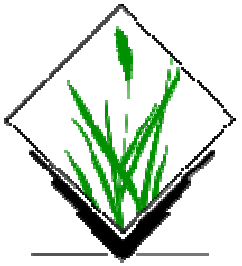
GRASS GIS intro

Geographic Information Systems (GIS)

Sara Lucca
Luana Valentini



What's GRASS

- Free and Open source GIS
- **G**eographic **R**esources **A**nalysis **S**upport **S**ystem
- Portable:
  - GNU/Linux
  - Mac-OSX
  - ...

MS-Windows

- Written in: **C/C++** Makes also use of Tcl/tk, Python
- GRASS main web site: `http://grass.osgeo.org`

What’s GRASS - 2

- **Free** and **Open Source** Software (FOSS):
  - **0**: The freedom to run the program, for any purpose.
  - **1**: The freedom to study how the program works, and adapt it to your needs. Access to the source code is a precondition for this.
  - **2**: The freedom to redistribute copies so you can help your neighbor
  - **3**: The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. Access to the source code is a precondition for this.

Interoperability

- Proj.4: Projections
- OGR/GDAL: data conversion libraries
- Database:
  - PostgreSQL/PostGIS
  - MySQL
  - SQLite
  - ODBC: PostgreSQL, MySQL, ORACLE, MS-Access, MS-SQL Server, ecc.
- QGIS
- R-stats

How to obtain GRASS?

http://grass.osgeo.org/download/index.php

- LINUX:
  - download source code and compile it
  - download binary packages (generic or distribution-oriented)
- WINDOWS:
  - download stable binaries (Cygwin)
  - download testing installer
- Mac OSX:
  - download binary packages
- LiveCDs / LiveDVDs / LiveUSBs

How to obtain GRASS? - 2

- As for LiveUSBs: **slaxGIS**

http://geomatica.como.polimi.it/software/slaxGIS/

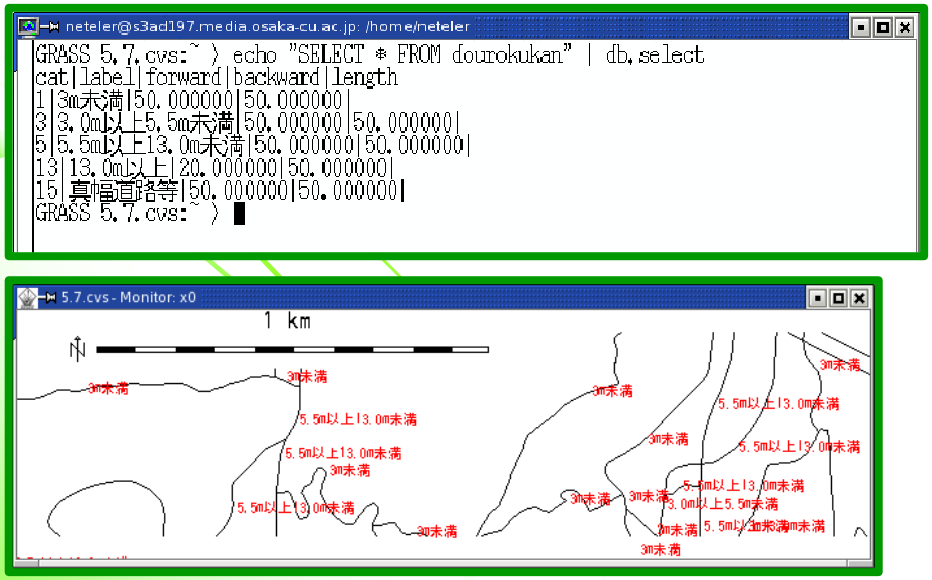
Complete Linux system bootable from a USB flash drive, containing (V090731B):

-grass-6.4.svn_2009_10_31
- MapServer 5.4.2
- OpenLayers 2.8
- uDig 1.1.1
- OpenJUMP PIROL 1.2.F
- PostgreSQL 8.4.0
- PostGIS 1.3.6

GRASS i18N

"Official" language: ***english***

Translations into **19 languages** for the user interface are currently ongoing.

