *band interleaved* format (BIL) and *band sequential* format (BSQ). In BIL format, each record on the tape contains one line for one band of data. If the data contains three bands, then the first five records will look like this:

band 1, line 1
band 2, line 1
band 3, line 1
band 1, line 2
band 2, line 2

In BSQ format, all lines of one band are stored together on a tape, followed by all lines of another band, followed by all lines of the next band, etc. These data are stored as if they were in a one

band BIL format:

band 1, line 1
band 1, line 2
band 1, line 3
.
.
.
band 2, line 1
band 2, line 2
.
.
band 2, line 156
band 2, line 157

Each pixel contains one byte and there is one line per record. BSQ format is the format that is usually created by optical scanning devices when they scan photographs, but not all digitized aerial imagery are stored in this format. The format of the data is usually written on the exterior of the tape; this should be checked.

length (in bytes) of the longest record on the tape

This must be set to the number of bytes in the longest data record. It is used to determine how large a buffer to use for reading the tape. This value can be obtained using *m.examine.tape.*

blocking factor of data in the file

This is the number of lines combining into one physical record on tape. It is usually one for most of imageries, i.e., one line per physical record. However, considering on data compressing and tape memory saving, sometimes more than one lines are combined into one physical record on the tape. This number may be written on the exterior of the tape, otherwise the user need to experiment on this number by running of *i.tape.other*.

BAND EXTRACTION MENU

The user is then asked to mark an *x* beside the bands to be extracted. See the subheading in this entry entitled ROW AND COLUMN EXTRACTION.

please mark an x by the bands you want extracted

_____1
_____2
_____3
_____4

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

PREFIX/GROUP NAME

The user is asked to enter the prefix/group for the (raster) band files to be created. This name will precede each band file extracted into GRASS. For example, if three bands are extracted the following three band files will result:

prefixname.1
prefixname.2
prefixname.3

The specified *prefixname* will also automatically become the name for the imagery group file being created. Each image (i.e., each run of *i.tape.other)* should be given a unique prefix/group name.

ROW AND COLUMN MENU

Finally, the starting row, ending row, starting column and ending column are required. This allows the user to extract any subset of the image from the tape.

EXTRACT
please select desired tape window (geographic region definition) to extract

start row:_0_
end row:_0_

start col:_0_
end col:_0_

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

The extraction process will begin by first skipping the number of specified files, advancing to the starting row, and then reading the tape. The percent completion of the extraction is displayed on the screen.

Following the extraction, the extracted band files will be listed as raster map layers available in the current MAPSET. These raster map layers may be displayed individually using the GRASS commands *d.display*, *d.rast* or *i.points*.

NOTES

This program can be used for extraction of TM, SPOT and other types of data from tape; however, the user must supply information to the program on how the image data is laid out on the tape. For example, the image data may be padded with surrounding extra rows and/or columns; the user may wish to skip over these rows and columns and extract only the actual image data from the tape. *i.tape.other* does not know where image data actually begins on the tape; the user must tell the program what portion of the tape data to extract. Often, information on the orientation and layout of the image data on tape will be included on a printout accompanying any tape data received by the user; however, this may not always be the case. The user may need to experiment with various runs of *i.tape.other* before extracting the portions of an image actually desired.

After extracting an image from tape, the geographic region in the x,y coordinate LOCATION_NAME will be set, based upon the extracted rows and columns from the tape. The relationship between the image rows and columns and the coordiates bounding the geographic region is discussed in the *imagery* manual entry.

This program is interactive and requires no command line arguments.

ROW AND COLUMN EXTRACTION

The display options in GRASS allow the user to locate rows and columns on the digital image. If enough disk space is available, one band of an entire image, or one band of a portion of an image known to contain the area of interest, can be extracted and displayed. The *measurements* option in *d.display*, or *d.where* (following a run of *d.rast*) will echo x and y coordinates to the screen. (These coordinates will display negative numbers in the north-south direction but ignoring the negative sign will yield the row number. See the *imagery* manual entry for further explanation.)

If a photograph of the digital image is available, the rows and columns to be extracted can be determined

from it by associating inches with the total number of known rows and columns in the scene. For example, if the total length of the photograph is 12 inches, the total number of rows on the tape is 2000, and the northwest corner of the area of interest begins 2 inches from the top of the photo, then:

12" / 2000 rows = 2" / x rows
x = 333.333

The northwest corner of the area of interest starts at row 333. The starting row, ending row, starting column, and ending column can be calculated in this manner.

SEE ALSO

GRASS Tutorial: Image Processing

d.display, d.rast, d.where, i.group, i.points, i.tape.mss, i.tape.mss.h, i.tape.tm, i.tape.tm.fast, imagery, m.examine.tape

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory
Tao Wen, University of Illinois at Urbana-Champaign, Illinois

NAME

i.tape.spot – An imagery function that extracts SPOT imagery from half-inch tape.
(GRASS Image Processing Program)

SYNOPSIS

i.tape.spot

DESCRIPTION

i.tape.spot is a program that extracts SPOT imagery from 9-track, half-inch tape.

This program must be run in a LOCATION_NAME with an x,y coordinate system (i.e., a coordinate system with projection 0). For further information regarding this LOCATION_NAME type refer to the *imagery* manual entry.

The first prompt in *i.tape.spot* asks the user for the tape device name.

Enter tape device name:

This is sometimes /dev/rmt0 (for a half-inch tape having a density of 1600 bpi), but this varies with each machine.

The next prompt is:

Mount SPOT tape and hit RETURN -->

IMAGE IDENTIFICATION MENU

The first menu in the program asks the user for information about the data.

Please enter the following information

TAPE IDENTIFICATION: _____

IMAGE DESCRIPTION: _____

TITLE FOR THE EXTRACTED CELL (RASTER) FILES: _____

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

This program automatically reads the satellite name, tape product code, instrument name, interleaving indicator, spectral mode, preprocessing level and work order number into the field for TAPE IDENTIFICATION. The mission, path, row, scene shift, scene center date and time, orientation, incidence, azimuth, elevation angle, and absolute calibration coefficients and offsets are automatically entered into the field for IMAGE DESCRIPTION. User can type in any other messages into the two sections and a description as the title of raster map layer. The second menu is:

SPOT IMAGE EXTRACT
Please select region of the image to extract

start row: 0_____(1-3002)
end row: 0_____(1-3002)

start col: 0_____(1-3166)
end col: 0_____(1-3166)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

The numbers in parentheses are the total number of rows and columns on the tape including zeros (filler). This information and additional information can also be obtained by running the program *m.examine.tape*. *m.examine.tape* will read any tape and provide the user with the number of files on a tape, the number of records on a tape, and the record lengths. Any subset of the image on the tape may be extracted. For a discussion of row and column extraction see the subheading entitled ROW AND COLUMN EXTRACTION below.

The next menu is:

Please mark an x by the bands you want extracted

_____ 1
_____ 2
_____ 3

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

SPOT imagery has three bands, but the user may want to extract a subset of these bands. See the subheading in this entry entitled ROW AND COLUMN EXTRACTION.

The user then is asked to enter the prefix/group name for the raster band files to be created. This name will precede each band file extracted into GRASS. For example, if two bands are extracted the following two band files will result:

prefixname.1
prefixname.2

The specified *prefixname* will also automatically become the name for the imagery group file being created. Each image or subset (i.e., each run of *i.tape.spot*) should be given a unique prefix/group name.

The extraction process will begin by first skipping a number of files which are not data or not requested, advancing to the first band requested, forwarding to the requested column, and then reading the data. After extracting the requested rows and columns from each band, the program creates support files for the raster band map layer. The percent completion of the extraction is displayed on the screen. Because sometimes SPOT imagery is very large and is stored in multiple tape sets, the program is designed to read image by pausing when the tape need to be changed and inform the user to mount and load next tape. The number of tapes required to store one scene depends on the number of bytes per inch (bpi) in which the data are stored.

The extracted band files will be listed as raster map layers available in the current MAPSET and may be displayed using the GRASS commands *d.display*, *d.rast*, or *i.points*.

ROW AND COLUMN EXTRACTION

The display options in GRASS allow the user to locate rows and columns on the digital image. If enough disk space is available, one band of an entire image or one band of a portion of an image known to contain the area of interest, can be extracted and displayed. The *measurements* option in *d.display*, or *d.where* (following use of *d.rast*) will echo x and y coordinates to the screen. (These coordinates will display negative numbers in the north-south direction, but ignoring the negative sign will yield the row number.) See the *imagery* manual entry for further explanation.

Each scene of a SPOT image contains filler on both the left and right sides of the quad. The user may want to identify the row and column numbers in order to exclude the filler. This filler will otherwise be extracted with the image and take up unnecessary disk space.

If a photograph of the digital image is available, the rows and columns to be extracted can be determined from it by associating inches with the total number of known rows and columns in the scene. For example, if the total length of the photograph is 12 inches, the total number of rows on the tape is 2000, and the northwest corner of the area of interest begins 2 inches from the top of the photo, then:

$$12" / 2000 \text{ rows} = 2" / x \text{ rows}$$
$$x = 333.333$$

The northwest corner of the area of interest starts at row 333. The starting row, ending row, starting column, and ending column can be calculated in this manner.

NOTES

After extracting an image from tape the geographic region definition in the x,y coordinate LOCATION_NAME will be set based upon the extracted rows and columns from the tape. The relationship between the image rows and columns and the coordinates of the geographic region is discussed in the *imagery* format manual entry.

This program is interactive and requires no command line arguments.

ROW AND COLUMN EXTRACTION

SEE ALSO

GRASS Tutorial: Image Processing

d.display, d.rast, d.where, i.group, i.points, i.tape.mss, i.tape.mss.h, i.tape.other, i.tape.tm, imagery, m.examine.tape

AUTHOR

Tao Wen, University of Illinois at Urbana-Champaign, Illinois

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

i.tape.tm − An imagery function that extracts LANDSAT Thematic Mapper (TM) imagery from half-inch tape.
(GRASS Image Processing Program)

SYNOPSIS

i.tape.tm

DESCRIPTION

i.tape.tm is a program that extracts LANDSAT Thematic Mapper (TM) imagery from half-inch tape.

This program must be run in a LOCATION_NAME with a x,y coordinate system (i.e., a coordinate system with projection 0). For further information regarding this LOCATION_NAME refer to the *imagery* manual entry.

The first prompt in *i.tape.tm* asks the user for the tape device name. This is sometimes /dev/rmt0 (for a half-inch tape having a density of 1600 bpi), but this varies with each machine.

The next prompt is:

Please mount and load tape, then hit RETURN -->

IMAGE IDENTIFICATION MENU

The first menu in the program asks the user for information about the data.

please enter the following information

Tape Identification: _____

Image Description: _____

Title for the Extracted Raster (Cell) Files: _____

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

This program automatically enters the scene ID number into the field for Tape Identification. The mission, path, row, quadrant, date, and whether the image is corrected is automatically entered into the field for Image Description.

The second menu is:

THEMATIC MAPPER EXTRACT
please select the desired tape window (geographic region definition) to extract

first row: _____(1-2984)
last row: _____(1-2984)

first col: _____(1-4220)
last col: _____(1-4220)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

The numbers in parentheses are the total number of rows and columns on the tape including zeros (filler). This information and additional information can also be obtained by running the program *m.examine.tape*. *m.examine.tape* will read any tape and provide the user with the number of files on a tape, the number of records on a tape, and the record lengthes. Any subset of the image on the tape may be extracted. For a discussion of row and column extraction see the subheading entitled ROW AND COLUMN EXTRACTION below.

The next menu is:

please make an x by the bands you want extracted

_____ 1
_____ 2
_____ 3
_____ 4
_____ 5
_____ 6
_____ 7

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

TM imagery has 7 bands, but the user may want to extract only a subset of these bands. See the subheading in this entry entitled ROW AND COLUMN EXTRACTION.

The user then is asked to enter the prefix/group for the raster band files to be created. This name will precede each band file extracted into GRASS. For example, if three bands are extracted the following three band files will result:

prefixname.1
prefixname.2
prefixname.3

The specified *prefixname* will also automatically become the name for the imagery group file being created. Each image or quad (i.e., each run of ***i.tape.tm***) should be given a unique prefix/group name.

The extraction process will begin by first skipping the number of specified files, advancing to the first band requested, and then reading the tape. After extracting the requested rows and columns for each band, the program creates support files for the raster band map layer. The percent completion of the extraction is

displayed on the screen. Because TM imagery is divided into four quads and is stored in multiple tape sets, the program is designed to read one quad at a time. The number of tapes required to store one quad depends on the number of bytes per inch in which the data is stored. If more than one tape is required to store one quad, the program will pause and inform the user to mount the next tape.

The extracted band files will be listed as raster map layers available in the current MAPSET and may be displayed using the GRASS commands ***d.display***, ***d.rast*** or ***i.points***.

NOTES

After extracting an image from tape the geographic region definition in the x,y coordinate LOCATION_NAME will be set based upon the extracted rows and columns from the tape. The relationship between the image rows and columns and the coordinates of the geographic region is discussed in the ***imagery*** manual entry.

This program is interactive and requires no command line arguments.

ROW AND COLUMN EXTRACTION

The display options in GRASS allow the user to locate rows and columns on the digital image. If enough disk space is available, one band of an entire image or, one band of a portion of an image known to contain the area of interest, can be extracted and displayed. The ***measurements*** option in ***d.display***, or ***d.where*** (following use of ***d.rast***) will echo x and y coordinates to the screen. (These coordinates will display negative numbers in the north-south direction, but ignoring the negative sign will yield the row number.) See the ***imagery*** manual entry for further explanation.

Each quad of a TM image contains filler on both the left and right sides of the quad. The user may want to identify the row and column numbers in order to exclude the filler. This filler will otherwise be extracted with the image and take up unnecessary disk space.

If a photograph of the digital image is available, the rows and columns to be extracted can be determined from it by associating inches with the total number of known rows and columns in the scene. For example, if the total length of the photograph is 12 inches, the total number of rows on the tape is 2000, and the northwest corner of the area of interest begins 2 inches from the top of the photo, then:

$$12" / 2000 \text{ rows} = 2" / x \text{ rows}$$
$$x = 333.333$$

The northwest corner of the area of interest starts at row 333. The starting row, ending row, starting column, and ending column can be calculated in this manner.

SEE ALSO

GRASS Tutorial: Image Processing

d.display, d.rast, d.where, i.group, i.points, i.tape.mss, i.tape.mss.h, i.tape.other, imagery, m.examine.tape

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

i.tape.tm.fast – An imagery function that extracts Thematic Mapper (TM) imagery from tape media *(GRASS Image Processing Program)*

SYNOPSIS

i.tape.tm.fast
i.tape.tm.fast help
i.tape.tm.fast [**-q**] **input**=*name* **group**=*name* **bands**=*value*[,*value*,...] [**rows**=*firstrow-lastrow*] [**cols**=*firstcol-lastcol*] [**title**=*name*]

DESCRIPTION

i.tape.tm.fast is a program that extracts TM imagery from tape media with different blocking factors (its value indicates how many rows are combined into one physical record on the tape).

i.tape.tm.fast must be run in a LOCATION_NAME with a (x,y) coordinate system (i.e., a coordinate system with projection 0). For further information regarding the LOCATION_NAME type, please refer to the *imagery* manual entry.

i.tape.tm.fast reads the blocking factor from the header file as well as other parameters, such as gains and offsets for each band, map projection, sun elevation and azimuth, etc., and writes into history file (depending upon the contents of the header file).

OPTIONS

This program can be run either non-interactively or interactively. It will be run non-interactively if the user specifies the name of input device, the name of output group file, bands to be extracted, and optionally other parameters (see below) on the command line using the form:

> **i.tape.tm.fast** [**-q**] **input**=*name* **group**=*name* **bands**=*value*[,*value*,...] [**rows**=*firstrow-lastrow*] [**cols**=*firstcol-lastcol*] [**title**=*name*]

where the ***input*** should be the device name on which the tape media are mounted. The ***group*** is an imagery group which will store the extracted TM imagery. The ***bands*** value in a list separated by commas is the bands the user wants to extract from the imagery. The ***rows*** and ***cols*** represent the region which the user wishes to extract, where the default is whole imagery. At last an optional ***title*** is for information only. Alternatively, the program will be run interactively if the user types only ***i.tape.tm.fast***; in this case the program will prompt the user for parameter values using the standard ***GRASS*** parser interface described in the manual entry for ***parser***.

Flags:

-q Run quietly. Supresses output of program percent-complete messages. If this flag is not used, these messages are printed out.

Parameters:

input=*name* The name of the device on which the tape media containing the imagery files mounted.

group=*name* The name of the group, which will store the imagery extracted from tape media.

bands=*value*[,*value*,...]
The bands the user wishes to extracted from the tape media.
Options: 1-7

rows=*firstrow-lastrow*
The values of first and last row of the extracting region.
Default: full imagery

cols=*firstcol-lastcol* The values of first and last column of the extracting region.

Default: full imagery

title=*name* The title of the extracting imagery.
Default: TM Imagery File Extracted from Tape

NOTES

Running in command line mode, ***i.tape.tm.fast*** will overwrite the group file and support files without prompting if the files existed.

SEE ALSO

GRASS Tutorial: Image Processing

i.group, i.tape.mss, i.tape.mss.h, i.tape.other, i.tape.tm, i.tape.spot, imagery m.examine.tape

AUTHOR

Tao Wen, University of Illinois at Urbana-Champaign, Illinois

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

i.target − An interactive imagery function that establishes a GRASS target location and mapset for an imagery group.
(GRASS Image Processing Program)

SYNOPSIS

i.target

DESCRIPTION

i.target targets an imagery group to a GRASS data base location name and mapset. During the imagery program *i.rectify*, a location name and mapset are required into which to transfer the rectified file just prior to completion of the program; *i.target* enables the user to specify this location. *i.target* must be run before *i.points* and *i.rectify*.

The first prompt in the program asks the user for the name of the imagery group that needs a target. The imagery group must be present in the user's current mapset.

The following menu asking for the target LOCATION NAME and MAPSET is displayed:

Please select the target LOCATION and MAPSET for group <*spot*>

CURRENT LOCATION: *location*_____
CURRENT MAPSET: *demo*_____

TARGET LOCATION:_____
TARGET MAPSET:_____

(Enter list for a list of location names or mapsets within a location)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL

An imagery group may be targeted to any GRASS location.

NOTES

This program is interactive and requires no command line arguments.

SEE ALSO

Grass Tutorial: Image Processing

i.group, i.points, i.rectify

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

i.vpoints − Identifies coordinate pairs of points from a vector map or keyboard entry and corresponding points in an image.

DESCRIPTION

This program enables the user to identify coordinate pairs of points from a vector map or keyboard entry and corresponding points in an image to be rectified. The map coordinate values of each point are used to calculate a transformation matrix. The operator may then use the *i.rectify* or *i.rectify2* program to rectify the image using the transformation matrix coefficients calculated from the control point file created in *i.vpoints*. The *i.rectify* program performs only a first order transformation of the image whereas *i.rectify2* can rectify an image using either a first, second, or third order polynomial equation.

The first step is to display the unrectified image and corresponding vector map data. The operator would then mark corresponding control point locations on the image and map. To identify the precise location of a point to be marked, i.vpoints has a zoom option. In addition to marking control points on an image to be rectified and inputting their world coordinate values using the keyboard, i.vpoints has the option to simultaneously display vector map data available in the targeted database, and identify on the vector map the location of the corresponding marked points. When this option is chosen, the coordinate values are input automatically. Any GRASS map layer or vector map in the targeted database LOCATION can be displayed using i.vpoints. The i.vpoints program also has the capability of overlaying (i.e., warping) the vector data onto the raster image to visually check the accuracy of the registration based on the current set of active control points. During the process of marking points and entering map coordinates, the user can compute the RMS (root mean square) error for each point entered. The i.vpoints program does this by calculating a transformation equation (the same one that is calculated in the GRASS program i.rectify2). Coefficients are computed for the equation. The coefficients are then used in the equation along with the x,y coordinates of the marked points. The results are plugged into an equation for RMS error. The interpretation of RMS error is described in the ANALYZE subsection.

The procedures for marking control points (registration points), displaying vector map layers, overlaying vector maps onto the raster image, and calculating RMS error are described in the following sections .

To enter the program (the i.vpoints program requires the use of a graphics monitor) type i.vpoints . . .

```
GRASS - GRID > i.vpoints
```

the first prompt in the program asks for the imagery group to be registered . . .

```
Enter imagery group to be registered
Enter 'list' for a list of existing imagery groups
Enter 'list -f' for a verbose listing
Hit RETURN to cancel request
>
```

for example, >list might produce the following response

```
<list>
Available groups
- - - - - - - - - - - - - - - - - - - - - - - - - -
test
- - - - - - - - - - - - - - - - - - - - - - - - - -
hit RETURN to continue -->
```

whereas, >list -f might produce the following response

```
<list -f>
Available groups
- - - - - - - - - - - - - - - - - - - - - - - - - -
test
    photo in PERMANENT
    res2 in tifftest
- - - - - - - - - - - - - - - - - - - - - - - - - -
hit RETURN to continue -->
```

The imagery group entered above should contain the files that you wish to rectify. After entering the group to be registered, the terminal screen displays the message:

```
>test
<test>
Use mouse now . . .
```

And the color graphics monitor displays the following screen:

Any single file in the imagery group may be used to mark points, and points can be marked on more than one file in the imagery group to accumulate the suggested minimum number of points (3 for a 1st order transformation, 6 for a 2nd order, and 10 for a 3rd order). Any file in the imagery group can be subsequently rectified (using i.rectify2) based on the transformation matrix computed from these points.

The chosen file is displayed in the upper left quadrant of the monitor at a default magnification based on the extent of the current active window.

RASTER IMAGE

The raster image option on the menu at the bottom of the window allows the user to display any single file in the imagery group in the upper right quadrant of the window screen. The option provides the same file selection pick list as is presented when you first enter the i.vpoints program. When you select this option, the program will erase the data contained in all of the four quadrant windows and will reinitialize all program values.

VECTOR MAPS

The vector maps option on the menu at the bottom of the screen allows the user to display vector map data in the upper right quadrant of the screen. After selecting the vector map layer to display, a menu selection bar appears along the bottom on the screen. This pick list is used to select the line color (blue, gray, green, red, white, or yellow) for the selected vector data layer.

Refresh

The refresh option on the main menu allows the user to "refresh" or re-draw the displayed vector data. This function will erase all outlines showing the limits of previously zoomed areas. A "yes/no" roompt will appear:

```
Refresh Map ?      NO   YES
```

Zoom

To enlarge a raster or vector image, place the mouse cross hairs on the word zoom on the main menu

and press the left button. The following menu will be displayed at the bottom. of the screen:

CANCEL BOX POINT Select type of zoom

You have the option to identify the map extent of the zoom window using either the mouse to define a box, or the mouse to mark a center point from which to enlarge the image. The box option first prompts you to identify a starting corner for the zoom region and then allows you to define the area to be zoomed using a rubber band box. The prompts appear as follows...

CANCEL Mark the first corner of region

CANCEL Define the region

After marking the first corner of the region to be enlarged, hold down the left button and move the mouse to change the size and shape of the rubber band box. After defining the area to be enlarged, press the right button to accept it.

The point method for enlarging an image will display a mouse menu to guide you in selecting the appropriate enlargement. To enlarge or reduce the magnification factor, place the cursor on the "+" or "-" box and press the left button on the mouse.

You may zoom either the raster or the vector display.

Upon accepting the new region limits, the raster or vector data are redisplayed in either the lower left (raster) or lower right (vector) windows.

The extent of the zoomed area is outlined on the unzoomed image in the main window area.

While the main menu is displayed, you can mark corrosponding control points on the raster and vector images or enter map coordinates from the keyboard. If you are using coordinates taken from a reference map, circle these points and then use whatever means you have available to identify as precisely as possible the coordinate values for these points. Digitizing software is recommended, especially GRASS 3,0 program digit/1/. Once you have determined the standard coordinates (for example, UTM's) of each circled point, you are ready to mark the points on the displayed image.

To mark the points on the image, that correspond to the points on the standard coordinate map, place the mouse cross hairs on the point on the image to be marked (you will probably have to ZOOM to find the exact spot) and press the left hand button on the mouse. A diamond shaped symbol will be marked on the image. The text monitor will display the following screen:

Point 1 marked on the image at
East: 1023.77
North: -164.41

Enter coordinates as east north:

Analyze

After a number of points have been marked (a minimum of 4 for a 1st order transformation, 7 for a 2nd order, and 11 for a 3rd order), the RMS error of the points marked on the image can be checked. This is

done by placing the cross hairs on the word ANALYZE on the main menu at the bottom of the monitor. The following error report is superimposed on the monitor:

```
    error                 image                 target
#   col  row  target       east      north        east
north

1   -0.9 0.0  1.0      1048.5       -144.8 679132.5      4351080.6
2   1.0  0.4  1.3      2153.1       -567.2 684314.7      4399001.4
.
.
.

    Overall rms error  76.85
```

The RMS error for the image being rectified is recorded under the column "error" and subtitled "row' and "col". In the above report, the marked point number 1 is 0.0 rows and -0.9 columns from the predicted location calculated by the transformation equation.

The RMS error for the target database map is recorded under the heading "error" and the subheading "target". This is the RMS error for the east and the north coordinate values of the target map, but it is represented in the table using one general value.

The overall RMS error for the image is displayed at the bottom of the screen in meters. Points that generate a high RMS error are displayed in red on the monitor.

The x,y coordinate values of the point marked on the image being rectified are recorded under the heading "image" and the subheadings "east" and "north".

The standard coordinate values of the point in the target database are recorded under the heading "target" and the subheadings "east" and "north".

If the user would like to exclude or include a point, this can be accomplished by placing the mouse cross hairs on the point number to be included (if the point is absent) or excluded (if the point is displayed) and then pressing the left button on the mouse twice. When a point is excluded, it is not included in the calculation of the RMS error, or included in the final transformation matrix. However, it can be retrieved within i.vpoints at any time by double clicking with the mouse as described above.

The following menu appears at the bottom of the monitor:

DONE PRINT FILE OVERLAY DELETE ON Transformation - -> 1st ORDER Double click on point to be DELETED

Selecting DELETE ON will toggle the option to DELETE OFF, the toggle option is used to allow the user to physically remove a control point from the POINTS file instead of just flagging it as an non-active reference point.

Overlay

Overlay allows the user to overlay the vector map(s) onto the raster image. Overlay can be used to warp (register) and display the selected vector file data on top of the raster image contained in the upper left window of the color screen. An inverse coordinate transformation is performed using the currently active order of transformation (i.e., first, second, or third).

Overlay vectors on raster image NO YES

By selecting the 1st ORDER option, the user may select the order of transformation desired:

Select order of transformation --> 1st Order 2nd Order 3rd Order

The program will immediately recalculate the RMSE and the number of points required.

Quit

To exit the i.vpoints program, place the mouse cross hairs on the word QUIT at the bottom of the monitor and all of the marked points (including coordinates) will be saved.

SEE ALSO

GRASS Tutorial: Image Processing

g.mapsets, i.group, i.points, i.rectify, i.rectify2, i.target

AUTHOR

William R. Enslin, Michigan State University Center for Remote Sensing

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

i.zc − Zero-crossing "edge detection" raster function for image processing.
(GRASS Image Processing Program)

SYNOPSIS

i.zc
i.zc help
i.zc input_map=*name* **zc_map**=*name* [**width**=*value*] [**threshold**=*value*] [**orientations**=*value*]

DESCRIPTION

i.zc is an image processing program used for edge detection. The raster map produced shows the location of "boundaries" on the input map. Boundaries tend to be found in regions of changing cell values and tend to run perpendicular to the direction of the slope. The algorithm used for edge detection is one of the "zero-crossing" algorithms and is discussed briefly below.

This program will be run interactively if the user types **i.zc** without program arguments on the command line. In this event, the program will prompt the user for parameter values using the standard interface described in the manual entry for *parser*. Alternately, the user can run the program non-interactively by specifying program parameter values on the command line.

Parameters:

input_map=*name* Name of input raster map layer.

zc_map=*name* Name of raster map layer to be used for zero-crossing values.

width=*value* This parameter determines the x-y extent of the Gaussian filter. The default value is 9; higher and lower values can be tested by the user. Increasing the width will result in finding "edges" representing more gradual changes in cell values.
Default: 9

threshold=*value* This parameter determines the "sensitivity" of the Gaussian filter. The default value is 10; higher and lower values can be tested by the user. Increasing the threshold value will result in fewer edges being found.
Default: 10

orientations=*value* This value is the number of azimuth directions the cells on the output raster map layer are categorized into (similar to the aspect raster map layer produced by the *r.slope.aspect* program). For example, a value of 16 would result in detected edges being categorized into one of 16 bins depending on the direction of the edge at that point.
Default: 1

The current region definition and mask settings are respected when reading the input map.

NOTES

The procedure to find the "edges" in the image is as follows: 1) The Fourier transform of the image is taken, 2) The Fourier transform of the Laplacian of a two-dimensional Gaussian function is used to filter the transformed image, 3) The result is run through an inverse Fourier transform, 4) The resulting image is traversed in search of places where the image changes from positive to negative or from negative to positive, 5) Each cell in the map where the value crosses zero (with a change in value greater than the threshold value) is marked as an edge and an orientation is assigned to it. The resulting raster map layer is output.

SEE ALSO

Personal communication between program author and Ali R. Vali, Space Research Center, University of Texas, Austin, 1990.

i.fft, i.ifft, r.mapcalc, r.mfilter, r.slope.aspect and *parser*

AUTHOR

David Satnik, GIS Laboratory, Central Washington University

NAME

m.datum.shift − Datum shift program.
(GRASS Data Import/Processing Program)

SYNOPSIS

m.datum.shift lat=*dd.mm.ss{n| s}* lon=*dd.mm.ss{e| w}* is=*input_spheroid* os=*output_spheroid* dx=*xshift* dy=*yshift* dz=*zshift*

DESCRIPTION

m.datum.shift returns geographic coordinates based on a different spheroid (and datum) than the one used to obtain the original coordinates.

The input and output spheroids, *is* and *os*, are the spheroids for two different datums. The input spheroid is the one on which the original coordinates are based. The output spheroid is that on which the resultant coordinates will be based. The "shifting" occurs between the two datums. The shift values, *dx*, *dy*, and *dz*, are constants. They indicate the mean differences between points in the second datum versus the first as measured in meters.

The list of spheroids available is somewhat dynamic. It may not contain exactly the ones listed below. To determine the current list of possible spheroids, type in the command:

m.datum.shift lat=0n lon=0w dx=0 dy=0 dz=0 is=help os=help

A list of available spheroids will be printed on the screen. If the spheroid desired is not on the list, the values for the semi-major axis and the eccentricity squared for the spheroid may be entered in place of a spheroid name in the following format:

s=a=*semi-major_axis*,**e=***eccentricity_squared*

SOME POSSIBLE SPHEROIDS
(*The on-line listing includes only the spheroid names*)

Spheroid	Commonly used for:	Semi-major axis	Eccentricity sqrd
australian	Australia	6378160.0	0.0066945419
bessel	Japan	6377739.155	0.0066743722
clark66	N. America	6378206.4	0.006768658
clark80	France, Africa	6378249.145	0.0068035113
everest	India, Burma	6377276.345	0.0066378466
international	Europe	6378388.0	0.00672267
wgs72	worldwide coverage	6378135.0	0.006694317778

EXAMPLE

m.datum.shift lat=0n lon=175w is=clark66 os=wgs72 dx=-22 dy=157 dz=176

Results:
lat=0.00.05.72999N
lon=174.59.55.004133W

NOTES

Essentially, the program follows these steps. The original point, as defined by a latitude and a longitude, is converted to geocentric coordinates. The shift values are added to the geocentric coordinates. The summed values are then converted to latitude and longitude based on the output spheroid.

For a brief discussion of spheroids and datums see *m.ll2u*. For a brief discussion of geocentric coordinates see *m.ll2gc*.

This remains under testing is still an experimental program. It is part of an initial effort to incorporate geographic coordinates into GRASS.

SEE ALSO

m.gc2ll, m.ll2gc, m.ll2u, m.u2ll

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

m.dem.examine − Provides a terse description of USGS Digital Elevation Model (DEM) data files stored on 1/2-inch magnetic tape.
(GRASS Data Import/Processing Program)

SYNOPSIS

m.dem.examine input=*name* **blocksize**=*value*

DESCRIPTION

m.dem.examine reads Digital Elevation Model (DEM) terrain elevation data produced and supplied by the USGS from the input file specified by ***name***, in the block size specified by ***value***. If no input file is named on the command line, the program assumes that /dev/rmt0 is the input file. The information provided to the user comes from each file included on tape. Included are the maximum and minimum elevation values and the approximate UTM boundaries of each file. The program assumes unlabeled tapes in ASCII format with no header or trailer files.

SEE ALSO

g.region, m.dem.extract, m.examine.tape, r.rescale, r.support

AUTHOR

Andrew Heekin, U.S. Army Construction Engineering Research Laboratory

NAME

m.dem.extract – Extracts USGS Digital Elevation Model (DEM) data from 1/2-inch magnetic tape.
(GRASS Data Import/Processing Program)

SYNOPSIS

m.dem.extract
m.dem.extract help
m.dem.extract input=*name* **output**=*name* **blocksize**=*value* **start**=*value* **end**=*value*

DESCRIPTION

m.dem.extract extracts USGS Digital Elevation Model (DEM) elevation data that fits into the user's current geographic region from the input file *input*, in blocks of *blocksize* bytes. If no *input* file is specified by the user, input is taken from /dev/rmt0, by default. Results are placed in the named *output* file, and stored beneath the *cell* directory of the user's current mapset. *m.dem.extract* will only extract data that fall within the boundaries of the user's current geographic region. Data falling outside this region will be ignored. *m.dem.extract* will not complain if the input file does not cover the entire geographic region. The user should run *m.dem.extract* without specifying output file (which only examines the input file) prior to extracting the data to determine the size of the geographic region needed. If the block size is unknown, run the command *m.examine.tape*.

The user can run this program either non-interactively or interactively. The program will be run non-interactively if the user specifies program arguments on the command line, using the form:

m.dem.extract input=*name* **[output**=*name*] **[blocksize**=*value*] **[start**=*value*] **[end**=*value*]

Alternately, the user can simply type **m.dem.extract** on the command line, without program arguments. In this case, the user will be prompted for needed parameter values using the standard GRASS interface described in the manual entry for *parser*.

Parameters:

input=*name*	The full path name of tape device or file from which DEM data are to be extracted. Default: /dev/rmt0
output=*name*	The name to be assigned to the output file containing raster DEM data extracted from 1/2-inch tape.
blocksize=*value*	The physical block size (record length) of each each record, in bytes. *m.examine.tape* can be used to determine block size.
start=*value*	First DEM quad to look at (default 1).
end=*value*	Last DEM quad to look at (default 9999999).

NOTES

The user should check the boundaries and resolution of the current region setting (see *g.region*) **BEFORE** extracting data, since *m.dem.extract* will only extract the data that falls within these boundaries and only use the set resolution during extraction.

Warning: This program assumes that the tape has **ONLY** DEM data. If the tape contains other files (DLG, for example) the program will not skip over them. In this case, forward the tape to the first DEM file. Use the appropriate UNIX commands (**mt** to forward the tapes past the non-DEM files, and **dd** to extract and copy the correct files) on the tape on another machine first to extract the non-DEM data from the tape.

USGS Tapes: This version of *m.dem.extract* is sensitive to formatting errors. If the format on a tape is wrong, the program exits with the error message: "The file with incorrect data format encountered ..." Nevertheless, the program continues. In this case, run the program without specifying input files. If the program exits again with the same error message, there is an inconsistency in the data format.

The format error: Each USGS Dem file consists of A and B, and sometimes C, records--

type A record	header
type B record	collection of profiles with various numbers of elevations
type C record	accuracy

This new version of ***m.dem.extract*** cannot skip over any records to the end of a DEM file. It first determines how many records there are in a file, reads those records, and then proceeds to the header (record A) of the next file. Thus it is very important for the program to know whether or not there is a C record to be read at the end of each file. This information is given in the file's header record (type A) which has a field set to 1 when a C-record exits, and to 0 when a C-record does not exist.

In some DEM files, the accuracy field is set to 0, even though a C-record does exist. Because of this error, when the program tries to read the data, it will not try to read the C-record, but instead attempts to go on to the next file and read its header--a type A record. This results in the format error because type A records and type C records have different formats, and the program is trying to read a type A record when the tape is presenting a type C record.

When this format error in a file is encountered, forward the tape to the next file.

SEE ALSO

g.region, m.examine.tape, r.rescale, r.support and ***parser***

AUTHOR

Andrew Heekin, U.S. Army Construction Engineering Research Laboratory

Improvements to program code were made for GRASS 4.0 by
David Satnik, Central Washington University

Major enhancements for GRASS 4.1 to allow the input to be a file or tape were made by
Olga Waupotitsch, U.S. Army Construction Engineering Research Laboratory

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

m.dmaUSGSread − Extracts digital terrain elevation data (DTED) produced by the Defense Mapping Agency (DMA) but supplied by the USGS (in a different tape format) on 1/2-inch magnetic tape.
(GRASS Data Import/Processing Program)

SYNOPSIS

dd [**if**=*tapedev*] **ibs**=*input-block-size* | **m.dmaUSGSread** **top**=*value* **bottom**=*value* **left**=*value* **right**=*value* **output**=*name* **logfile**=*name*

DESCRIPTION

m.dmaUSGSread extracts digital terrain elevation data (DTED) that was produced by the Defense Mapping Agency (DMA) but bought from the USGS; these two agencies distribute the same data in a slightly different tape formats. This program requires the use of the UNIX command *dd* to read the 1/2-inch magnetic tape. *m.dmaUSGSread* should be used prior to using the GRASS programs *m.rot90* and *r.in.ll*.

Parameters for the UNIX *dd* command are listed below.

if=*tapedev* The path name of the tape drive from which input will be taken (usually /dev/rmt0).

ibs=*input-block-size* The physical blocking value of the data on tape, usually written on the tape spool. If this value is unknown, run *m.examine.tape* prior to running *m.dmaUSGSread*.

Parameters for *m.dmaUSGSread* are listed below.

top=*value* Beginning row number of data.
Options: 1-1201

bottom=*value* Ending row number of data.
Options: 1-1201

left=*value* Beginning column number of data.
Options: 1-1201

right=*value* Ending column number of data.
Options: 1-1201

output=*name* Name of the output file to hold the extracted DEM data.

logfile=*name* Name of a file to hold related information about the extracted data.

EXAMPLE:

For example, the command:

```
dd if=/dev/rmt0 ibs=10240 | \
m.dmaUSGSread top=1 bottom=400 left=1 right=500 \
   output=dem logfile=log
```

will extract data from /dev/rmt0 and put the first 400 rows and first 500 columns into the file *dem* located in the user's current directory.

SEE ALSO

Options for Acquiring Elevation Data, by Stuart Bradshaw (ADP Report N-87/22, USACERL, 1988).

DTED and DEM Elevation Data Extraction, by Stuart Bradshaw, Mary Martin, and Chester Kos (ADP Report N-87/22, USACERL, 1988).

and the UNIX manual entry for dd[1]

g.region, m.dted.examine, m.dted.extract, m.examine.tape, m.region.ll, m.rot90, r.describe, r.in.ll, r.rescale, r.slope.aspect

AUTHORS

James Farley, Arkansas Archeological Survey, University of Arkansas
Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

m.dted.examine − Provides a terse description of level 1 and 2 digital terrain elevation data (DTED) files produced and distributed by the Defense Mapping Agency (DMA) on 1/2-inch magnetic tapes.
(GRASS Data Import/Processing Program)

SYNOPSIS

m.dted.examine [*tapedev*]

DESCRIPTION

m.dted.examine scans digital terrain elevation data (DTED - Levels 1 and 2) that was produced and supplied by the Defense Mapping Agency (DMA) on 1/2-inch magnetic tape from the tape device specified by ***tapedev***, and prints to the screen a terse description of each file.

If the user specifies no ***tapedev*** on the command line the program assumes that input is coming from /dev/rmt0. The information provided by ***m.dted.examine*** includes the UTM borders, cell resolution, and number of elevation profiles in each file.

BUGS

The header file for DTED Level 1 and 2 data was changed in 1987. ***m.dted.examine*** and ***m.dted.extract*** operate only on DTED data containing pre-1987 headers. DTED data containing the pre-1987 headers may be purchased from the Defense Mapping Agency (DMA) upon request.

SEE ALSO

Pursuance of Elevation Data, by Stuart Bradshaw, USACERL
DEM and DTED Elevation Extraction, by Stuart Bradshaw, Mary Martin, and Chester Kos, USACERL

g.region, m.dmaUSGSread, m.dted.extract, m.examine.tape, m.region.ll, m.rot90, r.describe, r.in.ll, r.rescale, r.slope.aspect

AUTHORS

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory
Andrew Heekin, U.S. Army Construction Engineering Research Laboratory

NAME

m.dted.extract − Extracts digital terrain elevation data (DTED - levels 1 and 2) produced and supplied by the Defense Mapping Agency (DMA) on 1/2-inch magnetic tapes.
(GRASS Data Import/Processing Program)

SYNOPSIS

m.dted.extract if=*tapedev* of=*outfile* hf=*headfile* n=*lat* s=*lat* e=*lon* w=*lon*

DESCRIPTION

m.dted.extract extracts DTED (Levels 1 and 2) digital terrain elevation data produced by the Defense Mapping Agency (DMA) from 1/2-inch magnetic tapes obtained from DMA. Data is read from the input file (if) specified by *tapedev*. If the user does not specify any input file name on the command line, the program assumes that input is coming from /dev/rmt0. The extracted data is placed in the output file (of) specified by *outfile*. This program should be used in conjunction with the programs *m.rot90* and *r.in.ll* to convert DTED data to GRASS raster format.

if The pathname of the tape device where the raw DTED data exists (default is /dev/rmt0)

of The full pathname of the output file into which the extracted tape data is to be copied

hf The full pathname of a file to contain descriptive information about the extracted data; should be placed in the same directory as the output file

n North latitude value defining the boundaries of the extraction geographic region (format: dd.mm.ss[n|s])

s South latitude value defining the extraction geographic region (format: dd.mm.ss[n|s])

ea East longitude value defining the extraction geographic region (format: dd.mm.ss[e|w])

w West longitude value defining the extraction geographic region (format: dd.mm.ss[e|w])

The *n s e* and *w* parameters define a geographic region that should completely encompass the data set. *dd.mm.ss* are degree, minute, and second values. Only data that falls within this defined geographic region will be extracted from the tape.

EXAMPLE

The command line:

m.dted.extract if=/dev/rmt0 of=dted.out hf=dted.head n=37.30.00n s=37.15.00n e=103.30.00w w=103.45.00w

will extract DTED data from /dev/rmt0, store it in a file named *dted.out*, and store some supporting information in the file *dted.head*. Only data that falls within the geographic region defined by the coordinates *n,s,e* and *w* will be extracted.

NOTE

The user should examine the contents of the header file produced by this program; it contains information needed as inputs to the data rotation and raster file import programs *m.rot90* and *r.in.ll,* respectfully.

BUGS

The format of the header file for DTED Level 1 and 2 data was changed in 1987. *m.dted.extract* and *m.dted.examine* only operate on DTED data containing pre-1987 headers. DTED data containing the pre-1987 headers may be purchased from the DMA upon request.

DIAGNOSTICS

Invalid requests are so designated.

SEE ALSO

Pursuance of Elevation Data, by Stuart Bradshaw, USACERL
DEM and DTED Elevation Extraction, by Stuart Bradshaw, Mary Martin, and Chester Kos, USACERL

g.region, m.dmaUSGSread, m.dted.examine, m.examine.tape, m.region.ll, m.rot90, r.describe, r.in.ll, r.rescale, r.slope.aspect

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

m.examine.tape – Provides a description of the files on a 1/2-inch magnetic tape.
(GRASS File Management Tool)

SYNOPSIS

m.examine.tape

DESCRIPTION

m.examine.tape is a convenient tool for previewing a tape and obtaining block size values. After the user enters the command

m.examine.tape

the program will prompt the user for the full path name of the tape device on which the tape is mounted. In most cases, this will be /dev/rmt0. However, the user must enter this, as there is no default value for the tape device name. The user is then prompted to enter the buffer size to be used when reading the tape. The buffer size is the amount of memory allocated to read one physical record of the tape. If the user hits RETURN, the program will assume a default buffer size of 32767 bytes.

m.examine.tape reads the tape specified by the user and reports back the block size (record length) and the number of blocks for each file contained on the tape. *m.examine.tape* can be used on any 1/2-inch magnetic tape. The user has the option of sending the output into a file or viewing the output on the terminal screen.

NOTES

The buffer size is the amount of memory allocated to read one physical record on the tape. Magnetic tape devices will accept a request to read more bytes than actually exist in any given record on a tape. However, *m.examine.tape* reads only as many bytes as physically exist on the tape and returns the number of bytes actually read. The user is allowed to alter the buffer size in order to request smaller reads for tape devices unable to handle requests this large (32767 bytes), or to request larger reads for tapes with larger record sizes read on drives able to handle larger record sizes.

Note that *m.examine.tape* cannot be used to examine the content of either ordinary files or 1/4-inch tape cartridges.

SEE ALSO

m.dem.examine, m.dem.extract, m.dmaUSGSread, m.dted.examine, m.dted.extract, m.lulc.USGS, m.lulc.read

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

m.flip – Flips elevation data extracted from systems that retrieve data by rows from south to north.
(GRASS Data Import/Processing Program)

SYNOPSIS

m.flip
m.flip help
m.flip [**-q**] **input**=*name* **output**=*name* **rows**=*value* **cols**=*value* **bpc**=*value*

DESCRIPTION

m.flip is similar to *m.rot90*. However, rather than rotating a raster file, *m.flip* flips the contents of the raster generated by tape extraction programs about the east-west axis. Flipping may be necessary to compensate for the orientation of the data read from tape. This program can be used in conjunction with the program *r.in.ll* to convert this rotated data into GRASS raster format.

m.flip requires five inputs to be entered by the user. These parameters and the optional flag setting are described below.

Flag:

-q	Run quietly, suppressing output of messages on program progress to standard output.

Parameters:

input=*name*	The full pathname of an existing file containing the data to be flipped.
output=*name*	The full pathname of the output file in which the rotated data are to be stored.
rows=*value*	The number of rows of data in the *input* file. Values must be positive integers.
cols=*value*	The number of columns of data in the *input* file. Values must be positive integers.
bpc=*value*	The number of bytes per cell (i.e., per data value) in the *input* file. Values must be positive integers.

EXAMPLE

The following command:

m.flip input=/tmp/foo.out output=/tmp/flip.out rows=301 cols=358 bpc=2

will rotate the file /tmp/foo.out, and place the rotated file in /tmp/flip.out. Here, the input file is 301 rows by 358 columns, at 2 bytes per data value.

SEE ALSO

Pursuance of Elevation Data, by Stuart Bradshaw, USACERL
DEM and DTED Elevation Extraction, by Stuart Bradshaw, Mary Martin, and Chester Kos, USACERL

g.region, m.dmaUSGSread, m.dted.examine, m.dted.extract, m.examine.tape, m.region.ll, m.rot90, r.describe, r.in.ll, r.rescale, r.slope.aspect

AUTHORS

Donald Newcomb, U.S. Naval Oceanographic Office,
borrowing heavily from the *m.rot90* program written by Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

m.gc2ll – Converts geocentric to geographic coordinates.
(GRASS Data Import/Processing Program)

SYNOPSIS

m.gc2ll x=*#* y=*#* z=*#* s=*spheroid*

DESCRIPTION

m.gc2ll returns geographic coordinates for geocentric ones supplied by the user. It performs the reverse operation of the GRASS program *m.ll2gc*. The **x**, **y** and **z** values are the three dimensions needed to locate a point in three-dimensional space. The values that are printed include the latitude, the longitude and the height above (or distance below) the spheroid.

The list of spheroids available is somewhat dynamic. It may not contain exactly the ones listed below. To determine the current list of possible spheroids, simply type in:

m.gc2ll x=0 y=0 z=0 s=help

A list of available spheroids will be printed on the screen. If the spheroid desired is not on the list, the values for the semi-major axis and the eccentricity squared for the spheroid may be entered in place of a spheroid name in the following format:

s=a=*semi-major_axis*,e=*eccentricity_squared*

SOME POSSIBLE SPHEROIDS
(*The on-line listing includes only the spheroid names*)

Spheroid	Commonly used for:	Semi-major axis	Eccentricity sqrd
australian	Australia	6378160.0	0.0066945419
bessel	Japan	6377739.155	0.0066743722
clark66	N. America	6378206.4	0.006768658
clark80	France, Africa	6378249.145	0.0068035113
everest	India, Burma	6377276.345	0.0066378466
international	Europe	6378388.0	0.00672267
wgs72	worldwide coverage	6378135.0	0.006694317778

EXAMPLE

m.gc2ll x=0.0 y=0.0 z=6356750.520017 s=wgs72

Results:
lat=90N
lon=90W
h=0.0000

NOTES

For a brief discussion of spheroids see *m.ll2u*.

For a brief discussion of geocentric coordinates see *m.ll2gc*.

This is an experimental program. It is part of an initial effort to incorporate geographic coordinates into GRASS.

SEE ALSO

m.datum.shift, m.ll2gc, m.ll2u, m.u2ll

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

m.in.pl94.db3 – Import Demographic records from Census PL94-171 dBase3 (CDROM) Files
(GRASS Sites Program)

SYNOPSIS

m.in.pl94.db3 help
m.in.pl94.db3 [-f] [-p] [-n] sc=string [sc=str . .] in=/cdrom/pl9417ss.dbf > result_file
This command may be run in command line mode only. Parameters used must be in the order specified.

PARAMETERS

-f Output to stdout the list of all field name and starting positions for the Identification Section of the PL94 file. Other parameters are not necessary and will be ignored.

-p Print several information items to stderr input file is opened.

-n Output only the sequential number and PL94 logical record number (LOGRECNU) of matching records to stdout.

sc=string sc is starting column or a capitilized reference name from the Identification Section of the PL94 Data Dictionary; string is characters to match there. A negative test may also be made with: sc!=string. If more than one test is specified the results are "anded" resulting in a conjunction of the results of the expressions. (See *m.in.stf1.tpe* for complete details.)

in Path/file for dBase files for state *ss*.

DESCRIPTION

This program is provided to read the Census PL94 data records from a file for one state. The file is usually on a CD-ROM, but this is not necessary providing the proper path is given for the ***in=*** parameter and the contents of the file duplicate that on CD-ROM.

The output consists of records of 517 characters extracted from the input file with a suitable line termination character added (0 to create a normal text file.

Users of GRASS 4.1 and the Census PL94 files on CD-ROM will generally use this program to select appropriate subsets of PL94 records; write them to one or more files; and then either process these files directly with *v.apply.census* to print demographics, create site maps or label vector maps, or use *m.in.stf1.tpe* to extract further subsets.

The PL94 CD-ROM data file for most states is very large, 100+ MB. This program may take 20 to 60 minutes for a single extraction, so plan carefully and don't discard useful subsets that you create on disk.

EXAMPLES

Insert the appropriate CD-ROM and mount it (see your system administrator). (We assume here that it is mounted as the file system /cdrom.)

To extract all Census tract summary records for the state of South Dakota:

m.in.pl94.db3 -p 11=050 in=/cdrom/pl9417sd.dbf > counties.sd

The 11=050 parameter selects those records which have the character string Summary Level (SUMLEV) field which is used in almost every selection from CD-ROM.

To extract all block records for Lawrence County, South Dakota:

m.in.pl94.db3 -p SUMLEV=100 CNTY=081 in=/cdrom/pl9417sd.dbf >

100 is the SUMLEV for blocks and 081 is the county code for Lawrence Co.

SEE ALSO

m.in.stf1.tpe is used as a post-processor to select further subsets of PL94 records. NOTE: Unix utility programs, such as *grep* or *awk*, can also be used for the for processing the extracted PL94 files.

AUTHOR

Dr. James Hinthorne, GIS Laboratory, Central Washington University, August 1992.

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

m.in.stf1.db3 – Import Demographic records from Census STF1A dBase3 (CDROM) Files
(GRASS Sites Program)

SYNOPSIS

m.in.stf1.db3 help
m.in.stf1.db3 [-f] [-p] [-n] sc=string [sc=str . .] in=/cdrom/stf1a0ss.dbf > result_file
This command may be run in command line mode only. Parameters used must be in the order specified.

PARAMETERS

-f Output to stdout the list of all field name and starting positions for the Identification Section of the STF1A file. Other parameters are not necessary and will be ignored.

-p Print several information items to stderr as the 10 input files are opened.

-n Output only the sequential number and STF1A logical record number (LOGRECNU) of matching records to stdout.

sc=string sc is starting column or a capitilized reference name from the Identification Section of the STF1A Data Dictionary; string is characters to match there. A negative test may also be made with: sc!=string. If more than one test is specified the results are "anded" resulting in a conjunction of the results of the expressions. (See *m.in.stf1.tpe* for complete details.)

in Path/file for dBase files for state *ss*.

DESCRIPTION

This program is provided to read and assemble the Census STF1A data records from a set of 10 files for one state. The files are usually on a CD-ROM, but this is not necessary providing the proper path is given for the *in=* parameter and all 10 files are in the same directory with their normal names.

The output consists of data from the 10 input files assembled into records identical to the STF1A tape distribution format. The output records are each about 4800 characters and a pair constitute a logical STF1A record.

Users of GRASS 4.1 and the Census STF1A files on CD-ROM will generally use this program to select appropriate subsets of STF1A records; write them to one or more files (in the STF1A tape format); and then either process these files directly with *v.apply.census* to print demographics, create site maps or label vector maps, or use *m.in.stf1.tpe* to extract further subsets.

The STF1A CD-ROM data sets for most states are very large, 100+ MB. This program may take 20 to 60 minutes for a single extraction, so plan carefully and don't discard useful subsets that you create on disk.

EXAMPLES

Insert the appropriate CD-ROM and mount it (see your system administrator). (We assume here that it is mounted as the file system /cdrom.)

To extract all Census tract summary records for the state of South Dakota:

m.in.stf1.db3 -p 11=140 in=/cdrom/stf1a0sd.dbf > tracts.SD

The 11=140 parameter selects those records which have the character string Summary Level (SUMLEV) field which is used in almost every selection from CD-ROM.

To extract all block group records for Lawrence County, South Dakota:

m.in.stf1.db3 -p SUMLEV=150 CNTY=081 in=/cdrom/stf1a0sd.dbf >

150 is the SUMLEV for block groups and 081 is the county code for Lawrence Co.

SEE ALSO

m.in.stf1.tpe is used as a post-processor to select further subsets of STF1A records. NOTE: Unix utility programs, such as ***grep*** or ***awk***, cannot be used for the for processing the STF1 files due to their very long record lengths.

AUTHOR

Dr. James Hinthorne, GIS Laboratory, Central Washington University, May 1992.

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

m.in.stf1.tape – Filter to extract lines from a text file based on column contents, especially for Bureau of the Census STF1 files.
(GRASS Support Program)

SYNOPSIS

m.in.stf1.tape
m.in.stf1.tape help
m.in.stf1.tape -f
m.in.stf1.tape [-n] sc=str [sc=str . . .] < infile > outfile

This program must be run in command mode only.

PARAMETERS

sc is a starting column number of a desired field in the input file, or is the name of one of the Identification Section field names for the STF1 records (all upper case letters), and *str* is a string to match against input lines starting at column *sc*. *sc=str* may be repeated resulting in a conjunction (anding) of the results of each *sc=str* expression. A '?' may be used as a single character wild card in *str*; if *sc=str* contains '?' or other shell interpreted characters, it should be protected in quotes. Preceeding *sc* by 'N', or preceeding the = by '!' reverses the sense of the test ("not equals").

Examples:
m.in.stf1.tape 11=050 < infile > outfile
m.in.stf1.tape SUMLEV=050 < infile > outfile
m.in.stf1.tape 1=T450 '7=Bu??s' < infile > outfile
m.in.stf1.tape 51=tract N37=9753 < infile > outfile
m.in.stf1.tape 51=tract 37!=9753 < infile > outfile

Running the program with the *-f* flag generates the list of STF1 Identification Section field names to stdout.

DESCRIPTION

This is a text filter program written in C especially to work with the Census STF1 files, but useful for selecting subsets of lines from any text file. It will work with arbitrarily long input lines, up to 10,000 characters. Input lines may be of variable length.

A *sc=str* condition which refers to columns beyond the end of the input line is assumed to be true.

Multiple tests are 'anded' into a single test; that is, lines which pass all tests are written to the output file.

Null characters are always filtered out; thus, this program can be used as a null filter by specifying a *sc=str* condition which will always be true (e.g., '1=?'). In the same way, files with lines terminated with <LF><CR> or just <CR> can be "fixed" to have the standard Unix <LF> terminator.

NOTE: One special property of *m.in.stf1.tape* is that it will pad (with 0) its output lines to 4806 characters if the line begins with the characters "STF1" and is greater than 4000 characters long. Experience has shown that some STF1 records are a few characters short (but no data has been omitted), and this corrects them so that other programs will be able to read full lines.

BUGS/FEATURES

Input lines must be terminated with <LF>, <CR> or <LF><CR>. Output lines will be terminated with <LF> only.

Input characters lexically less than 'space' (32 decimal; the "control" characters) which are not line

terminators will be perceived as line terminators and thus cause improper functioning.

SEE ALSO

For extracting lines from files, Unix programs such as *grep*, or *awk* are somewhat more flexible than *m.in.stf1.tape*, but have line length limitations and do not adapt to lines terminated with <LF> and <CR>.

This program was created as a preprocessor for Census STF1 files, to produce input files for the program *v.apply.census*. The program *m.in.stf1.db3* performs the same preprocessing function for the Census STF1 files distributed in "dBase 3" format on CD-ROM.

AUTHOR

Dr. James Hinthorne, GIS Laboratory, Central Washington University.

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

m.ll2gc – Converts geographic coordinates to geocentric coordinates.
(GRASS Data Import/Processing Program)

SYNOPSIS

m.ll2gc lat=*dd.mm.ss{n| s}* lon=*dd.mm.ss{e| w}* [h=*height*] s=*spheroid*

DESCRIPTION

m.ll2gc returns geocentric coordinates for geographic coordinates (latitude and longitude). Geographic coordinates are in degrees, minutes, and seconds and must include designation of north or south {n| s} and east or west {e| w}. The spheroid on which they are based must also be entered. The height (in meters) above the spheroid is optional.

The list of spheroids available is somewhat dynamic. It may not contain exactly the ones listed below. To determine the current list of possible spheroids, type in:

m.ll2gc lat=0n lon=0w s=help

A list of available spheroids will be printed on the screen. If the spheroid desired is not on the list, the values for the semi-major axis and the eccentricity squared for the spheroid may be entered in place of a spheroid name in the following format:

s=a=*semi-major_axis*,e=*eccentricity_squared*

SOME POSSIBLE SPHEROIDS
(***The on-line listing includes only the spheroid names***)

Spheroid	Commonly used for:	Semi-major axis	Eccentricity sqrd
australian	Australia	6378160.0	0.0066945419
bessel	Japan	6377739.155	0.0066743722
clark66	N. America	6378206.4	0.006768658
clark80	France, Africa	6378249.145	0.0068035113
everest	India, Burma	6377276.345	0.0066378466
international	Europe	6378388.0	0.00672267
wgs72	worldwide coverage	6378135.0	0.006694317778

EXAMPLE

m.ll2gc lat=0n lon=90w s=wgs72

Results:

x=0.0
y=6378135.0
z=0.0

The distances designated by "x," "y" and "z" are in meters from the center of the earth. Because the sample point is on the equator and is at 90 degrees west, two of the three parameters have a value of zero (i.e., the point is at the origin of those two dimensions).

NOTES

For a brief discussion of spheroids see ***m.ll2u***.

Geographic coordinates (latitude/longitude) describe the location of a point on the earth relative to the equator in the north/south directions and relative to Greenwich in the east/west directions. The same point will have different coordinates depending on the spheroid used as the approximation of the shape of the earth. The geocentric coordinates given here describe the point on the basis of the same spheroid, but they use the center of the spheroid as their origin. This can be thought of as being the center of the earth, which

is why they are called "geocentric." Describing the location of a point in a spheroid requires three points: x, y, and z; these are measured out from the center along three different axes. The distances given are in meters.

This is an experimental program. It is part of initial efforts to incorporate geographic coordinates into GRASS.

SEE ALSO

m.datum.shift, m.gc2ll, m.ll2u, m.u2ll

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

m.ll2u – Converts geographic coordinates to Universal Transverse Mercator (UTM) coordinates. (GRASS Data Import/Processing Program)

SYNOPSIS

m.ll2u
m.ll2u help
m.ll2u [**-rwoz**] **spheroid**=*name* [**zone**=*value*] [**input**=*name*] [**output**=*name*]

DESCRIPTION

m.ll2u converts geographic coordinates (i.e., latitudes and longitudes) to Universal Transverse Mercator (UTM) eastings and northings. The user must specify the spheroid on which to base the UTM conversion. The user may optionally specify the UTM zone; however, the program can determine this from the geographic coordinates submitted.

The list of spheroids available is somewhat dynamic. At the time of this release, available spheroids included: airy, australian, bessel, clark66, everest, grs80, hayford, international, krasovsky, wgs66, wgs72, and wgs84 (see table below).

This command can be run either non-interactively or interactively. The user can run the program non-interactively by entering desired flag settings and parameter values on the command line using the following format:

m.ll2u [**-rwoz**] **spheroid**=*name* [**zone**=*value*] [**input**=*name*] [**output**=*name*]

Alternately, the user can simply type:

m.ll2u

on the command line. In this case, the user will be prompted for parameter values and flag settings through the standard interface described in the manual entry for ***parser***.

Input can be entered from the keyboard or from an input file. In either case, input should be entered with one longitude and latitude pair per line, in either of the below forms:

degrees:minutes:seconds{E|W} degrees:minutes:seconds{N|S}
degrees:minutes:seconds{E|W} degrees:minutes:seconds{N|S}
degrees:minutes:seconds{E|W} degrees:minutes:seconds{N|S}
degrees:minutes:seconds{E|W} degrees:minutes:seconds{N|S}
.
.
end

degrees.decimal{E|W} degrees.decimal{N|S}
degrees.decimal{E|W} degrees.decimal{N|S}
degrees.decimal{E|W} degrees.decimal{N|S}
degrees.decimal{E|W} degrees.decimal{N|S}
.
.
end

If the user sets the -r flag, ***m.ll2u*** will expect the order of the coordinates to be reversed, and stated as latitude, longitude pairs, rather than as longitude, latitude pairs.

Similarly, the user can elect to send output to an output file or (by default) to standard output (the user's terminal screen). If the user sets the -w flag, output will be printed in a format suitable for input to programs like ***d.points***. Example input and output are shown below (see EXAMPLE).

Program flag settings and parameters have the following meanings.

Flags:

-r The order of coordinates is reversed in the input, and entered as: *lat lon*

-w Do not flag invalid lon,lat input lines as errors.

-o Flag other invalid input lines as errors.

-z Suppress printing of the UTM zone in the output. (Note. This will produce output in a format suitable for direct input to programs like *d.points*.)

Parameters:

spheroid=*name* Reference spheroid (ellipsoid).
Options: airy, australian, bessel, clark66, everest, grs80, hayford, international, krasovsky, wgs66, wgs72, wgs84

zone=*value* UTM zone number (results will be forced into this UTM zone).
Options: 1-60

input=*name* Name of an existing input file containing longitude, latitude coordinates to be converted. Input lines may either be input in the form of degrees:minutes:seconds, or as decimal degrees. If input as decimal degrees, *m.ll2u* recognizes negative numbers.

output=*name* Name to be assigned to the output file containing UTM coordinates and zone designations.

AVAILABLE SPHEROIDS
(*The on-line listing includes only the spheroid names*)

Spheroid:	Semi-major axis (Equatorial Radius) (a):	Eccentricity sqrd (e), Flattening (f), or Polar Radius (b):	Commonly used for:
airy	a=6377563.396	e=.006670540	
australian	a=6378160	f=1/298.25	Australia
bessel	a=6377397.155	e=.006674372	Japan
clark66	a=6378206.4	b=6356583.8	N. America
everest	a=6377276.345	e=.0066378466	India, Burma
grs80	a=6378137	f=1/298.257	
hayford	a=6378388	f=1/297	
international	a=6378388	f=1/297	Europe
krasovsky	a=6378245	f=1/298.3	
wgs66	a=6378145	f=1/298.25	worldwide coverage
wgs72	a=6378135	f=1/298.26	worldwide coverage
wgs84	a=6378137	f=1/298.257223563	worldwide coverage

EXAMPLE

Assume the user has input the command:

m.ll2u spheroid=wgs72 input=ll.infile output=utm.outfile

where the input file *ll.infile* contains the following longitude, latitude values:

```
155:30:00W 19:35:00N
165:30:00W 19:35:00N
145:30:00W 20:00:00N
135:30:00W 21:00:00N
end
```

Output would then be sent to the output file *utm.outfile*, containing the below Easting and Northing coordinate values and UTM zone designations:

237740.85270818 2167292.1076231 5
447560.64349407 2165450.19058336 3
656921.61734802 2212183.40032627 6
448035.11644906 2322228.3038167 8
end

NOTES

Spheroids, the solids associated with an ellipse, are also known as ellipsoids. They are used as the best possible model to simulate the shape of the earth with its flattened poles and bulging equator. They are the mathematical means for establishing control points to use as a reference when determining exact locations on the earth. A cohesive set of these control points is called a **datum**.

The spheroids listed above have been used as the basis for a number of different datums. The North American Datum of 1927 (NAD 27) was based on the Clark 1866 ("clark66") spheroid. This was a recent current standard datum for North America. Be aware, however, that a new datum, NAD 83, has been developed using the Geodetic Reference System 1980 spheroid; this is now available in ***m.ll2u*** as the "grs80" spheroid. The "wgs66", "wgs72", and "wgs84" spheroids are for worldwide use. The "wgs72" spheroid has been used fairly widely by the Department of Defense (DOD) and is the basis for the World Geodetic System 1972 datum. This datum has also been recently replaced; the new DOD datum is WGS 84 (wgs84). Both datums are available within ***m.ll2u***.

To use spheroids other than those listed here, the user can add lines to the ellipsoid parameter definition file in $GISBASE/etc/ellipse.table.

Read the marginalia of your source map to determine which spheroid was used to produce the map on which you are working.

This program recognizes negative numbers if coordinates are input in decimal degrees rather than in the form of degrees:minutes:seconds.

This program has received only limited testing. It should be used with some caution.

FILES

See ellipsoid parameter definition file in $GISBASE/etc/ellipse.table.

SEE ALSO

For australian, clark66, grs80, hayford, international, krasovsky, and wgs72 ellipsoid parameters, see:
John P. Snyder, ***Map Projections - A Working Manual***, U.S. Government Printing Office, Washington DC, 1989. U.S. Geological Survey Professional Paper 1395; from Table 1, p.12.

For bessel, airy, everest, and wgs66 ellipsoid parameter values, see:
Thomas O. Seppelin, ***The Department of Defense World Geodetic System 1972***, presented at the International Symposium on Problems Related to the Redefinition of North American Geodetic Networks, Fredericton, New Brunswick, Canada in May, 1974; see Table 9, p.35.

For wgs84 parameter values, see:
U.S. Naval Oceanographic Labs.

Also read GRASS User's Reference Manual entries for:
d.labels, d.points, d.sites, d.where, m.datum.shift, m.gc2ll, m.ll2gc, m.u2ll, and ***parser***

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

m.lulc.USGS – Creates raster map layers from a Composite Theme Grid (CTG) file created by ***m.lulc.read***. ***m.lulc.read*** extracts the CTG data from an ASCII landuse/landcover (lulc) CTG format file supplied by the USGS.
(GRASS Data Import/Processing Program)

SYNOPSIS

m.lulc.USGS *input_file*

DESCRIPTION

m.lulc.USGS creates the following raster map layers from the CTG file created by ***m.lulc.read***, and places them into the user's current mapset:

1. Land Use & Land Cover
2. Political Units
3. Census Units
4. Hydrologic Units
5. Federal Land Ownership
6. State Land Ownership

Since the CTG file may not contain all of layers listed above, ***m.lulc.USGS*** will extract only the raster map layers which exist in the CTG file.

The CTG file contains all of the geographic region definition information necessary for creating the raster map layers, including:

1. Grid Zone
2. Projection Type (UTM)
3. Cell Resolution (ew_res = nw_res)
4. Multi-byte data formation
5. Geographic region coordinates

m.lulc.USGS will use the geographic region information definition supplied with the CTG file and NOT the geographic region definition currently set for the user's mapset; (note that this is different than is the case with many of the other GRASS map development commands).

When ***m.lulc.USGS*** is executed, it will check the file for a specified layer and then ask the user if he wishes to extract the associated raster map layer. The user is then prompted for the name of a new raster file in which output is to be placed. ***m.lulc.USGS*** will then create this raster file and all supporting (e.g., category, cell header, color table, etc.) files.

EXAMPLE

In the example below, raster data is extracted from a CTG file (named *lulc*) that contains only LAND USE/LAND COVER data:

m.lulc.USGS lulc

MAP TITLE: DODGE CITY, KS 1:250,000 LU, PB, CN, HU, FO
The Composite Theme Grid file contains <1> overlays
and has a map code type of <01>

Do you Wish to Create <LAND USE/LAND COVER> Raster File (y/n) [y] **y**

Enter File Name for LAND USE/LAND COVER Overlay
Enter 'list' for a list of existing raster files
Hit RETURN to cancel request
> **landuse**

Creating <landuse> from <LAND USE/LAND COVER> Overlay: 95%
Number of Categories: 76 (UNLABELED)
Writing Cell Header Information
Writing Color Table Information

Conversion is Complete

BUGS

m.lulc.USGS does not currently extract Census unit data. Also, only the cataloging unit is extracted from the CTG file (see the ***Land Use and Land Cover Data User's Guide*** supplied by the USGS).

SEE ALSO

USGS **Land Use and Land Cover Data User's Guide**

g.region, m.lulc.read

AUTHOR

Kenneth Shepardson
Spectrum Sciences & Software, Inc.
(904) 862-3031

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

m.lulc.read − Extracts Landuse/Landcover data in the ASCII Composite Theme Grid (CTG) data format distributed by the USGS in to a working file for *m.lulc.USGS*.
(GRASS Data Import/Processing Program)

SYNOPSIS

dd [**if**=*tapedev*] **ibs**=*input block_size* cbs=*80* **conv**=*unblock* **| m.lulc.read** *arg1*

DESCRIPTION

m.lulc.read extracts USGS Land Use/Land Cover data distributed in the ASCII CTG format. Extracted data is placed into a file specified by *arg1*. *m.lulc.read* should be used prior to using the GRASS program *m.lulc.USGS*.

m.lulc.read reads the data from standard input, allowing the user to pipe in the data from a file.

Data can also be read directly from the 1/2-inch magnetic tape distributed by USGS, by using the UNIX command *dd*.

Parameters:

if=*tapedev*
The pathname of the input tape drive (usually **/dev/rmt0**).

ibs=*input*
The physical blocking value of the data on tape, usually written on the tape spool. If this value is unknown, run *m.examine.tape* prior to *m.lulc.read*.

cbs This is the conversion blocking factor for ASCII CTG file and is by default set to **80**.

conv Set *conv* to **unblock** when extracting ASCII CTG data.

arg1 The name of the output file where the data will be stored in binary format.

EXAMPLES

READING FROM A FILE:

m.lulc.read outfile < infile

The above command will read the data from *infile* and place the results into *outfile*. Note that *infile* must be extracted from the USGS CTG 1/2 magnetic tape using the UNIX *dd* command:

dd if=/dev/rmt0 of=infile ibs=32000 cbs=80 conv=unblock

READING DIRECTLY FROM 1/2 INCH MAGNETIC TAPE:

dd if=/dev/rmt0 ibs=32000 cbs=80 conv=unblock | m.lulc.read outfile

The above command will extract data with a blocksize of 32000 from */dev/rmt0*, and put the results in the file *outfile*.

SEE ALSO

UNIX manual entry for *dd*

m.examine.tape, m.lulc.USGS

AUTHOR

Kenneth Shepardson, Spectrum Sciences & Software, Inc. (904) 862-3031

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

m.proj – Calculates conversion coordinates for geographic positions.
(GRASS Map Development Program)

SYNOPSIS

m.proj
m.proj help

DESCRIPTION

This program allows a user to interactively convert coordinates from one projection to another.

It allows a user to do all of the above: reading from a file and writing to the screen, or reading from the keyboard and writing to a file, or reading from a file and writing to a file.

Note: The program does not transform GRASS files; it is designed to determine coordinate values on an individual position.

The map projections currently supported are listed in etc/projections file.

Each of the above projections (with the exception of State Plane) can be computed with the spheroids listed in the etc/ellipse.table file.

INPUT FILE FORMAT

When reading from a file of LATITUDE/LONGITUDE data, the file will contain two (2) columns of information:

the first column: latitude - in degrees minutes seconds
the second column: longitude - in degrees minutes seconds

For example:

+40 36 31.4563	-87 2 7.8193
40n 36 31.4563	87w 2 7.8193

When reading from a file of PROJECTION COORDINATES data, the file will contain two (2) columns of information:

the first column: easting - ground coordinates
the second column: northing - ground coordinates

For example:

500000.00	4496918.64
-424489.11	1908736.13

Note: No column headings are required, only the numbers.

AUTHORS

Irina Kosinovsky, US Army CERL
R.L. Glenn, USDA, SCS, NHQ-CGIS

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

m.region.ll − Converts Universal Transverse Mercator (UTM) coordinates falling within the current geographic region from UTM coordinates to geographic (latitude/longitude) coordinates.
(GRASS Data Import/Processing)

SYNOPSIS

m.region.ll
m.region.ll help
m.region.ll spheroid=*name*

DESCRIPTION

m.region.ll takes current geographic region settings in UTM coordinates, and converts them to geographic coordinates (i.e., latitudes and longitudes). It also prints the length (in meters) of one arc-second at each of the four edges of the geographic region. The user must enter the spheroid upon which to base the geographic coordinates.

The list of spheroids available is somewhat dynamic. It may not contain exactly the ones listed below. To determine the current list of possible spheroids, simply type in:

m.region.ll help

A list of available spheroids will be printed on the screen.

AVAILABLE SPHEROIDS
(The on-line listing includes only the spheroid names.)

Spheroid:	Semi-major axis (Equatorial Radius) (a):	Eccentricity sqrd (e), Flattening (f), or Polar Radius (b):	Commonly used for:
airy	a=6377563.396	e=.006670540	
australian	a=6378160	f=1/298.25	Australia
bessel	a=6377397.155	e=.006674372	Japan
clark66	a=6378206.4	b=6356583.8	N. America
everest	a=6377276.345	e=.0066378466	India, Burma
grs80	a=6378137	f=1/298.257	
hayford	a=6378388	f=1/297	
international	a=6378388	f=1/297	Europe
krasovsky	a=6378245	f=1/298.3	
wgs66	a=6378145	f=1/298.25	worldwide coverage
wgs72	a=6378135	f=1/298.26	worldwide coverage
wgs84	a=6378137	f=1/298.257223563	worldwide coverage

EXAMPLE

m.region.ll spheroid=clark66

Results:

```
        WINDOW    4928000.00N     609000.00E    ZONE 13
                  4914000.00S     590000.00W

  44.30.06N       44.29.57N
103.52.04W      103.37.44W

  44.22.32N       44.22.23N
103.52.13W      103.37.55W

At northern edge 1 arc-second longitude=22.088500m
At southern edge 1 arc-second longitude=22.135998m
At western edge 1 arc-second latitude=30.860285m
At eastern edge 1 arc-second latitude=30.863082m
```

The values for the geographic coordinates are rounded to the nearest second in this example. They would be more precise in the actual output that is printed on the screen.

SEE ALSO

m.datum.shift, m.ll2u, m.u2ll, r.in.ll

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

m.rot90 – Rotates elevation data extracted by either *m.dted.extract* or *m.dmaUSGSread*
(GRASS Data Import/Processing Program)

SYNOPSIS

m.rot90
m.rot90 help
m.rot90 [**-q**] **input**=*name* **output**=*name* **rows**=*value* **cols**=*value* **bpc**=*value*

DESCRIPTION

m.rot90 is used as a companion program to the DTED and DEM digital elevation data tape extraction programs *m.dted.extract* and *m.dmaUSGSread*. *m.rot90* rotates the contents of the output files generated by these tape extraction programs 90 degrees. Rotation is necessary to compensate for the orientation of the data read from tape. This program can be used in conjunction with the program *r.in.ll* to convert this rotated data into GRASS raster map layer form.

m.rot90 requires five inputs to be entered by the user. These parameters and the optional flag setting are described below.

Flags:

-q Run quietly, suppressing output of messages on program progress to standard output.

Parameters:

input=*name* The full pathname of an already-existing file containing the data to be rotated. This *input* file is usually the output file created by *m.dted.extract* or *m.dmaUSGSread*.

output=*name* Name to be assigned to the resultant, rotated output file. The full pathname of the output file in which the rotated data are to be stored.

rows=*value* The number of rows of data in the *input* file.

cols=*value* The number of columns of data in the *input* file.

bpc=*value* The number of bytes per cell in the *input* file.

EXAMPLE

The following command:

m.rot90 input=/tmp/dma.out output=/tmp/rot.out rows=301 cols=358 bpc=2

will rotate the file /tmp/dma.out, and place the rotated file in /tmp/rot.out. Here, the input file is 301 rows by 358 columns, at 2 bytes per data value.

NOTES

The user should note that since the output file is rotated 90 degrees from the original input file, the rows and columns have been interchanged. Hence, in the above example, the number of rows (301) and columns (358) stated by the user on the command line were those present in the *input* file. The output file, however, will have 358 rows and 301 columns.

SEE ALSO

Pursuance of Elevation Data, by Stuart Bradshaw, USACERL
DEM and DTED Elevation Extraction, by Stuart Bradshaw, Mary Martin, and Chester Kos, USACERL

g.region, m.dmaUSGSread, m.dted.examine, m.dted.extract, m.examine.tape, m.flip, m.region.ll, r.describe, r.in.ll, r.rescale, r.slope.aspect

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

m.tiger.region – Finds geographic region information for U.S. Census Bureau TIGER input data.
(GRASS Data Import/Processing Program)

SYNOPSIS

m.tiger.region
m.tiger.region help
m.tiger.region infile=*name [***zone**=*value*] [**spheroid**=*name*]

DESCRIPTION

m.tiger.region is a program designed to evaluate a file of raw type 1 Census (TIGER) data and determine the geographic region covered by that input file. Output is sent to standard out, and gives the east, west, north, and south boundaries for the given input data file.

If the user specifies the input file name and (optionally) the zone number or spheroid to be used on the command line, the program will run non-interactively; if no zone number or spheroid name is given, the default(s) will be used (see below). Alternately, the user may simply type *m.tiger.region* on the command line; in this case, the program will prompt the user for parameter values using the standard GRASS parser interface described in the manual entry for *parser*.

COMMAND LINE OPTIONS

Parameters:

infile=*name* Input file name in which raw TIGER data (type 1) are stored.

zone=*value* Universal Transverse Mercator (UTM) zone for this county.
Options: -60 - 60
Default: 0

spheroid=*name* Name of spheroid to be used.
Default: clark66

Available spheroids are:

australian
bessel
clark66
clark80
everest
international
wgs72
wgs84

It is recommended that the user choose the clark66 (default) spheroid when dealing with TIGER data as it is the most consistent with the original data.

EXAMPLES

If the user typed simply:

m.tiger.region infile=inputfilename

program output would look similar to this:

Number of calculated zones is: 2

INFO FOR ZONE 1:
zone number: 13

percentage of data points	
in this zone:	0.799489
regional spread of points	
within this zone:	
north:	5092049.155918
south:	5049238.983803
east:	734139.517650
west:	732514.747908

INFO FOR ZONE 2:	
zone number:	14
percentage of data points	
in this zone:	99.200508
regional spread of points	
within this zone:	
north:	5092041.463966
south:	5036134.342322
east:	398030.217441
west:	265527.656108

If the user does not input the UTM zone number, it is calculated for them. Then the zone number and region information are output, and if the program finds that the input data contains information in more than one UTM zone, then the output is given for all applicable zones.

If instead the user supplies the UTM zone number, the output would look like that shown below:

REGION FOR THIS DATA FILE:

north border:	5092049.155918
south border:	5036134.342322
east border:	398030.217441
west border:	265527.656108
(zone number:	14)

NOTES

This command must be compiled separately. It will not automatically be included in the compile of the main GRASS code. Although ***m.tiger.region*** does not need a FORTRAN compiler, it is used to support other TIGER data functions (like ***v.db.rim***, ***v.in.tiger***, and ***rim***) which do require access to a FORTRAN compiler.

TIGER data are presented in latitude/longitude format, and are converted to UTM coordinates using coordinate conversion routines contained in the GRASS library. If no UTM zone number is supplied by the user, the program calculates the appropriate zone(s) based on the input data provided. The output then provides the UTM zone numbers found (if more than one), the geographic region covered within each zone, and the percentage of data points found in each zone. The user must then decide which of these UTM zones contains the major or most important portion of data values, so that the zone number can be supplied in creating the GRASS location to hold the imported data and can be provided to the importing program (***v.in.tiger***). Zone edges will be extended (reasonably) to include data values lying outside the chosen zone. If desired, ***m.tiger.region*** can be re-run, supplying the chosen zone number, in order to evaluate the region edges of the input data set (with the extended zone).

FILES

Source code for RIM is located under $GISBASE/../src.related/rim

Source code for ***v.db.rim*** is located under $GISBASE/../src.garden/grass.rim/v.db.rim

Source code for ***v.in.tiger*** is located under $GISBASE/../src.garden/grass.tiger/v.in.tiger

Source code for ***m.tiger.region*** is located under $GISBASE/../src.garden/grass.tiger/m.tiger.region

SEE ALSO

Gen.Maps, Gen.tractmap, g.region, v.db.rim, v.in.tiger, tiger.info.sh

AUTHOR

Marjorie Larson, U.S. Army Construction Engineering Research Laboratory

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

m.u2ll – Converts Universal Transverse Mercator (UTM) coordinates to geographic (latitude/longitude) coordinates.
(GRASS Data Import/Processing Program)

SYNOPSIS

m.u2ll
m.u2ll help
m.u2ll [**-srwod**] **spheroid**=*name* [**zone**=*value*] [**input**=*name*] [**output**=*name*]

DESCRIPTION

m.u2ll converts Universal Transverse Mercator (UTM) northings and eastings to geographic coordinates (i.e., latitudes and longitudes). The user must specify the UTM coordinates to be converted and the spheroid on which the geographic coordinates will be based. The program also needs to know the UTM zone in which the input coordinates are located. However, if the user is running GRASS from a UTM data base LOCATION, *m.u2ll* will use this data base's UTM zone designation, if no zone is specified by the user.

The GRASS program *m.ll2u* performs the reverse operation, converting geographic coordinates to UTM coordinates.

The list of spheroids available is somewhat dynamic. At the time of this release, available spheroids included: airy, australian, bessel, clark66, everest, grs80, hayford, international, krasovsky, wgs66, wgs72, and wgs84 (see table below).

This command can be run either non-interactively or interactively. The user can run the program non-interactively by entering desired flag settings and parameter values on the command line using the following format:

m.u2ll [**-srwod**] **spheroid**=*name* [**zone**=*value*] [**input**=*name*] [**output**=*name*]

Alternately, the user can simply type:

m.u2ll

on the command line. In this case, the user will be prompted for parameter values and flag settings through the standard interface described in the manual entry for *parser*.

Input can be entered from the keyboard or from an input file. In either case, input should be entered with one UTM easting and northing pair per line, in the format shown below:

```
easting northing
easting northing
easting northing
easting northing
  .
  .
end
```

If the user sets the -r flag, *m.u2ll* will expect the order of the coordinates to be reversed, and stated as northing, easting pairs, rather than as easting, northing pairs. This is useful for passing ASCII GRASS vector (*/dig*) files, whose coordinates are stated as northing, easting pairs, directly through *m.u2ll*.

Similarly, the user can elect to send output to an output file or (by default) to standard output (the user's terminal screen). Example input and output are shown below (see EXAMPLE).

Program flag settings and parameters have the following meanings.

Flags:

-s Specified UTM zone is in the southern hemisphere.

-r The order of coordinates is reversed in the input, and entered as: *north east*. This option allows the user to pass an ascii vector file through *m.u2ll*.

-w Do not flag invalid east, north input lines as errors.

-o Flag other invalid input lines as errors.

-d Output latitude/longitude values in decimal degrees, rather than in the form *dd:mm:ss*.

Parameters:

spheroid=*name* Reference spheroid (ellipsoid).
Options: airy, australian, bessel, clark66, everest, grs80, hayford, international, krasovsky, wgs66, wgs72, wgs84

zone=*value* UTM zone in which UTM coordinates are located.
Options: 1-60

input=*name* Name of input file containing UTM values to be converted.

output=*name* Name to be assigned to output file containing longitude and latitude values.

AVAILABLE SPHEROIDS
(The on-line listing includes only the spheroid names.)

Spheroid:	Semi-major axis (Equatorial Radius) (a):	Eccentricity sqrd (e), Flattening (f), or Polar Radius (b):	Commonly used for:
airy	a=6377563.396	e=.006670540	
australian	a=6378160	f=1/298.25	Australia
bessel	a=6377397.155	e=.006674372	Japan
clark66	a=6378206.4	b=6356583.8	N. America
everest	a=6377276.345	e=.0066378466	India, Burma
grs80	a=6378137	f=1/298.257	
hayford	a=6378388	f=1/297	
international	a=6378388	f=1/297	Europe
krasovsky	a=6378245	f=1/298.3	
wgs66	a=6378145	f=1/298.25	worldwide coverage
wgs72	a=6378135	f=1/298.26	worldwide coverage
wgs84	a=6378137	f=1/298.257223563	worldwide coverage

EXAMPLE

Assume the user has input the command:

m.u2ll -s spheroid=wgs72 zone=4 input=utm.infile output=ll.outfile

where the input file *utm.infile* contains the following easting and northing UTM coordinate values and zone designations:

```
237740.85 2167292.10
238740.00 2167000.00
239000.00 2167100.00
237100.00 2166000.00
end
```

Output would then be sent to the output file *ll.outfile*, containing the below longitude and latitude coordinate values:

166:02:25.645137W 70:27:46.615528S
166:00:53.237056W 70:27:59.692673S
166:00:27.23258W 70:27:57.454281S
166:03:41.428895W 70:28:25.61617S
end

NOTES

Users can add information to the ellipsoid parameter definition file on their systems (located in $GISBASE/etc/ellipse.table) to add spheroids not now among those supported by GRASS.

See *m.ll2u* for a brief discussion of spheroids.

The UTM zone designation determines on what area of the earth a point is found. The same UTM coordinates will be found in each different UTM zone. Look at the marginalia of your source map to determine into which UTM zone your UTM coordinates fall. Although the user can permissibly omit specification of a UTM zone when running this program under a UTM data base LOCATION, it is safer to specify it (see DESCRIPTION, above).

m.u2ll converts the first pair of coordinates on each line of input and leaves anything else on the line alone. If a line begins:

xxxxxx.xx xxxxxxx.xx

then the xxxxxx.xx xxxxxxx.xx UTM coordinate pair is converted to a longitude, latitude pair. Any other information appearing on the line is left alone. If the line doesn't begin with a pair of coordinates in the above format, then the line is left as it is.

FILES

See ellipsoid parameter definition file in $GISBASE/etc/ellipse.table.

SEE ALSO

For australian, clark66, grs80, hayford, international, krasovsky, and wgs72 ellipsoid parameters, see:
John P. Snyder, *Map Projections - A Working Manual*, U.S. Government Printing Office, Washington DC, 1989. U.S. Geological Survey Professional Paper 1395; from Table 1, p.12.

For bessel, airy, everest, and wgs66 ellipsoid parameter values, see:
Thomas O. Seppelin, *The Department of Defense World Geodetic System 1972*, presented at the International Symposium on Problems Related to the Redefinition of North American Geodetic Networks, Fredericton, New Brunswick, Canada in May, 1974; see Table 9, p.35.

For wgs84 parameter values, see:
U.S. Naval Oceanographic Labs.

Also read GRASS User's Reference Manual entries for:
d.labels, d.points, d.sites, d.where, m.datum.shift, m.gc2ll, m.ll2gc, m.ll2u, and *parser*

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

p.chart – Prints the color chart of the currently selected printer.
(GRASS Hardcopy Output Program)

SYNOPSIS

p.chart

DESCRIPTION

This function prints the color chart of the user's hardcopy color printer. The colors in the chart are numbered with the (device-dependent) number associated with each color. This function is useful both for the *p.colors* command as well as for color selection with *p.map*.

The user must select a printer using *p.select* before running this command.

SEE ALSO

d.colors, p.colors, p.map, p.select, r.colors

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

p.colors − Allows the user to develop a color table that associates map categories with user-specified printer colors.
(GRASS Hardcopy Output Program)

SYNOPSIS

p.colors

DESCRIPTION

This function allows the user to modify a color table for a raster map layer, by assigning colors to the categories in the raster map layer based on printer color numbers (instead of red, green, blue percentages).

The user will need to have the printer color chart available to properly select the colors (see manual entry for *p.chart*).

The *p.colors* function allows the user to exercise perfect control over the colors which appear in the hardcopy image.

NOTES

If the user assigns a printer color number outside of the printer's range, *p.colors* will not complain, but will not save the out-of-range printer color number to the raster map layer's color table.

This command modifies the color table associated with a map layer, and will therefore also affect the colors in which a map layer is displayed on the user's graphics monitor. (See map color table files stored under $LOCATION/colr and $LOCATION/colr2.)

SEE ALSO

d.colors, p.chart, p.map, r.colors

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

p.icons – Creates and modifies icons for map display and output.
(GRASS Hardcopy Output Program)

SYNOPSIS

p.icons

DESCRIPTION

This program allows the user to create and maintain icons which are used by the *p.map* and *d.icons* commands to depict sites.

FILES

Icon files created by the user are stored under $LOCATION/icons.

SEE ALSO

d.icons, d.points, d.sites, p.map, s.menu

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

p.labels – Create labels for hardcopy maps.
(GRASS Hardcopy Output Program)

SYNOPSIS

p.labels

DESCRIPTION

This module allows the user to create or modify labels files. These labels files, which are stored in the database, define text information for printing with *p.map* and for graphics display with *d.paint.labels.* Each label has components which determine the text, the location of the text on the image, its size, and the background for the text.

The interface is a screen-oriented input/edit layout. Each label is entered with a single screen. After filling in the required information (described below), the user hits <ESC> to accept the label and start a new one. After the last label has been accepted, the user then hists the <ESC> one more time (on an empty label screen) to exit the module and save the labels.

SCREEN LAYOUT

The screen layout for the labels looks like this:

```
-----------------------------------------------------------------
PAINT LABELS: labelfile          new labels                [1]

TEXT: ____________________________________________   SKIP: no__
      ____________________________________________
      ____________________________________________
      ____________________________________________

LOCATION:  EAST: _________________  OFFSET: ________
           NORTH: _________________  OFFSET: ________
      PLACEMENT: center___________

FONT:              standard__________
TEXT SIZE:         500_____
TEXT COLOR:        black____________  WIDTH: 1_______
HIGHLIGHT COLOR:   none_____________  WIDTH: 0_______

BACKGROUND COLOR:  white____________
BORDER COLOR:      black____________
OPAQUE TO VECTORS: yes_____


              AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
                          (OR <Ctrl-C> TO CANCEL)
-----------------------------------------------------------------
```

The label information that must be provided is:

TEXT: Up to four lines of text. Lines in multiple line labels will appear one above the next.

SKIP: yes| no. If *no*, label will be printed. If *yes*, the label will be retained in the file but not printed.

LOCATION: Determines where the text will be located on the image. The user specifies the easting and

northing, and (optionally) specifies a vertical and horizontal offset (in printer pixels) from the specified easting/northing. (The vertical offset will shift the location to the south if positive, north if negative. The horizontal offset will shift the location east if positive, west if negative.) These offsets are provided to allow finer placement of labels.

PLACEMENT: Determines which part of the label to which the location refers. If placement is unspecified, the label is centered (*center*), by default. Label placement may be specified as:

lower left	(lower left corner of the text)
lower right	(lower right corner of the text)
lower center	(bottom center of the text)
upper left	(upper left corner of the text)
upper right	(upper right corner of the text)
upper center	(top center of the text)
center	(center of the text)

FONT: This specifies the font to use. The following fonts are available:

cyrilc gothgbt gothgrt gothitt greekc greekcs greekp greeks italicc italiccs italict romanc romancs romand romans romant scriptc scripts

The word *standard* can be used to specify the default font (which is *romans*).

TEXT SIZE: This determines the size of the letters. The size specifies the vertical height of the letters in meters on the ground. Thus text will grow or shrink depending on the scale at which the map is drawn. The default text size, if none is specified, is *500*.

TEXT COLOR: This selects the text color. If unspecified, the label's text is drawn in *black*, by default. The text color can be specified in one of four ways:

1) By color name:
aqua black blue brown cyan gray green grey indigo magenta orange purple red violet white yellow

2) As red, green, blue percentages. for example: .5 .4 .7
(This form is not supported by *d.paint.labels*.)

3) By printer color number to get the exact printer color.
(This form is not supported by *d.paint.labels*.)

4) Specify *none* to suppress the lettering.

WIDTH: This determines the line thickness of the letters. The normal text width should be set to 1. Larger numbers can be used to simulate bold face. (*d.paint.labels* ignores this value and always uses 1. *1* is also the default width to which the width is set by *paint*, if none is specified by the user.)

HIGHLIGHT COLOR: The text can be highlighted in another color so that it appears to be in two colors. The text is drawn first in this color at a wider line width, and then redrawn in the text color at the regular line width. No highlight color (*none*) is used, by default, if unspecified by the user. To specify use of no highlight color, specify *none*. (See TEXT COLOR above for a list of permissible color names.)

HIGHLIGHT WIDTH: Specifies how far from the text lines (in units of pixels) the highlight color should extend. The default highlight width is set to *0* (i.e., no highlight color).

BACKGROUND COLOR: Text may be boxed in a solid color by specifying a background color. Specify *none* for no background. The default background color setting, if unspecified by the user, is *white*. (See TEXT COLOR above for a list of permissible color names).

BORDER COLOR: Select a color for the border around the background. Specify *none* to suppress the border. The default border color used, if unspecified, is *black*. (See TEXT COLOR above for a list of permissible color names).

OPAQUE TO VECTORS: *yes*| *no*. This field only has meaning if a background color is selected. *yes* will prevent vector lines from entering the background. *no* will allow vector lines to enter the background. The default setting, if unspecified by the user, is *yes*.

SEE ALSO

p.map, p.select, p.icons, d.paint.labels

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

p.map – Hardcopy color map output utility.
(GRASS Hardcopy Output Program)

SYNOPSIS

p.map
p.map help
p.map [**input**=*name*] [**scale**=*mapscale*]

DESCRIPTION

p.map produces hardcopy color map products on your system's color output device. Output can include a raster map, any number of vector overlays, site data, text labels, and other spatial data.

This program has 2 distincts modes of operation. The command-line mode requires the user to prepare a file of mapping instructions describing the various spatial and textual information to be printed prior to running *p.map*. The interactive mode (i.e., no command-line arguments) will prompt the user for items to be mapped and does not require the user to prepare a file of instructions.

The command-line parameters are:

input=*name* File containing mapping instructions. (or enter **input=-** to enter from keyboard). These instructions are described in detail below.

scale=*mapscale* Scale of the output map, e.g. 1:25000
Default: 1panel

This parameter is provided as a convenience. It is identical to the *scale* mapping instruction described below.

Note: the user must select an output device using *p.select* before running *p.map*. Also, the *preview* device can be selected to view the output from *p.map* on the graphics monitor instead of sending it to a paper printer.

MAPPING INSTRUCTIONS

The mapping instructions allow the user to specify various spatial data to be plotted. These instructions are normally prepared in a regular text file using a system editor. Some instructions are single line instructions while others are multiple line. Multiple line instructions consist of the main instruction followed by a subsection of one or more additional instructions.

colormode

Selects the appropriate method to color raster map layers and images.

USAGE: **colormode** approx| best

There are two methods: *approximate* and *best.* From a user perspective, *approximate* can be used for raster map layers with few categories, such as soils, and *best* should be used for images like LANDSAT images or NHAP photos, or maps with many categories. The *approximate* mode treats each pixel independently, giving it the printer color that best approximates the true color. The *best* mode "blends" colors from pixel to pixel using a dithering technique to simulate more colors than the printer can actually print. The default, if unspecified, is *best.*

This example would select the *approximate* colormode. The assumption is that the raster map layer being printed is has few colors or that the colors would not look good dithered.

EXAMPLE: **colormode** approx

colortable

Includes the color table for the raster map layer in the legend below the map

USAGE: **colortable** [y| n]

The color table will display the colors for each raster map layer category value and the category label. To get a color table, you must have previously requested a raster map layer. Omitting the **colortable** instruction would result in no color table. **Note**: Be careful about asking for color tables for raster map layers which have many categories, such as elevation. This could result in the printing of an extremely long color table!!!!!

This example would print a color table as part of the legend to the map.
EXAMPLE: **colortable** y

comments

Prints comments beneath the map . You may submit comment text line by line during *p.map* execution or a via a prepared comments file.

USAGE: **comments** [commentfile]
comments
end

This example prints the comment "This is a comment" in the legend below the map.
EXAMPLE: **comments**
This is a comment.
end

This example prints whatever is in the file *veg.comments* in the legend below the map.
EXAMPLE: **raster** vegetation
comments veg.comments
end

Presumably, the file *veg.comments* contain comments pertaining to the raster map layer *vegetation*, such as "This map was created by classifying a LANDSAT TM image".

defpat

Defines area fill patterns to be used in the *setpat* instruction.

USAGE: **defpat** name
pattern
color # color
end

The pattern is given a name on the *defpat* instruction line. The pattern which follows is composed of a sequence of numbers 0-9 (and blanks, which are equivalent to 0). The blanks or zeros indicate holes in the pattern where the normal category color would show thru. The other number 1-9 indicate pattern pixels and can be assigned any color. The default color for all the digits will be black unless specified with the *color*

instruction. The color option will begin by entering the word *color* followed by one of the digits (1-9) in the pattern, followed by one of the NAMED COLORS. This should be repeated for each of the digits specified to avoid using black. The instruction *end* terminates the pattern definition. Of course, the user can define more patterns by entering more *defpat* instructions.

NOTE: Do NOT indent the pattern. Leading blank spaces will be interpreted as 0's.

This example creates a black horizontal line pattern.

EXAMPLE: **defpat** horiz
1
0
0
0
color 1 black
end

This example creates a green vertical line pattern.

EXAMPLE: **defpat** vert
1000
color 1 green
end

This example creates a red diagonal line pattern.

EXAMPLE: **defpat** diag
00001
0001
001
01
1
color 1 red
end

This example creates a two-toned tree pattern with orange trunks and green leaves.

EXAMPLE: **defpat** tree
2
222
22122
22 1 22
1

2
222
22122
22 1 22
1
color 1 orange
color 2 black
end

grid

Overlays a coordinate grid onto the output map.

USAGE: **grid** spacing
color color
numbers # [color]
end

The **spacing** of the grid is given (in the geographic coordinate system units) on the main instruction line. The subsection instructions allow the user to specify the **color** of the grid lines, whether coordinate **numbers** should appear on the grid lines, and if they should appear every grid line (1), every other grid line (2), etc., and what color the numbers should be. The defaults are black grid lines, unnumbered.

This example would overlay a green grid with a spacing of 10000 meters (for a metered database, like UTM) onto the output map. Alternate grid lines would be numbered with red numbers.

EXAMPLE: **grid** 10000
color green
numbers 2 red
end

labels

Selects a labels file for output (see manual entry for ***p.labels).***

USAGE: **labels** labelfile| ***list***

This example would paint labels from the labels file called *town.names*. Presumably, these labels would indicate the names of towns on the map.

EXAMPLE: **labels** town.names

line

Draws lines on the output map.

USAGE: **line** east north east north
line x% y% x% y%
color color
width #
masked [y| n]
end

The beginning and ending points of the line are entered on the main instruction. These points can be defined either by map coordinates or by using percentages of the geographic region. The user may also specify line **color, width** in pixels, and if the line is to be **masked** by the current mask. (See manual entry for *r.mask* for more information on the mask.)

This example would draw a yellow line from the point x=10% y=80% to the point x=30% y=70%. This line would be 2 pixels wide and would appear even if there is a mask.

EXAMPLE: **line** 10% 80% 30% 70%
color yellow
width 2
masked n
end

Of course, multiple lines may be drawn with multiple *line* instructions.

outline

Outlines the areas of a raster map layer with a specified color.

USAGE: **outline**
color color
end

Distinct areas of the raster map will be separated from each other visually by drawing a border (or outline) in the specified **color** (default: black). Note: it is important the user enter the instruction **end** even if a color is not chosen. (It is hoped that in the future the outline of a different raster map layer other than the one currently being painted may be placed on the map.)

This example would outline the category areas of the *soils* raster map layer in grey.

EXAMPLE: **raster** soils
outline
color grey
end

point

Places additional points or icons on the output map.

USAGE: **point** east north
point x% y%
color color
icon iconfile| *list*
size #
masked [y| n]
end

The point location is entered in the main instruction line by giving either the map coordinates or by using percentages of the geographic region. The user may also specify the point **color,** the **icon** file to be used to represent the point location (see the the manual entry for *p.icons),* the **size** of the icon in integer multiples of the pattern in the icon file, and whether the point is to be **masked** by the current mask. (See manual entry for *r.mask* for more information on the mask.)

This example would place a purple diamond (from icon file *diamond*) at the point (E456000 N7890000). This diamond would be the same size it is in the diamond icon file and would not be masked by the current mask.

EXAMPLE: **point** 456000 7890000
color purple
icon diamond
size 1
masked n
end

Of course, multiple points may be drawn with multiple *point* instructions.

raster

Selects a raster map layer for output.

USAGE: **raster** mapname| *list*

For each *p.map* run, only one raster map layer can be requested. If no raster map layer is requested, a completely white map will be produced. It can be useful to select no raster map layer in order to provide a white background for vector images.

This example would paint a map of the raster map layer *soils*.
EXAMPLE: **raster** soils

read

Provides *p.map* with a previously prepared input stream.

USAGE: **read** previously prepared UNIX file

Mapping instructions can be placed into a file and read into *p.map.*

Note: *p.map* will not search for this file. The user must be in the correct directory or specify the full path on the **read** instruction. (Note to /bin/csh users: ~ won't work with this instruction).

This example reads the UNIX file *pmap.roads* into *p.map*. This file may contain all the *p.map* instructions for placing the vector map layer *roads* onto the output map.

EXAMPLE: **read** pmap.roads

The user may have created this file because this vector map layer is particularly useful for many *p.map* outputs. By using the **read** option, the user need not enter all the input for the **vector** instruction, but simply **read** the previously prepared file with the correct instructions.

region

Places the outline of a smaller geographic region on the output.

USAGE: **region** regionfile| *list*
color color
width #
end

Geographic region settings are created and saved using *g.region*. The *p.map region* option can be used to show an outline of a smaller region which was printed on a separate run of *p.map* on other user-created maps.

The user can specify the **color** and the **width** (in pixel units) of the outline. The default is a black border of one pixel width.

This example would place a white outline, 2 pixels wide, of the geographic region called *fire.zones* onto the output map. This geographic region would have been created and saved using *g.region*.

EXAMPLE: **region** fire.zones
color white

width 2
end

scale

Selects a scale for the output map.

USAGE: **scale** *scale*

The scale can be selected either as:

a relative ratio, e.g. 1:25000;

an absolute width of the printed map, e.g. 10 inches;

the number of printed paper panels, e.g. 3 panels;

the number of miles per inch, e.g. 1 inch equals 4 miles.

This example would set the scale of the map to 1 unit = 25000 units.
EXAMPLE: **scale** 1:25000

setcolor

Overrides the color assigned to one or more categories of the raster map layer.

USAGE: **setcolor** cat(s) color

This example would set the color for categories 2,5 and 8 of the raster map layer *watersheds* to white and category 10 to green. (**NOTE**: no spaces are inserted between the category values.)

EXAMPLE: **raster** watersheds
setcolor 2,5,8 white
setcolor 10 green

Of course, *setcolor* can be requested more than once to override the default color for additional categories. More than one category can be changed for each request by listing all the category values separated by commas (but with no spaces).

setpat

Assigns a (previously defined) pattern on a raster map layer category.

USAGE: **setpat** cat name
or **setpat** *builtin*
or **setpat** *all*

The user can choose to use: the name of a specific pattern created using **defpat** (see above); the patterns built into *p.map*; or all the patterns the user may have created.

This example assigns the vertical pattern created using **defpat** (see example in **defpat** above) to category 3 of the raster map layer *vegetation* and the tree pattern (see example in **defpat** above) to category 10.
EXAMPLE: **raster** veg
setpat 3 vert
setpat 10 tree

This example reads a previously prepared UNIX file *horiz.pat* with the correct **defpat** instructions for creating a black horizontal pattern (see example in **defpat** above) and assigns that pattern to category 5 of the raster map layer *soils* via the **setpat** instruction.

EXAMPLE: **raster** soils
read horiz.pat
setpat 5 horiz

To select the builtin patterns:

EXAMPLE: **raster** soils
setpat builtin

To select individual builtin patterns:

EXAMPLE: **raster** soils
setpat 5 #1
setpat 10 #2

sites

Selects sites data to be placed on the output map (see manual entry for *s.menu*).

USAGE: **sites** sitemap| *list*
color color
icon iconfile| *list*
size #
desc [y| n]
end

The user may specify the the **color** of the sites (see section on NAMED COLORS below); the **icon** to be used to represent the presence of a site (see the manual entry for *p.icons*); the **size** of the icon (number of times larger than the size it is in the icon file); and whether or not the **desc**ription associated with each site is also to be printed.

This example would paint a sites map with blue windmills (from an icon file created by the user using the *p.icons* GRASS command) placed at all windmill locations (from a sites list). These windmills would be two times larger than the size of the icon in the icon file and have descriptions from the sites list file printed beside them.

EXAMPLE: **sites** windmills
color blue
icon windmill
size 2
desc y
end

text

Places text on the map.

USAGE: **text** east north text
text x% y% text
font fontname
color color| none
width #
hcolor color| none
hwidth #
background color| none

border color| none
size #
ref reference point
xoffset #
yoffset #
opaque [y| n]
end

The user specifies where the text will be placed by providing map coordinates or percentages of the geographic region map. The text follows these coordinates on the same instruction line. More than one line of text can be specified by notating the end of a line with **\n** (e.g. USA \nCERL).

The user can then specify various text features:

font: cyrilc gothgbt gothgrt gothitt greekc greekcs greekp greeks italicc italiccs italict romanc romancs romand romans romant scriptc scripts (The default font is romans);

color (see NAMED COLORS);

width of the lines used to draw the text (to make thicker letters);

size as the vertical height of the letters in meters on the ground (text size will grow or shrink depending on the scale at which the map is painted);

the highlight color (**hcolor**) and the width of the highlight color (**hwidth**);

the text-enclosing-box **background** color; the text box **border** color;

ref. This reference point specifies the text handle - what part of the text should be placed on the location specified by the map coordinates. Reference points can refer to: [lower| upper| center] [left| right| center] of the text to be printed;

yoffset, which provides finer placement of text by shifting the text a vertical distance in pixels from the specified north. The vertical offset will shift the location to the south if positive, north if negative;

xoffset, which shifts the text a horizontal distance in pixels from the specified east The horizontal offset will shift the location east if positive, west if negative;

whether or not the text should be **opaque** to vectors. Entering **no** to the opaque option will allow the user to see any vectors which go through the text's background box. Otherwise, they will end at the box's edge.

This example would place the text *SPEARFISH LAND COVER* at the coordinates E650000 N7365000. The text would be a total of 3 pixels wide (2 pixels of red text and 1 pixel black highlight), have a white background enclosed in a red box, and be 500 meters in size. The lower right corner of the text would be centered over the coordinates provided. All vectors on the map would stop at the border of this text.

EXAMPLE: **text** 650000 7365000 SPEARFISH LAND COVER
font romand
color red
width 2
hcolor black
hwidth 1

background white
border red
size 500
ref lower left
opaque y
end

vector

Selects a vector map layer for output.

USAGE: **vector** vectormap| *list*
color [#] color
width #
hcolor color
hwidth #
masked [y| n]
style 0-9
end

The user can specify the **color** of the vectors; the **width** of the vectors lines in pixels; the highlight color (**hcolor**) for the vector lines; the width of the highlight color (**hwidth**) in pixels; whether or not the raster map layer is to be **masked** by the current mask (see manual entry *r.mask* for more information on the mask); and the line **style**. The line style allows the vectors to be dashed in different patterns and colors. This is done by typing a series of numbers (0-9) in a desired sequence or pattern. Colors for the numbers (1-9) can be assigned using the **color** instruction. Blanks and non-digit characters are recognized as 0's. Using 0 would allow the colors of the raster map layer (or the background color if no raster map layer was selected) to show through.

This example would paint a map of the vector map layer named *streams*. These streams would be a total of 3 pixels wide (the inner two pixels blue and the outer highlight pixel white). The map would not show streams outside of the current mask.

EXAMPLE: **vector** streams
color blue
width 2
hcolor white
hwidth 1
masked y
end

This example would paint a map of the vector map layer *roads*. These roads would be 2 pixels wide and would be dashed blank-black-red (the blank areas would show what lies under the roads). This map would show roads inside and outside of the current mask.

EXAMPLE: **vector** roads
width 2
style 001122
color 1 black
color 2 red
masked n
end

verbose

Changes the amount of talking *p.map* will do.

USAGE: **verbose** [0| 1| 2]

A higher value implies more chatter. The default is 2. This example sets the amount of chatter to a minimum.

EXAMPLE: **verbose** 0

end

Terminates input and begin painting the map.

USAGE: **end**

NAMED COLORS

The following are the colors that are accepted by *p.map*:

aqua black blue brown cyan gray green grey indigo magenta orange purple red violet white yellow

ICONS VS. PATTERNS

Icons and patterns as used in *p.map* are not the same thing. Patterns are defined and are normally used to cover those extended areas covered by a raster map layer category. A pattern will repeat above, below and adjacent to itself. Icons are used to represent a single point.

Patterns are supported directly within *p.map* using the *defpat* instruction, while icons must be created using the *p.icons* program.

EXAMPLE p.map INPUT FILE

The following is an example of a *p.map* script file. The file has been name *spear.soils*. For the purposes of illustration, the file is in two columns. This script file can be entered at the command line:

p.map input=spear.soils

```
                          (cont.)
raster soils              defpat diag
vector streams            000001
  color blue              00001
  width 2                 0001
  hcolor white            001
  hwidth 1                01
  masked y                1
  end                       color 1 red
vector roads                end
  width 2                 setpat 4 diag
  style 001122            text 608000.00 3476004.73 SPEARFISH SOILS MAP
  color 1 black             color red
  color 2 red               width 2
  masked n                  hcolor black
  end                       hwidth 1
labels town.names           background white
region subregion            border red
  color white               size 500
  width 2                   ref lower left
  end                       opaque y
grid 10000                  end
  color green             line 606969.73 3423092.91 616969.73 3423092.91
  numbers 2 red             color yellow
  end                       width 2
outline                     opaque yes
  color black               end
  end                     point 40% 60%
colortable y                color purple
comments                    icon diamond
  This is a comment         size 2
  end                       masked n
scale 1:25000   end
setcolor 6,8,9 white      end
setcolor 10 green
```

INTERACTIVE MODE

If the user simply enter *p.map* without arguments, then a simple prompting session occurs. Some, but not all of the non-interactive requests are available at this level.

SEE ALSO

p.chart, p.icons, p.labels, p.select

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

p.map.new – Color map output utility.
(GRASS Hardcopy Output Program)

SYNOPSIS

p.map.new
p.map.new [**input**= *name*] [**scale**= *mapscale*]

DESCRIPTION

The *p.map.new* command produces color maps for output on a color hardcopy device or a graphics monitor. Output can include a raster map, any number of vector overlays, site data, text labels, and other map elements.

This command has three modes of operation. The command-line mode requires a previously prepared file of mapping instructions describing the map elements to be printed. The interactive mode (i.e., no command-line arguments) will prompt the user for items to be mapped and does not require the user to prepare a file of instructions. The keyboard mode is started by entering a hyphen (-) for the **input** parameter. The *p.map.new* instructions would then be entered via the keyboard.

The command-line parameters are:

input= File containing mapping instructions (or enter **input= -** to enter instructions from the keyboard). These instructions are described in detail below.

scale= Scale of the output map, e.g. 1:25000
Default: 1 panel
The options for this parameter are identical to the *scale* mapping instruction described below. If a *scale* instruction is present in an input file, it is superseded by the command-line **scale** parameter.

An output device can be selected using *p.select* before running *p.map.new*. Valid devices include on-line hardcopy devices, plus *preview, preview2,* and *ppm*. See manual entry for *p.select*.

The current geographic region determines the area that is mapped using *p.map.new.*

NON-INTERACTIVE MAPPING INSTRUCTIONS

Mapping instructions allow the user to specify various map elements to be plotted. These instructions are normally prepared in an ASCII text file using a system editor. All of the listed mapping instructions are usable in a prepared file in the command-line mode. Not all of them are available in the interactive and keyboard modes.

Some instructions are single line instructions while others are multiple line. Multiple-line instructions consist of the main instruction followed by a subsection of one or more additional instructions. All multiple-line instructions must be completed by the **end** terminator.

Some instructions, such as those using data layers, icons, or labels, access files via the current mapset search path.

barscale

Places a barscale on the output map.

USAGE: **barscale** east north
barscale x% y%
unit *ft*| *mi*| *m*| *km*
length #
interval #
style *dash*| *tick*
width #
color color
textsize #
textcolor color| *none*
textfont font
background color| *none*
border color| *none*
end

The location of the zero point of the scale bar is entered on the first instruction line. The location can be defined either by map coordinates or by percentages of the map area, where 0% 0% is the lower left corner of the map.

The user specifies the barscale **unit** of measurement, the total **length** using that unit, and the length of one **interval** (a smaller length evenly divisible into the total length).

The **style** of the scale bar can be specified. The *dash* style has solid lines representing each interval, separated by gaps. The *tick* style has a solid total length with vertical ticks marking each interval.

The user can also specify the **width** of the bar in pixels, its **color** (see VALID COLORS NAMES), the **textcolor**, **textsize** in geographic units, **textfont** (see VALID FONT NAMES), **background** color, and **border** color.

The **barscale** instruction set must be completed with the **end** terminator.

This example would result in a scale bar representing two kilometers. Vertical ticks would be placed at the scale origin, the mid-point, and at the end. The black bar and its accompanying black text would overlay a white box trimmed be a red border.

EXAMPLE: **barscale** 605000 4915000
unit km
length 2
interval 1
style tick
width 2
color black
textcolor black
textsize 150
background white
border red
end

colormode

Selects the method to portray the colors of the raster map layer or image.

USAGE: **colormode** *approx*| *best*

There are two options for **colormode**: *approx* and *best*. The *approx* option should be used for raster map layers with few categories, and *best* should be used for images like LANDSAT images or NHAP photos, or maps with very many categories. The *approx* mode treats each pixel independently, giving it the printer color that best approximates the true color. The *best* mode "blends" colors from pixel to pixel using a dithering technique to simulate more colors than the printer can actually print. If unspecified, the default is *best.*

This example would select the *approx* colormode. The assumption is that the raster map layer being printed has few colors or that the colors would not look good dithered.

EXAMPLE: **colormode** approx

colortable

Includes the color table for the raster map layer in the area below the map on hardcopy output.

USAGE: **colortable** [*y*| *n*]

The color table will display the colors for each raster map layer category and the category value. The **colortable** instruction can not precede the **raster** instruction in the *p.map.new* input. The color table is not shown when the output device is the color monitor.

The user should be careful about asking for color tables for raster map layers that have very many categories, such as an elevation layer. This could result in the printing of an extremely long and generally useless color table!

This example would print a color table below the data area of the map.

EXAMPLE: **colortable** y

comments

Prints comments beneath the map on hardcopy output.

USAGE: **comments** [commentfile]
comments
end

Comment text can be entered in the *p.map.new* file or from a separate, previously prepared file. Comments are not shown when the output device is the color monitor. The **comments** instruction set must be completed by the **end** terminator.

This example prints the comment "This is a comment" below the data area on the the map.

EXAMPLE: **comments**
This is a comment.
end

This example prints the text in a file called "veg.comments" in the current directory.

EXAMPLE: **raster** vegetation
comments veg.comments
end

defpat

Defines an area fill pattern to be used in **setpat** instructions.

USAGE: **defpat** name
pattern
color # color
end

An area fill pattern is given a name on the **defpat** instruction line. This name can then be used in subsequent *setpat* instructions. The **defpat** instruction can be used more than once to specify multiple patterns.

The specified pattern is composed of a sequence of numbers (0-9, and blanks, which are equivalent to 0) on one or more lines. The zeros and blanks indicate areas in the pattern where the normal category colors are visible. The other digits, 1-9, indicate pattern pixels and can be assigned any valid color.

The **color** option specifies a non-zero digit in the pattern, followed by a valid color name. It can be repeated for each of the non-zero digits in the pattern. The default color for all non-zero digits is black unless specified with the **color** option.

The **defpat** instruction set must be completed by the **end** terminator.

In the *p.map.new* input, the **defpat** instruction must precede any **setpat** instruction using the specified pattern.

Note: Indented pattern specifications will be interpreted as having leading blanks.

This example creates a black horizontal line pattern called "horiz". Each black line in the pattern would be one pixel wide and would be three pixels from neighboring lines.

EXAMPLE: **defpat** horiz
1
0
0
0
color 1 black
end

This example creates a green vertical line pattern.

EXAMPLE: **defpat** vert
1000
color 1 green
end

The following example creates a red diagonal line pattern.

EXAMPLE: **defpat** diag
00001
0001
001
01
1
color 1 red
end

This example creates a two-toned tree pattern with orange "trunks" and green "leaves".

EXAMPLE: **defpat** tree
2
222
22122
22 1 22
1

2
222
22122
22 1 22
1
color 1 orange
color 2 black
end

end

Terminates input and begins the painting of the map to the output device.

USAGE: **end**

An **end** instruction completes the entire input to *p.map.new*. It is normally the last line in an input file, but it can be moved forward to eliminate any instructions following its position.

The **end** instruction for the entire input should not be confused with **end** terminators that are required with all multiple-line instruction sets.

grid

Overlays a coordinate grid on the output map.

USAGE: **grid** spacing
pattern *notick*| *tick* # #
masked [*data*| *nodata*| *all*]
style sequence
width #
color color
numbers # [color] [*in*| *out*]
numbersbg color| *none*
numbersize #

end

The spacing of the grid in geographic coordinate system units must be be specified on the first instruction line. The user can specify the overall look of the grid using the **pattern** parameter. The *notick* option is for a complete net of intersecting lines. The *tick* option is for smaller tick marks where grid intervals intersect. The horizontal and vertical lengths, in pixels, must be specified with the *tick* option.

The user can control the areas covered by the grid by using the **masked** parameter. With the *data* option of **masked**, the grid will be seen over all areas of the map's raster layer except the no-data (category 0) areas. With the *nodata* option, the grid will be seen only over the no-data areas. The entire grid is seen with the *all* option to **masked**.

The grid line **style** can be specified using a series of 1's and 0's. The 1's represent the visible dashes and the 0's represent gaps between the dashes. The default is solid lines. The **width** (in pixels) and **color** of the grid lines can also be specified.

The user can control the placement and look of grid label numbers using the **numbers**, **numbersbg**, and **numbersize** parameters. The **numbers** parameter is used to include grid labels, to specify which labels should be shown (where 1 is every grid label, 2 is every other grid label, etc.), and to specify the label color. It is also used to place the labels inside or outside the current region. The background color behind each label is specified by the **numbersbg** parameter. The user controls the grid label size using the **numbersize** parameter, in geographic units.

The **grid** instruction set must be concluded by the **end** parameter.

When used in a metric location, this example would produce grid ticks every 5000 meters. The purple ticks would have "arms" ten pixels long and would be visible over the entire map area. The purple grid numbers would be 350 meters high (to scale), inside the current region map area, and have no background color. A grid label would appear every 5000 meters.

EXAMPLE:
```
grid 5000
  pattern tick 10 10
  masked all
  width 1
  color purple
  numbers 1 purple in
  numbersbg none
  numbersize 350
  end
```

The following example would produce black grid lines every 1000 meters. The lines would be visible only in the areas of category 0, and they would be dashed, with one long dash for every short gap. Every other grid label would be shown, each with a white background.

EXAMPLE:
```
grid 1000
  pattern notick
  masked nodata
  style 11111100
  width 1
  color black
  numbers 2 black in
  numbersbg white
```

numbersize 200
end

labels

Selects a labels file for output.

USAGE: **labels** labelfile| *list*

The **labels** instruction includes previously prepared label specifications. See manual entry for *p.labels* for correct format of the labels file. The labels file must be accessible via the current mapset search path. The *list* option is available in keyboard mode.

This example would paint labels from a labels file called *town.names*.

EXAMPLE: **labels** town.names

legend

Places a user-designed map legend on the output.

USAGE: **legend** east north
legend x% y%
height #
width #
vlen #
textcolor color
textsize #
textwidth #
xspace #
yspace #
background color
border color
beginrast
ramp *value*| *label vertical*| *horizontal*
catnum cat description
end
beginvect
vectname vectormap description
vecttitle vectormap
end
beginsite
sitename sitemap description
end
end

The location of the upper left corner of the legend must be entered on the first instruction line. The location can be defined either by map coordinates or by percentages of the map area, where 0% 0% is the lower left corner of the map.

The user specifies the **height** and **width** of the boxes that will show raster category colors and/or patterns. The length of line segments that will show vector line colors and patterns is specified with the **vlen** parameter. The user controls horizontal spacing between legend symbols and legend text using **xspace**.

The **yspace** parameter is used to control vertical spacing between legend symbols. All of these measurements are in pixels.

The user designs the legend text using the **textcolor**, **textsize**, and **textwidth** parameters. Colors are listed in the VALID COLOR NAMES section of this manual entry. The **textsize** is specified in geographic units. The **textwidth** is specified in pixels.

The user can specify the color for the **background** box containing the entire legend. If a color is chosen, underlying map elements are opaqued. The user can also specify a **border** color for the legend box.

The user specifies the symbols to be included in the legend using the **beginrast**, **beginvect**, and **beginsite** parameters. Each of these parameters starts a subsection of the **legend** instruction that must be completed by an **end** terminator. These should not be confused with the **end** terminator for the entire **legend** instruction set.

If the user simply uses the **beginrast** parameter followed by **end**, all categories of the map's raster layer will be shown in individual boxes, and the legend labels will be the corresponding category names in the layer's *cats* file. The user can include specific categories and optional labels by using one or more **catnum** lines, each including a category number and the accompanying legend text. If the map's raster layer portrays a continuous range of data, a **ramp** in the legend might be appropriate. The **ramp** can be vertical or horizontal, and it's accompanying text can be either the smallest and largest category values, or the *cats* labels associated with the smallest and largest categories. The **beginrast** subsection must be completed with an **end** terminator.

Symbols for vector data on the map can be included in the legend by using the **beginvect** parameter. If the user simply follows **beginvect** with **end**, all vector layers in the map will be included in the legend. The user can include specific vector layers in the legend by using the **vectname** line one or more times, each including a vector layer name and an accompanying description. The vector layer titles as written in *dig_cats* files can be included as the legend text by using the **vecttitle** line one or more times. The **beginvect** subsection must be completed with an **end** terminator.

Site symbols are included in the legend by using one or more **sitename** lines in the **beginsite** subsection. Each line includes the name of the site list and an accompanying description. The **beginsite** subsection must be completed with an **end** terminator.

The entire **legend** instruction set must be completed by an **end** terminator.

In the *p.map.new* input, the **legend** instruction can not precede the instructions for any of the map elements that are to be shown in the legend.

This example would produce a legend with five symbols: a point symbol, the colors and patterns for three raster categories, and a line representing one vector layer, in that order. The background of the legend would be white and surrounded by a red border. All text in the legend would be black.

EXAMPLE:
```
legend 589000 4921200
  height 10
  width 20
  vlen 20
  xspace 10
  yspace 7
  textcolor black
  textsize 250
```

```
textwidth 1
background white
border red
beginsite
 sitename archsites Arch. site
 end
beginrast
 catnum 4 Sandstone
 catnum 5 Limestone
 catnum 6 Shale
 end
beginvect
 vectname roads Road
 end
end
```

The following example would produce a legend with a vertical ramp showing all the colors in the map's raster layer. The labels of the first and last categories would be included.

EXAMPLE:
```
legend 589000 4921200
 height 10
 width 20
 xspace 10
 textcolor black
 textsize 250
 textwidth 1
 background gray
 border black
 beginrast
  ramp label vertical
  end
 end
```

line

Draws a line that is independent of any vector map layer on the output map.

USAGE:
```
line east north east north
line x% y% x% y%
 style sequence
 color [# ] color
 width #
 hcolor color
 hwidth #
 masked [y| n]
 end
```

The beginning and ending points of the line are entered on the main instruction line. These points can be defined either by map coordinates or by using percentages of the geographic region, where 0% 0% is the lower left corner of the map.

The default line style is a continuous, solid line, but the user can specify a dashed line using the **style** parameter. The **style** parameter can contain a sequence of digits (0-9) that represent a colored pattern on

the desired line. Colors can be assigned to each non-zero digit by using the **color** parameter multiple times. If the **color** parameter is used without a specified digit, the named color will be assigned to the entire line. Colors are listed in the VALID COLOR NAMES section in this manual entry.

The user can specify line **width** in pixels. A highlight color can be assigned with **hcolor**, and the highlight's width in pixels can be assigned with **hwidth**. The user can also specify if the line is to be **masked** by the current mask. (See manual entry for *r.mask* for more information on the mask.)

The **line** instruction set must be completed by an **end** terminator.

The **line** instruction can be used more than once to create multiple lines.

This example would draw a blue line from the point x= 10% y= 80% to the point x= 30% y= 70%. The line would be two pixels wide and would appear even if there is a mask.

EXAMPLE: **line** 10% 80% 30% 70%
color blue
width 2
masked n
end

The following example would draw a line with yellow dashes on a black background.

EXAMPLE: **line** 605000 4915000 595300 4918200
style 1111100
color 1 yellow
width 1
hcolor black
hwidth 1
end

outline

Outlines areas of a raster map layer with a specified color.

USAGE: **outline**
color color
end

The **outline** instruction can be used to place a border around all contiguous groups of same-value cells in a raster map layer. A valid color name can be specified with the optional **color** parameter. The default color is black.

The **outline** instruction set must be completed by the **end** terminator, even if the **color** parameter is not used.

The **outline** instruction can not precede a **raster** instruction in a *p.map.new* input file.

The instruction sequence in this example would outline in grey the category areas of a raster map layer called "soils".

EXAMPLE: **raster** soils

outline
color grey
end

point

Places a point symbol on the output map.

USAGE: **point** east north
point x% y%
icon iconfile| *list*
color color
size #
masked [*y*| *n*]
end

The user enters a point symbol location on the main instruction line. The location can be defined either by map coordinates or by percentages of the map area, where 0% 0% is the lower left corner of the map.

The icon to be used can be specified with the **icon** parameter. The user can use any icon in an *icons* directory within the current mapset search path. Icons can be created using *p.icons* or by simply using a system editor. The default icon is a diamond. The *list* option for **icon** is available in keyboard mode.

The user can specify the symbol **color**. Colors are listed in the VALID COLOR NAMES section of this manual entry.

The icon **size** is a positive, floating-point scaling factor of the pattern in the icon file. A size of 1 produces an icon with the same number of pixels (at the output device's resolution) as ASCII characters in the icon file.

The user can also specify whether the point symbol is to be **masked** by the current mask. (See manual entry for *r.mask* for more information on the mask.)

The **point** instruction set must be completed be an **end** terminator. Multiple points may be drawn with multiple **point** instructions.

This example would access an icon file called "box" within the current mapset search path. The red box symbol would be placed at the point E603000, N4921750. The box would have the same number of pixels as characters in the icon file. It would not be masked by the current mask.

EXAMPLE: **point** 603000 4921750
icon box
color red
size 1
masked n
end

raster

Selects a raster map layer for output.

USAGE: **raster** rastermap| *list*

Only one GRASS raster map layer can be specified in a *p.map.new* input file. If no raster map layer is requested, a white background will be produced. The *list* option is available in keyboard mode.

The raster layer must be accessible within the current mapset search path. In a *p.map.new* input file, the **raster** instruction must precede these instructions: **colortable**, **outline**, **setcolor**, and **setpat**. It also must precede any **legend** instruction set that applies to the raster map layer.

This example would paint a map of the raster map layer *soils*.

EXAMPLE: **raster** soils

read

Provides input to *p.map.new* from a previously prepared instruction file.

USAGE: **read** filename

Mapping instructions can be placed in a file and read as input to *p.map.new*. If a certain set of mapping instructions are used in many different maps, they can be placed in one separate file and efficiently accessed by each map's instructions using the **read** instruction.

Note: *p.map.new* will not search for the file to be read. The file must be in the current directory or a full path needs to be specified on the **read** instruction line. (Note to /bin/csh users: the tilde [~] path alias will not work with this instruction).

This example reads the ASCII file "pmap.roads" into *p.map.new*.

EXAMPLE: **read** pmap.roads

region

Places the outline of a geographic region on the output map.

USAGE: **region** regionfile| *list*
style sequence
color [#] color
width #
hcolor color
hwidth #
masked [*y*| *n*]
end

The user can place the outline of a saved geographic region on the map using **region**. The named region file must be in a *windows* directory within the current mapset search path. Geographic region settings can be created and saved using *g.region*. The *list* option is available in keyboard mode.

The default region outline style is a continuous, solid line, but the user can specify a dashed line using the **style** parameter. The **style** parameter can contain a sequence of digits (0-9) that represent a colored pattern on the desired line. Colors can be assigned to each non-zero digit by using the **color** parameter multiple times. If the **color** parameter is used without a specified digit, the named color will be assigned to the entire region outline. Colors are listed in the VALID COLOR NAMES section in this manual entry.

The user can specify the region outline **width** in pixels. A highlight color can be assigned with **hcolor**, and

the highlight's width can be assigned with **hwidth**. The user can also specify if the outline is to be **masked** by the current mask. (See manual entry for *r.mask* for more information on the mask.)

The **region** instruction set must be completed by an **end** terminator.

The **region** instruction can be used more than once to show multiple regions.

This example would produce a white outline, two pixels wide, showing the geographic region called "fire.zones".

EXAMPLE: **region** fire.zones
color white
width 2
end

scale

Specifies the scale of the hardcopy output map.

USAGE: **scale** scale

The scale of the output map can be specified in one of several different forms:

relative ratio e.g. 1:25000

number of geographic units per map unite.g. 1 inch equals 4 miles

absolute width of the printed mape.g. 10 inches

width in number of printed panelse.g. 3 panels

Map inches can be equated with these geographic units: miles, kilometers, and meters. Valid width units are inches, centimeters, and panels. One panel is the single-sheet maximum width available on the hardcopy medium.

The final size of the hardcopy map output is determined by the combination of the specified scale and the current geographic region.

The **scale** instruction does not affect output to the *preview* device. If used, the command-line *scale* parameter overrides the **scale** instruction.

This example would set the scale of the map to one map unit represents 25,000 geographic units.

EXAMPLE: **scale** 1:25000

The following example would specify an output map that would be fifteen centimeters wide.

EXAMPLE: **scale** 15 centimeters

setcolor

Overrides the color assigned to one or more categories of the raster map layer.

USAGE: **setcolor** cat(s) color

The user can assign a desired color to categories in a raster map layer by using **setcolor**. Categories are

specified on the parameter line before a valid color name. One or more category numbers can appear on the parameter line, separated by commas (with no spaces), or in ranges using hyphens.

The **setcolor** instruction can be used more than once for assignment of multiple colors.

In the input *p.map.new* file, the **setcolor** instruction must follow the **raster** instruction.

Colors are listed in the VALID COLOR NAMES section of this manual entry.

In this example, the color for raster map categories 1 through 3, plus category 5, would be set to green, categories 4, 6, and 8 would be set to blue, and category 7 would be set to red, regardless of their assigned colors in the database.

EXAMPLE: **raster** watersheds
setcolor 1-3,5 green
setcolor 4,6,8 blue
setcolor 7 red

setpat

Assigns a previously defined pattern to one or more raster map layer categories.

USAGE: **setpat** cat(s) name
setpat cat(s) **#**number
setpat *all*| *builtin*

The user can assign a pattern to categories in a raster map layer by using **setpat**. Categories are specified on the parameter line before the name of a pattern defined earlier using **defpat**, or before the number signifying a built-in *p.map.new* pattern. One or more category numbers can appear on the parameter line, separated by commas (with no spaces), or in ranges using hyphens.

The built-in patterns are defined in etc/paint/patterns in the compiled GRASS code directory. Each built-in pattern has an assigned number. These numbers can be used following a pound sign (#) on a **setpat** instruction line. By using the *builtin* option, each category in a raster map layer can be assigned the correspondingly numbered builtin pattern.

All raster map categories can be assigned the same defined pattern if the *all* option is used. In this case, only one pattern should be defined within the *p.map.new* mapping instruction file.

The **setpat** instruction can be used more than once for assignment of multiple patterns.

In the input *p.map.new* file, the **setpat** instruction must follow the **raster** instruction, as well as the **defpat** instruction defining the pattern that is used.

This example assigns a pattern called "vert" to categories 3 and 4 of the raster map layer "vegetation" and a pattern called "tree" to category 10.

EXAMPLE: **raster** veg
setpat 3-4 vert
setpat 10 tree

This example reads a previously prepared ASCII file called *horiz.pat* containing **defpat** instructions for creating a black, horizontal pattern called "horiz", and assigns that pattern to category 5 of the raster map

layer "soils".

EXAMPLE: **raster** soils
read horiz.pat
setpat 5 horiz

This example assigns built-in pattern 1 to category 1 of the "soils" raster layer, pattern 2 to category 2, and so on.

EXAMPLE: **raster** soils
setpat builtin

This example assigns built-in pattern 1 to categories 5 through 7 in the "soils" raster map layer, and built-in pattern 2 to categories 10 and 12.

EXAMPLE: **raster** soils
setpat 5-7 # 1
setpat 10,12 # 2

sites

Selects sites data to be placed on the output map.

USAGE: **sites** sitemap| *list*
icon iconfile| *list*
color color
size #
desc [*y*| *n*]
textcolor color
textsize #
end

GRASS sites data can be portrayed on the map using the **sites** instruction. The user can specify the point symbol to be used, and whether labels are to appear next to the symbols. The sites data must be accessible via the current mapset search path. The *list* option is available in keyboard mode.

An icon can be specified with the **icon** parameter. The user can use any icon in an *icons* directory within the current mapset search path. Icons can be created using *p.icons* or by simply using a system editor. The default icon is a diamond. The *list* option for **icon** is available in keyboard mode.

The user can specify the symbol **color**. Colors are listed in the VALID COLOR NAMES section of this manual entry.

The icon **size** is a positive, floating-point scaling factor of the pattern in the icon file. A size of 1 produces an icon with the same number of pixels (at the output device's resolution) as ASCII characters in the icon file.

The **desc** parameter is used to specify whether or not the description of each site in the *site_lists* file is also to be printed. These labels will appear directly to the right of each site symbol. The user controls the color of the labels using the **textcolor** parameters. Valid colors are listed in the named colors section of this manual entry. The label size is specified in geographic units using **textsize**.

The **sites** instruction set must be completed by the **end** terminator.

This instruction can be used more than once to portray multiple site lists.

This example would produce point symbols representing the data in a ***site_lists*** file called "windmills". An icon called "windmill" would be placed at each site location. These symbols would be two times larger than the size of the icon in the icon file (twice as many pixels as there are characters in the icon file). Descriptions from the sites list file would not be produced in this example.

EXAMPLE:
```
sites windmills
  icon windmill
  color blue
  size 2
  desc n
  end
```

text

Places text at a user-specified location on the map.

USAGE:
```
text east north text
text x% y% text
  textfont font
  size #
  color color| none
  width #
  hcolor color| none
  hwidth #
  ref reference_point
  rotation #
  xoffset #
  yoffset #
  background color| none
  opaque [y| n]
  border color| none
  end
```

The user specifies where text will be placed by providing map coordinates or percentages of the map area, where 0% 0% is the lower left corner of the map. The text follows the locational information on the same instruction line. Multiple lines of text can be specified by notating the end of a line with **\n** (e.g., USA \nCERL). Leading blanks an be inserted by preceding the text string with a backslash and the blanks (e.g., text 600000 4920500 \ See \nWall Drug).

The user can control the appearance of the text, its location, and the appearance of its background box.

The user can specify **textfont** (see VALID FONT NAMES in this manual entry), **sizein geographic units, color** (see VALID COLOR NAMES), and **width** in pixels. The user can further control the text appearance by specifying a highlight color (**hcolor**) and the width of the highlight color (**hwidth**).

The text is located at the specified coordinate or percentage pair in relation to a reference point on the text string. This point, specified with the **ref** parameter, has two parts. The first part refers to a vertical location on the text string. Valid choices are *lower*, *center*, and *upper*. The second part refers to a horizontal location: *left*, *center*, and *right*.

The text string can be rotated at the reference point by using the **rotation** parameter. The value specified will be the counter-clockwise rotation in degrees from the horizontal.

The **xoffset** parameter provides finer placement of text by shifting the text a horizontal distance in pixels from the specified easting. The **xoffset** will shift the text location east if positive and west if negative. The **yoffset** parameter shifts the text a vertical distance in pixels from the specified northing. The **yoffset** will shift the location to the south if positive, north if negative.

The user can specify if a **background** box is present, and what color it should be. The user can also specify whether or not the background box is **opaque** to other map elements. The color of the **border** of this box can be specified.

This example would place the text "SPEARFISH LAND COVER" at the coordinates E650000, N7365000. The text would be a total of three pixels wide (one pixel of red text and one pixel of black on each side), have a white background enclosed in a red box, and be 500 meters in size (to scale). The lower left corner of the text would be placed at the coordinates provided. All other map elements would not be seen under the text.

EXAMPLE: **text** 650000 7365000 SPEARFISH LAND COVER
textfont romand
color red
width 1
size 500
ref lower left
hcolor black
hwidth 1
background white
border red
opaque y
end

vector

Selects a vector map layer for output.

USAGE: **vector** vectormap| *list*
style sequence
color [#] color
width #
hcolor color
hwidth #
masked [*y*| *n*]
end

GRASS vector data can be portrayed on the map using the **vector** instruction. The name of the vector layer is specified on the first instruction line. The named vector layer must be accessible via the current mapset search path. The *list* option is available in keyboard mode.

The default vector line style is a continuous, solid line, but the user can specify a dashed line using the **style** parameter. The **style** parameter can contain a sequence of digits (0-9) that represent a colored pattern on the vectors. Colors can be assigned to each non-zero digit by using the **color** parameter multiple times. If the **color** parameter is used without a specified digit, the named color will be assigned to the entire lengths of the vectors. Colors are listed in the VALID COLOR NAMES section in this manual entry.

The user can specify the vector line **width** in pixels. A highlight color can be assigned with **hcolor**, and the highlight's width in pixels can be assigned with **hwidth**. The user can also specify if the vectors are to be **masked** by the current mask. (See manual entry for *r.mask* for more information on the mask.)

The **vector** instruction set must be completed by an **end** terminator.

The **vector** instruction can be used more than once to portray multiple vector data layers.

This example would include a vector map layer named "streams" in the output map. These streams would be a total of four pixels wide (two blue pixels with a white outer highlight one pixel wide on each side). The map would not show streams outside of the current mask.

EXAMPLE: **vector** streams
color blue
width 2
hcolor white
hwidth 1
masked y
end

The following example would portray a vector map layer named "roads". These roads would be two pixels wide and would be dashed blank-black-red (the blank areas would show other map elements under the roads). The roads would be visible inside and outside of the current mask.

EXAMPLE: **vector** roads
width 2
style 001122
color 1 black
color 2 red
masked n
end

verbose

Sets the amount of feedback sent out by *p.map.new*.

USAGE: **verbose** *0| 1| 2*

A higher value set using **verbose** results in more feedback. The default is 2.

This example sets the amount of feedback to a minimum.

EXAMPLE: **verbose** 0

VALID COLOR NAMES

The following are the valid color names in *p.map.new*:

aqua	cyan	indigo	red
black	gray	magenta	violet
blue	green	orange	white
brown	grey	purple	yellow

any integer from 0 through 124, representing printer color numbers (see *p.colors* manual entry)

VALID FONT NAMES

The following are the valid font names in *p.map.new*:

cyrilc	greekcs	italict	romant
gothgbt	greekp	romanc	scriptc
gothgrt	greeks	romancs	scripts
gothitt	italicc	romand	
greekc	italiccs	romans (default)	

ICONS VS. PATTERNS

Icons and patterns as used in *p.map.new* are not the same things. Patterns can only be used to cover the extended areas of a raster map layer category. A pattern will repeat above, below and adjacent to itself. Icons are used to represent single points.

Patterns can be defined directly within *p.map.new* using the **defpat** instruction, while icons are created outside of *p.map.new* using the *p.icons* command or a system editor.

EXAMPLE p.map.new INPUT FILE

The following is an example of a *p.map.new* script file. The file has been named "spear.soils". For the purposes of illustration only, the file is shown in two columns. This script file can be entered at the command line.

p.map.new input= spear.soils

```
                          (cont.)
raster soils              defpat diag
vector streams            000001
  color blue              00001
  width 2                 0001
  hcolor white            001
  hwidth 1                01
  masked y                1
  end                       color 1 red
vector roads                end
  width 2                 setpat 4 diag
  style 001122            text 608000 3476004 SPEARFISH SOILS MAP
  color 1 black             color red
  color 2 red               width 2
  masked n                  hcolor black
  end                       hwidth 1
labels town.names           background white
region subregion            border red
  color white               size 500
  width 2                   ref lower left
  end                       opaque y
grid 10000                  end
  color green             line 606969 3423092 616969 3423092
  numbers 2 red             color yellow
  end                       width 2
outline                     opaque yes
  color black               end
  end                     point 40% 60%
colortable y                color purple
comments                    icon diamond
  This is a comment         size 2
  end                       masked n
scale 1:25000                end
setcolor 6,8,9 white      end
setcolor 10 green
```

INTERACTIVE MODE

If the user enters *p.map.new* on the command-line without arguments, a prompting session occurs. Some, but not all, of the non-interactive requests are available in this mode.

SEE ALSO

g.mapsets, g.region, p.chart, p.colors, p.icons, p.labels, p.ppm, p.select, r.mask

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratories
Joo Joo Chia, U.S. Army Construction Engineering Research Laboratories

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

p.ppm – Reads portable pixmap (ppm) files created by PPM utilities.
(GRASS Paint/Print Program)

SYNOPSIS

p.ppm
p.ppm help
p.ppm [**-f**] [**input**=*name*]

DESCRIPTION

This program, *p.ppm*, reads a user-specified portable pixmap (ppm) file and outputs it to the currently selected printer (see *p.select*). The *input* ppm file should be one that has been created using the PPM utilities developed by Jeff Poskanzer. These utilities can import various image formats (including SUN raster, X Windows pixmaps, and others) into the PPM formats *ppm* (color pixmaps), *pgm* (grey scale maps), and *pbm* (black and white bit-maps).

If the image doesn't fit the output device, it won't get printed. If you want the image printed (but clipped), use the **p.ppm -f** option, or scale the input using *ppmscale*, or rotate the image using *ppmrotate* (if it will fit that way -- otherwise you might have to scale it as well). If the image doesn't fit, *p.ppm* will tell you what scaling value to enter to *ppmscale* that will make it fit.

EXAMPLES

If the user is running GRASS on a SUN machine, the following command could be used to send a monitor screen image to the printer:

screendump | rasttoppm | p.ppm

If you are running suntools, the user might type:

sleep 10; screendump | rasttoppm | ppmrotate 90 | p.ppm

The UNIX *sleep* command allows you time to arrange the frames before the screen dump starts. The *ppmrotate* is usually needed because the SUN screens are wider than they are long (and wider than 1024 pixels - which is the width of most of our printers).

If you are running X, the user might type:

xwd | xwdtoppm | ppmrotate 90 | p.ppm

NOTES

This program only supports the ppm binary format (P6).

Maximum color level is 255. If the ppm file has more color levels, use *ppmcscale* to reduce the number of colors.

No scaling is done. Use *ppmscale* to change image size.

No rotation is done. Use *ppmrotate* to rotate the image.

SEE ALSO

See also:

The PPM utilities *ppmrotate (rotates ppm images), ppmscale (scales ppm images for printing), rasttoppm (converts a SUN raster file* and *xwdtoppm (converts an X Windows dump*

The SUN program *screendump* (dumps the image on the color graphics monitor into a file in SUN raster file format).

The X program *xwd* (dumps the image in an X window into a file in X window dump [xwd] format).

The GRASS programs *d.save, d.savescreen, p.map, p.select,* and *parser*

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

This program uses the PPM utilities developed by Jeff Poskanzer.

NAME

p.select − Selects a device (printer) for GRASS hardcopy output.
(GRASS Hardcopy Output Program)

SYNOPSIS

p.select
p.select help
p.select [**−lpq**] [**painter**=*name*]

DESCRIPTION

p.select allows the user to select the device for GRASS hardcopy output. The user must select a device *before* running the other GRASS hardcopy output functions (e.g., *p.map*, *p.labels*, *p.chart*).

COMMAND LINE OPTIONS

Flags:

-l	List all available painters.
-p	Print name of currently selected painter.
-q	Quietly select painter.

Parameter:

painter=*name*	Name of painter to select. Options: (this will be a list of available hardcopy output devices, and also *preview*, the user's graphics monitor)

INTERACTIVE MODE

If the user runs *p.select* without specifying program arguments on the command line, the program will prompt the user for the name of a hardcopy output device to select.

NOTE

The options available with *p.select* vary from system to system and depend upon which drivers have been compiled.

SEE ALSO

p.chart, p.labels, p.map

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

NAME

GRASS parser interface − A set of GRASS routines that standardizes GRASS commands.

DESCRIPTION

A set of routines in GRASS provides a mechanism for command-line parsing. The routines standardize commands that expect command-line arguments. When a user is familiar with the general form of command-line input as defined by the parser, it simplifies the necessity of remembering (or at least guessing) required command-line arguments for any GRASS command.

The parser behaves in one of three ways:

If no command-line arguments are entered by the user, the parser searches for an interactive version of the command. If it is found, control is passed over to this interactive version. If it is not found, the parser prompts the user for programmer-defined options and flags. This prompting conforms to the same standard for every GRASS command that uses the parser routines.

If command-line arguments are entered but do not include all the options and flags that the programmer has defined as required arguments, three things happen: 1) the parser passes an error message to the user indicating which required options and/or flags were missing from the command-line, 2) then the parser displays a complete usage message for that command, and finally 3) the parser cancels execution of the command.

If all necessary options and flags are entered on the command line by the user, the parser executes the command with the given options and flags.

This page was derived from the GRASS 4.1 Programmer's Manual, Command-Line Parsing, *which offers detailed information about the GRASS parsing routines.*

This is not a command in itself, but an option of the command i.ortho.photo.

NAME

photo.2image - An imagery function that enables you to mark known reference points (fiducial marks, reseau marks, etc.) on imagery group files, and compute the image-to-photo coordinate transformation parameters. The transformation parameters are required for the GRASS Image Processing Program *i.ortho.photo*.
(GRASS Image Processing Program)

SYNOPSIS

Selected from i.ortho.photo

DESCRIPTION

photo.2image is an imagery function that enables you to mark fiducial or reseau points on an image to be ortho-rectified and then computes the image-to-photo coordinate transformation parameters. The coordinates of the fiducials or reseau marks can be choosen for a camera reference file.

During the process of marking reference points with known photo coordinates, you may compute the RMS (root mean square) error for each reference point entered. *photo.2image* does this by calculating the transformation equation (the same one that is calculated in the GRASS program ***i.points***), and then plugging these results into an equation for RMS error.

photo.2image offers a zoom option to locate precisely the fiducial or reseau point to be marked on an image.

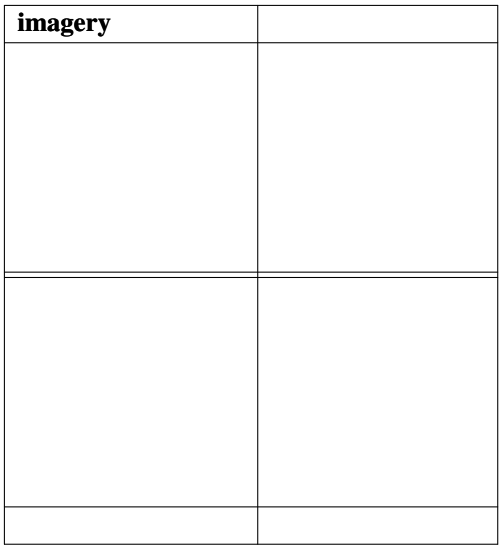
To run *photo.2image*, a graphics monitor is required.

The procedure for marking fiducial or reseau points, entering known photo-coordinates, and analyzing the RMS error is described below.

The terminal screen displays this message:

use mouse now...

The graphics monitor displays the following screen:



A pop-down menu like that shown below will be superimposed on the left half of the screen:

Mapset *imagery*	
gs13.1	**gs21.1**
gs14.1	**gs22.2**

Any single file in the imagery group may be used on which to mark points, and you may mark points on more than one file in the imagery group to accumulate the suggested minimum number of 4-8 fiducial or ressue points. The imagery file selected is displayed in the upper left quadrant of the screen.

CAMERA FILE

The camera reference file may be viewed by placing the mouse cross hairs on the words CAMERA. The camera reference file will be displayed in the lower left quadrant of the display:

```
-------------------------------------------------------------|
|Camera Reference File                                       |
-------------------------------------------------------------|
| CAMERA NAME :      camera name                             |
| CAMERA ID   :      L1234                                   |
| CAMERA CFL  :      153.021 mm.                             |
| CAMERA XP   :      0.050 mm.                               |
| CAMERA XP   :      0.056 mm.                               |
| number of fid.     4                                       |
```

ID	(X PHOTO) mm.	(Y PHOTO) mm.
1	-105.023	110.123
2	107.987	109.834
3	110.965	-104.329
4	-103.932	-110.352

The following menu is displayed at the bottom of the graphics display:

Quit	Zoom	Plot Cell	Camera	Analyze	Input->KEYBOARD	CAMERA FILE

ZOOM

To magnify the displayed file, you must place the mouse cross hairs on the word ZOOM. The following menu will then be displayed at the bottom of the screen:

Cancel	Box	Point	Select type of ZOOM

You have the option of identifying the zoom window either by using the mouse to make a box, or by using the mouse to mark the center of the desired window, and adjusting the magnification factor. The terminal screen will display a mouse button menu to guide you in identifying the window. The section of the image within the zoom window will be displayed in the upper right hand quadrant.

MARKING REFERENCE POINTS

To mark the known reference points (fiducial marks, reseau marks, etc.) on the image, you must place the mouse cross hairs on the corresponding location on the image to be marked and press the left hand button on the mouse. A diamond shaped symbol will be marked on the image.

If you wish to use the camera reference file only as a comparative reference, then the keyboard can be chosen as the means to input photo coordinates corresponding to the marked reference points on the image. This is done by placing the mouse cross hairs on the word KEYBOARD and pressing the left button on the mouse.

The following menu is displayed on the graphics terminal:

Point 1 marked at IMAGE COORDINATES **east: 1023.77** **north: 506.56**	
Enter PHOTO COORDINATES as x and y:	

You then enter the known (x,y) photo coordinates, relative to the perspective center, for the reference point marked on the image. If you do not wish to enter a coordinate, simply hit RETURN to continue; the marked reference point will disappear.

If you select the CAMERA FILE option, then reference points marked on the image will be associated with selected photo coordinates from the displayed camera reference file. In this option, when you mark a point on the image, the following menu is displayed on the graphics terminal:

CANCEL Double click on the fiducial mark to be referenced

If you would like to select the photo coordinates of a displayed reference mark, this can be accomplished by placing the mouse cross hairs on the reference point to be selected and pressing the left button on the mouse twice. After a reference point is selected from the display, you are prompted with "Look ok? (Y/N)". If you respond with no, the reference point is ignored. If you respond with yes, the following is displayed on the terminal:

Point 1 marked at IMAGE COORDINATES **east: 1023.77** **north: 1065.41** **Point 1 referenced at PHOTO COORDINATES** **X: -105.023** **Y: 110.122**	
use mouse now...	

The photo coordinates and the corresponding image coordinates are automatically saved in the photo_points file associated with the imagery group.

ANALYZE

After a number of points have been marked (4 to 8), you can check the RMS error of the points marked on the image. This is done by placing the mouse cross hairs on the word ANALYZE at the bottom of the monitor. An error report resembling the one shown below is superimposed on the monitor:

		error		**image**	**photo**		
#	**row**	**col**	**photo**	**east**	**north**	**x**	**y**
1	**0.0**	**-0.9**	**1.0**	**1048.5**	**-144.8**	**-105.023**	**110.122**
2	**0.4**	**1.0**	**1.3**	**2153.1**	**-567.2**	**107.987**	**109.834**
3	**-1.2**	**-0.5**	**.6**	**1452.8**	**-476.5**	**110.965**	**-104.329**
4	**1.1**	**0.5**	**1.3**	**1034.0**	**-109.2**	**-103.932**	**-110.352**
	overall	rms	error:	4.46			

The following menu then appears at the bottom of the monitor:

DONE	**PRINT**	**FILE**	**Double click on point to be included/excluded**

The RMS error for the image is given under the column titled "error" and subtitled "row" and "col". In the above report, point number 1 is 0.0 rows and -0.9 columns from the predicted location calculated from the transformation equation. The RMS error for the photo coordinates is listed under the heading "photo". This is the RMS error for the x and y coordinates of the photo coordinates but it is presented in the table using one general value. The overall RMS error is displayed at the bottom of the screen in milimeters. Points that create high RMS error are displayed in red on the monitor (represented here in italics).

The location of the point marked on the imagery group file is given under the heading "image" and the subheadings "east" and "north". The location of the point in the photo coordinates is given under the heading "photo" and the subheadings "x" and "y". If you would like to exclude or include a point, this can be accomplished by placing the mouse cross hairs on the point number to be included (if the point is absent) or excluded (if the point is displayed) and pressing the left button on the mouse twice. When a point is excluded, it is not afterwards included in the calculation of the RMS error, or included in the final transformation matrix. However, it can be retrieved within *photo.2image* at any time by double clicking with the mouse as described above.

QUIT

To end the *photo.2image* program place the mouse cross hairs on the word QUIT; the marked reference points (including coordinates) will be saved.

NOTES

A good rule of thumb is to mark at least 4 to 8 points which are evenly distributed over the perimeter of the imagery group file in order to obtain an accurate transformation equation for the rectification process. The RMS error may increase with more points added, but the transformation equation will be more accurate.

An RMS error of less than or equal to approximately one resolution unit (pixel) for the image being rectified is generally considered acceptable.

SEE ALSO

i.ortho.photo[2], photo.camera[2], photo.2target[2], photo.init[2], photo.rectify[2]

AUTHOR

Mike Baba DBA Systems, Inc.

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

This is not a command in itself, but an option to the command i.ortho.photo.

NAME

photo.2target - An imagery function that enables you to mark control points on an image to be ortho-rectified, and then input the coordinates of each control point for determination of ortho-rectification parameters. The ortho-rectification parameters are required as input for the GRASS program *photo.rectify*. *(GRASS Imagery Processing Tool)*

SYNOPSIS

Selected from i.ortho.photo

DESCRIPTION

photo.2target is an imagery function that enables you to mark control points on an image to be ortho-rectified and then input the coordinates of each point for calculation of rectification parameters.

Rectification is the mapping of an image from one coordinate system to another. The geometry of an image extracted into a GRASS LOCATION having an x,y coordinate system is not planimetric. To create a planimetric image, that is, to convert the x,y coordinate system into a standard coordinate system (for example, the UTM coordinate system or the State Plane coordinate system), points from a map having the standard coordinates must be associated with the same points on the image to be rectified.

The ortho-rectification parameters are computed in two phases. The first phase computes a transformation matrix between image (row,col) coordinates and photo (x,y) coordinates relative to the perspective center. The transformation matrix is computed explicitly in the option *photo.2image*.

The second phase, *photo.2target*, enables you to mark control points on the image and then input the standard coordinates (Easting,Northing, and elevation) to determine the parameters for a three dimensional projective transformation.

photo.init may be run before running *photo.2target* to modify the initial camera exposure station parameters, and/or modify the standard deviation of these parameters. *photo.init* is not required but generally is helpful.

During the process of marking control points and entering standard coordinates, you may compute the RMS (root mean square) error for each control point entered. photo.2target does this by calculating the transformation equation, and then plugging these results into an equation for RMS error.

photo.2target offers a zoom option to locate precisely the point to be marked on an image. This program also offers you the option of acquiring standard coordinates for a marked point from a map layer in the target database, or from a digitizer.

To run *photo.2target*, a graphics monitor is required.

The procedure for marking points, entering coordinates, and calculating RMS error is described below.

The terminal screen displays the following message:

use mouse now...

The graphics monitor displays the following screen:

imagery *filename* **(mag)**	**target** *filename* **(mag)**
QUIT ZOOM PLOTCELL ANALYZE	

A pop-down menu like that shown below will be superimposed on the left half of the screen:

Double click on cell file to be plotted **Double click here to cancel**

Mapset *demo*	
gs13.1	**gs14.1**
gs21.1	**gs22.2**

Any single file in the imagery group may be used to mark points, and you can mark points on more than one file in the imagery group to accumulate the 12 points suggested minimum. Any file in the imagery group can be rectified (using *photo.rectify*) based on the rectification parameters computed from these control points.

The imagery file you select is displayed in the upper left quadrant of the screen.

ZOOM

To magnify the displayed file, you must place the mouse cross hairs on the word ZOOM. The following menu will then be displayed at the bottom of the screen:

Cancel	**Box**	**Point**	**Select type of ZOOM**

You may identifying the zoom window either by using the mouse to make a box, or by using the mouse to mark the center of the window and entering a magnification factor. The terminal screen will display a mouse button menu to guide you in identifying the window.

MARKING POINTS

To mark the points on the image that correspond to the points on a standard coordinate system map, you must place the mouse cross hairs on the corresponding location on the image to be marked and press the left hand button on the mouse. A diamond shaped symbol will be marked on the image. The terminal will display the following menu:

Point 1 marked at IMAGE COORDINATES **IMAGE X: 1023.77** **IMAGE Y: -164.41**	
Enter CONTROL COORDINATES as east,north,elevation:	

You then enter the easting, northing, and elevation for the point marked on the image. If you wish not to enter a coordinate, simply hit RETURN to return control to the mouse; the marked point then disappears.

PLOT CELL

In addition to acquiring control points from a standard map, you have the option of acquiring the points from a cell-map in the target database. The database map is displayed by placing the mouse cross hairs on the words PLOT CELL. The following line is then displayed at the bottom of the monitor:

Cancel	**Indicate which side should be plotted**

Which side of the monitor is to be plotted is indicated by placing the mouse cross hairs on the half of the monitor screen that you would like to use, and pressing the left mouse button. The following pop-down menu will be superimposed on the half of the screen that was chosen:

Double click on cell file to be plotted **Double click here to cancel**

Mapset *demo*	
tm.rectified	
tm.classified	
Mapset PERMANENT	
elevation	**geology**
slope	**soils**
aspect	
roads	
streams	
airfields	

After the map is displayed the following message appears at the bottom of the monitor:

input method -->	**keyboard**	**screen**

If you wish to use the plotted map only as a comparative reference, the keyboard can be chosen as the means to input coordinates corresponding to the marked control points. This is done by placing the mouse cross hairs on the word KEYBOARD and pressing the left button on the mouse.

If you select the SCREEN option, points marked on the image will automatically be associated with the coordinates from the corresponding points on the target database map, and a corresponding elevation from the cell-file selected for elevation data. In this option, when you mark a point on the image, the following menu is displayed on the terminal:

Point 5 marked at IMAGE COORDINATES **IMAGE X: 1023.77** **IMAGE Y: -164.41** **Control Point location** **East: 679132.57** **North: 4351080.67** **Elevation: 1010.00**	
use mouse now...	

The coordinates for the target database map are automatically saved as the coordinates corresponding to the marked control point on the image.

ANALYZE

After a number of points have been marked (4 to 7), you can check the RMS error of the points marked on the image. This is done by placing the mouse cross hairs on the word ANALYZE at the bottom of the monitor. An error report resembling that shown below is superimposed on the monitor:

					error			
#	**east**	**north**	**target**	**east**	**north**	**east**	**north**	**elev.**
1	**0.0**	**-0.9**	**1.0**	**48.5**	**4.8**	**79132.5**	**351080.6**	**010.0**
2	**0.4**	**1.0**	**1.3**	**53.1**	**7.2**	**84314.7**	**399001.4**	**239.3**
3	**-1.2**	**-0.5**	**.6**	**52.8**	**6.5**	**67841.4**	**457682.8**	**209.5**
4	**1.1**	**0.5**	**1.3**	**34.0**	**9.2**	**77573.8**	**352626.4**	**432.5**
5	*-2.7*	*14.0*	*14.2*	*48.6*	*-144.9*	*79132.6*	*351080.7*	*985.0*
		overall	**rms**	**error:**	**4.46**			

The following menu then appears at the bottom of the monitor:

DONE	PRINT FILE	Double click on point to be included/excluded

The RMS error for the image is given under the column titled "error" and subtitled "east" and "north". In the above report, point number 1 is 0.0 and -0.9 meters (east and north) from the predicted location calculated from the transformation equation. The RMS error for the target map is listed under the heading "target". This is the RMS error for the east and north coordinates of the target map but it is presented in the table using one general value. The overall RMS error is displayed at the bottom of the screen in meters. Points that create high RMS error are displayed in red on the monitor (represented here in italics).

The image coordinates of the point marked on the imagery group file is given under the heading "image" and the subheadings "east" and "north". The location of the control point in the target database is given under the heading "control" and the subheadings "east","north", and "elev". If you would like to exclude or include a control point, this can be accomplished by placing the mouse cross hairs on the control point number to be included (if the point is absent) or excluded (if the point is displayed) and pressing the left button on the mouse twice. When a point is excluded, it is not afterwards included in the calculation of the RMS error, or included in the final rectification parameters. However, it can be retrieved within *photo.2target* at any time by double clicking with the mouse as described above.

QUIT

To end the *photo.2target* program place the mouse cross hairs on the word QUIT; the marked control points (including coordinates) will be saved.

NOTES

During the course of marking control points and computing the ortho-rectification parameters, a matrix inversion error may occur. This is caused by trying to invert a non-singular normal equation matrix. When this situation arises, the status of all previously selected control points are modified, the control points are excluded. Running *photo.init* for the selected imagery group with accurate camera exposure station parameters should remedy the situation. The excluded control points may again be included as described in the section ANALYZE.

A good rule of thumb is to mark at least 12 to 15 points which are evenly distributed over the entire imagery group file in order to obtain an accurate transformation parameters for the rectification process. The RMS error may increase with more points added, but the transformation parameters will be more accurate over the entire image.

An RMS error of less than or equal to approximately one resolution unit (pixel) for the image being rectified is generally considered acceptable.

SEE ALSO

i.ortho.photo[2], photo.camera[2], photo.2image[2], photo.init[2], photo.rectify[2]

AUTHOR

Mike Baba DBA Systems, Inc.

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

This is not a command in itself, but an option to the command i.ortho.photo.

NAME

photo.camera - An imagery function that creates or modifies entries in a camera reference file.
(GRASS Image Processing Program)

SYNOPSIS

photo.camera option must be selected from GRASS Image Processing Program *i.ortho.photo*.

DESCRIPTION

photo.camera creates or modifies entries in a camera reference file. For ortho-photo rectification, a camera reference file is required for computation of scanned image to photo-coordinate transformation parameters.

The first prompt in the program will ask you for the name of the camera reference file to be created or modified. You may create a new camera reference file by entering a new name, or modify an existing camera reference file by entering the name of an existing camera file.

After entering the camera file name, following menu is displayed:

Please provide the following information

CAMERA NAME: *camera name*_____
CAMERA IDENTIFICATION: *identification*___
CALIBRATED FOCAL LENGTH mm.:_____
POINT OF SYMMETRY (X) mm.:_____
POINT OF SYMMETRY (Y) mm.:_____
MAXIMUM NUMBER OF FIDUCIALS:_____

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

The camera name and identification describe the camera reference file. The calibrated focal length and the point of symmetry are used in computing the photo-to-target transformation parameters. These values should be entered from the camera calibration report (usually available from the photograph supplier). The maximum number of fiducials will determine the number of fiducial or reasuae coordinate pairs to be entered below.

You are then ask to enter the X and Y photo coordinates of each fiducial as follows:

Please provide the following information

Fid#	**FID ID**	**X**	**Y**
1__	**_____**	**0.0___**	**0.0___**
2__	**_____**	**0.0___**	**0.0___**
3__	**_____**	**0.0___**	**0.0___**
4__	**_____**	**0.0___**	**0.0___**
5__	**_____**	**0.0___**	**0.0___**
6__	**_____**	**0.0___**	**0.0___**
7__	**_____**	**0.0___**	**0.0___**
8__	**_____**	**0.0___**	**0.0___**
9__	**_____**	**0.0___**	**0.0___**
10_	**_____**	**0.0___**	**0.0___**

next: end__

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

The input display is repeated until the number of MAXIMUM FIDUCIALS is reached. On this screen you should enter the fiducial or ressuae photo-coordinates as given in the camera calibration report. The X, and Y coordinates are in milli-meters from the principle point.

SEE ALSO

i.ortho.photo[2], photo.2image[2], photo.2target[2], photo.init[2],

AUTHOR

Mike Baba DBA Systems, Inc.

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

This is not a command itself, but an option of the command i.ortho.photo.

NAME

photo.init - An imagery function that creates or modifies the optional entries in a camera initial exposure station file for an imagery group.
(GRASS Image Processing Program)

SYNOPSIS

Selected from i.ortho.photo

DESCRIPTION

photo.init creates or modifies entries in a camera initial exposure station file for imagery group referenced by a sub-block. These entries include: the (XC,YC,ZC) UTM approximate coordinates of the camera exposure station; initial roll, pitch, and yaw angles (in degrees) of the cameras attitude; and the ***a priori*** standard deviations for these parameters. During the imagery program, hoto.rectify, the initial camera exposure station file is used for computation of the ortho-rectification parameters. If no initial camera exposure station file exist, the default values are computed from the control points file created in *photo.2target*.

The following menu is displayed:

Please provide the following information

INITIAL XC: Meters _____
INITIAL YC: Meters _____
INITIAL ZC: Meters _____
INITIAL omega (roll) degrees: _____
INITIAL phi (pitch) degrees: _____
INITIAL kappa (yaw) degrees: _____

Standard Deviation XC: Meters _____
Standard Deviation YC: Meters _____
Standard Deviation ZC: Meters _____
Std. Dev. omega (roll) degrees: _____
Std. Dev. phi (pitch) degrees: _____
Std. Dev. kappa (yaw) degrees: _____

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

The INITIAL values for (XC,YC,ZC) are expressed in UTM coordinates, and represent an approximation for the location of the camera at the time of exposure.

The INITIAL values for (omega,phi,kappa) are expressed in degrees, and represent an approximation for the cameras attitude at the time of exposure.

The standard deviations for (XC,YC,ZC) are expressed in meters, and are used as ***a priori*** values for the standard deviations used in computation of the ortho rectification parameters.

The standard deviations for (omega,phi,kappa) are expressed in degrees, and are used as ***a priori*** values for the standard deviations used in computation of the ortho rectification parameters.

SEE ALSO

i.ortho.photo[2], photo.camera[2], photo.2image[2], photo.2target[2], photo.rectify[2]

AUTHOR

Mike Baba DBA Systems, Inc.

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

This is not a command in itself, but an option of the command i.ortho.photo.

NAME

photo.rectify - An imagery function that rectifies an image by computing a coordinate transformation for each cell in the image using parameters computed by the GRASS programs photo.2image and photo.2target.
(GRASS Image Processing Program)

SYNOPSIS

Selected from i.ortho.photo

DESCRIPTION

photo.rectify rectifies an image by using the image to photo coordinate transformation matrix created by photo.2image and the rectification parameters created by photo.2target. Rectification is the process by which the geometry of an image is made planimetric. This is accomplished by mapping an image from one coordinate system to another. In photo.rectify the parameters computed by photo.2image and photo.2target are used in equations to convert x,y image coordinates to standard map coordinates for each pixel in the image. The result is an image with a standard map coordinate system, compensated for relief distortions and photographic tilt. Upon completion of the program the rectified image is deposited in a previously targeted GRASS LOCATION.

You are asked to select the file(s) within the imagery group to be rectified:

Please select the file(s) to rectify by naming an output file

gs13.1 in demo *gs13.orect**...***
gs14.1 in demo **.....**
gs21.1 in demo **.....**
gs22.1 in demo **.....**

(enter list by any name to get a list of existing cell files)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

More than one file may be rectified at a time. Each file should have a unique output file name.

The next prompt asks you to select one of two windows:

Please select one of the following options
1. Use the current window in the target location
2. Determine the smallest window which covers the image
>

photo.rectify will only rectify that portion of the image that occurs within the chosen window. Only that portion will be relocated in the target database. It is therefore important to check the current window in the target LOCATION if choice number one is selected.

photo.rectify will run in the background and notify you by mail when it is finished. The process may take an hour or more depending on the size of the image, the number files, and the size and resolution of the selected window.

The rectified image will be located in the target LOCATION when the program is completed. The original unrectified files are not modified or removed.

SEE ALSO

i.ortho.photo[2], photo.camera[2], photo.2image[2], photo.2target[2], photo.init[2]

AUTHOR

Mike Baba DBA Systems, Inc.

NOTICE

This program is part of the ***alpha*** section of the GRASS distribution. Unlike the code in the ***main*** section of GRASS, the ***alpha*** code has not yet been fully tested for one release cycle.

NAME

ps.icon − Creates and modifies icons for use with *ps.map*.
(GRASS Hardcopy PostScript Output Program)

SYNOPSIS

ps.icon

DESCRIPTION

This program allows the user to create and maintain icons which are used by *ps.map* to depict sites.

FILES

Icon files created by the user are stored under $LOCATION/ps_icon.

SEE ALSO

ps.map

AUTHOR

Paul Carlson, USDA, SCS, NHQ-CGIS

NOTICE

This program is part of the *alpha* section of the GRASS distribution. Unlike the code in the *main* section of GRASS, the *alpha* code has not yet been fully tested for one release cycle.

NAME

ps.map − Hardcopy PostScript map output utility.
(GRASS Hardcopy PostScript Output Program)

SYNOPSIS

ps.map
ps.map help
ps.map [**r**] [**input**=*name*] [**scale**=*mapscale*] [**copies**=*n*] **output**=*name*

DESCRIPTION

ps.map produces an output file containing a PostScript program to produce hardcopy map products on your system's PostScript output device. Output can include a raster map, any number of vector overlays, site data, text labels, and other spatial data.

This program has 2 distincts modes of operation. The command-line mode requires the user to prepare a file of mapping instructions describing the various spatial and textual information to be printed prior to running *ps.map*. The interactive mode (i.e., no command-line arguments) will prompt the user for items to be mapped and does not require the user to prepare a file of instructions.

The command line flag is:

r Rotate plot 90 degrees.

The command-line parameters are:

input=*name* File containing mapping instructions. (or enter **input=-** to enter from keyboard). These instructions are described in detail below.

scale=*mapscale* Scale of the output map, e.g. 1:25000
Default: 1 panel

This parameter is provided as a convenience. It is identical to the *scale* mapping instruction described below.

output=*name* Name of output the file to contain the PostScript program.

Note: the user must select a PostScript output device using *ps.select* before running *ps.map*.

MAPPING INSTRUCTIONS

The mapping instructions allow the user to specify various spatial data to be plotted. These instructions are normally prepared in a regular text file using a system editor. Some instructions are single line instructions while others are multiple line. Multiple line instructions consist of the main instruction followed by a subsection of one or more additional instructions.

Instructions that may be included in the subsection under several different main instructions are:

where *x y* The top left corner of the bounding box of the item to be plotted is located *x* inches from the left edge of the paper and *y* inches from the top edge of the paper. If *x* is less than or equal to zero, the default horizontal location is used. If *y* is less than or equal to zero, the default vertical location is used.

font *font name* The name of the PostScript font. The default is *Helvetica*.

fontsize *font size* The size of the PostScript font in 1/72 inch. The default is 10.

colortable

Prints the color table for the raster map layer anywhere on the page.

USAGE: **colortable** [y| n]
where x y
width table width
cols table columns
font font name
fontsize font size
color text color
end

The color table will display the colors for each raster map layer category value and the category label. To get a color table, you must have previously requested a raster map layer. The default location for the colortable is immediately below any other map legend information, starting at the left margin. The default text color is black. Omitting the **colortable** instruction would result in no color table. **Note**: Be careful about asking for color tables for raster map layers which have many categories, such as elevation. This could result in the printing of an extremely long color table!!!!!

This example would print a color table immediately below any other map legend information, starting at the left margin.
EXAMPLE: **colortable** y

comments

Prints comments anywhere on the page.

USAGE: **comments** commentfile
where x y
font font name
fontsize font size
color text color
end

The default location is immediately below the last item item printed, starting at the left margin. The default text color is black.

This example prints in blue whatever is in the file *veg.comments* starting at 1.5 inches from the left edge of the page and 7.25 inches from the top of the page, using a 15/72 inch Helvetica Bold font.
EXAMPLE: **raster** vegetation
comments veg.comments
where 1.5 7.25
font Helvetica Bold
fontsize 15
color blue
end

Presumably, the file *veg.comments* contain comments pertaining to the raster map layer *vegetation*, such as "This map was created by classifying a LANDSAT TM image".

copies

Specifies the number of copies to be printed.

USAGE: **copies** n

Each page will be printed n times.

greyrast

Selects a raster map layer for output in shades of grey.

USAGE: **greyrast** mapname| *list*

For each *ps.map* run, only one raster map layer can be requested (using either the *greyrast* or the *raster* instruction).

grid

Overlays a coordinate grid onto the output map.

USAGE: **grid** spacing
color color
numbers # [color]
font font name
fontsize font size
end

The **spacing** of the grid is given (in the geographic coordinate system units) on the main instruction line. The subsection instructions allow the user to specify the **color** of the grid lines, whether coordinate **numbers** should appear on the grid lines, and if they should appear every grid line (1), every other grid line (2), etc., and what color the numbers should be. The defaults are black grid lines, unnumbered.

This example would overlay a green grid with a spacing of 10000 meters (for a metered database, like UTM) onto the output map. Alternate grid lines would be numbered with red numbers.

EXAMPLE: **grid** 10000
color green
numbers 2 red
end

header

Prints the map header above the map.

USAGE: **header**
file header file
font font name
fontsize font size
color text color
end

If the *header* instruction or the *filesub-instruction is absent, the header will consist of the map title and*

location, each centered on the page above the map. The default text color is black.

This example prints (in red) whatever is in the file soils.hdr above the map, using a 20/72 inch Courier font.

EXAMPLE: **header**
file soils.hdr
font Courier
fontsize 20
color red
end

labels

Selects a labels file for output (see manual entry for *p.labels).*

USAGE: **labels** labelfile| *list*
font font name
end

This example would paint labels from the labels file called *town.names*. Presumably, these labels would indicate the names of towns on the map.

EXAMPLE: **labels** town.names

line

Draws lines on the output map.

USAGE: **line** east north east north
line x% y% x% y%
color color
width #
masked [y| n]
end

The beginning and ending points of the line are entered on the main instruction. These points can be defined either by map coordinates or by using percentages of the geographic region. The user may also specify line **color, width** in pixels, and if the line is to be **masked** by the current mask. (See manual entry for *r.mask*