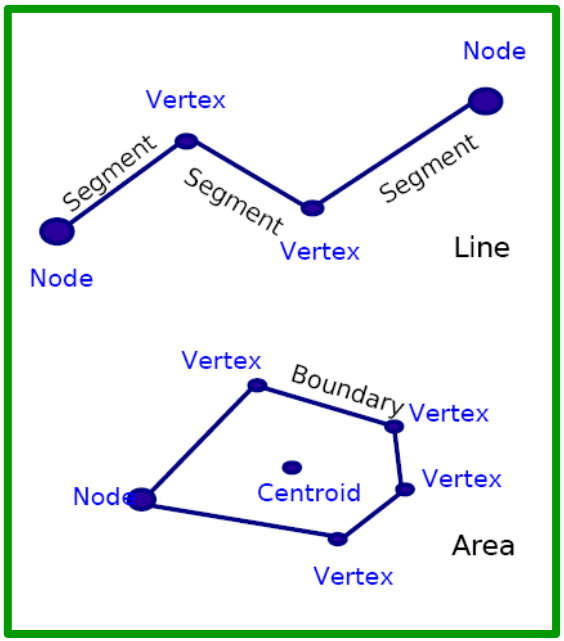


PO-Files		Translated messages	Fuzzy translations	Untranslated messages
Czech	grasstcl_cs.po	1429		
Thai	grasstcl_th.po	1424	6	1
Spanish	grasstcl_es.po	1422	2	5
Portuguese	grasstcl_pt_br.po	1361	39	31
Italian	grasstcl_it.po	1342	310	158
Portuguese	grasstcl_pt.po	1191	162	78
Turkish	grasstcl_tr.po	1126	3	302
French	grasstcl_fr.po	794	268	369
German	grasstcl_de.po	765	399	265
Polish	grasstcl_pl.po	732	375	324
Vietnamese	grasstcl_vi.po	726	420	285
Chinese	grasstcl_zh.po	221	15	1195
Amharic	grasstcl_am.po	174	80	1177
Slovenian	grasstcl_sl.po	0	1431	
Russian	grasstcl_ru.po	0	187	1244
Korean	grasstcl_ko.po	0	1431	
Greek, Modern (1453-)	grasstcl_el.po	0	1431	
Arabic	grasstcl_ar.po	0	1431	
Hindi	grasstcl_hi.po	0	1431	
Japanese	grasstcl_ja.po	0	667	764
Latvian	grasstcl_lv.po	0	125	1306
Marathi	grasstcl_mr.po	0	1431	

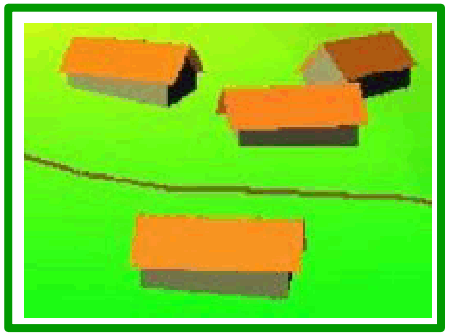
Geometry and topology

- Geometry is true 3D (x, y, z)*:
  - Point
  - Centroid
  - Line
  - Boundary
  - Area* = boundary + centroid
  - Face* = 3D area
  - Kernel* = 3D centroid
  - Volume* = face + kernel

(*) Not in all GIS!!



Face



OGC simple features…

Points, lines, polygons
Replicated boundaries for adiacent areas

Advantages:

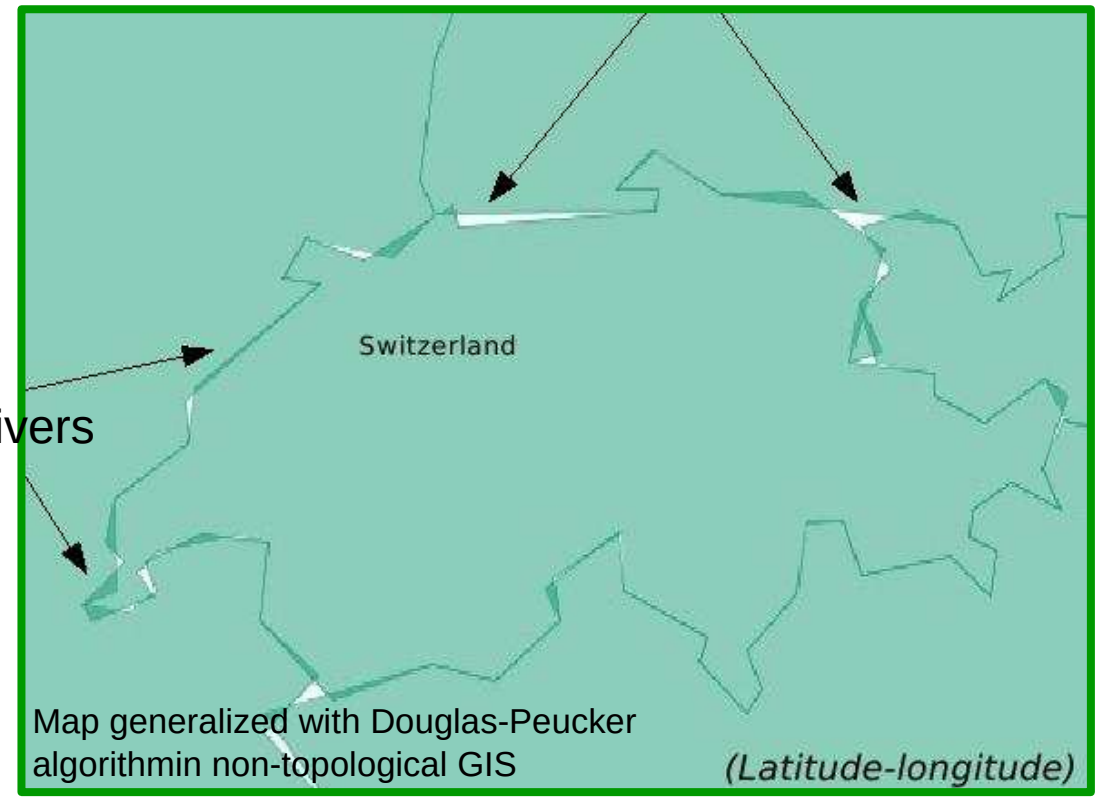
- faster computations

Disadvantages:

- Extra work for data maintenance
- Duplicated boundaries cause problems

gaps

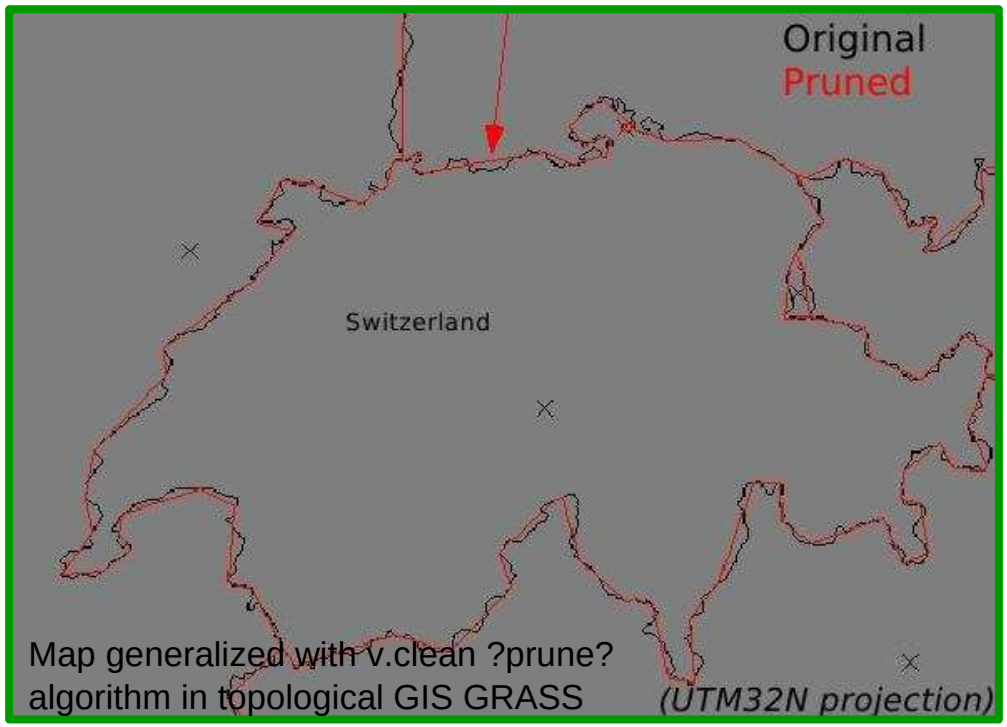
slivers



… VS Vector topology

Points, centroids, lines, boundaries
In topology centroid and boundary form an area
Single boundaries for adjacent areas

each boundary is a single line
divided by two polygons



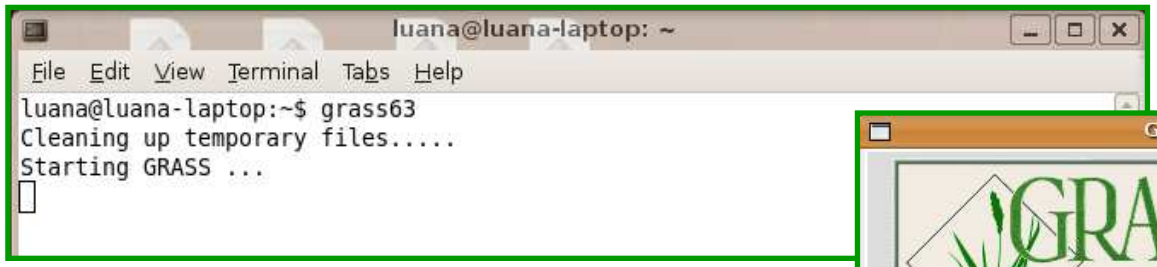
Advantages:

- less maintenance, high quality

Disadvantages:

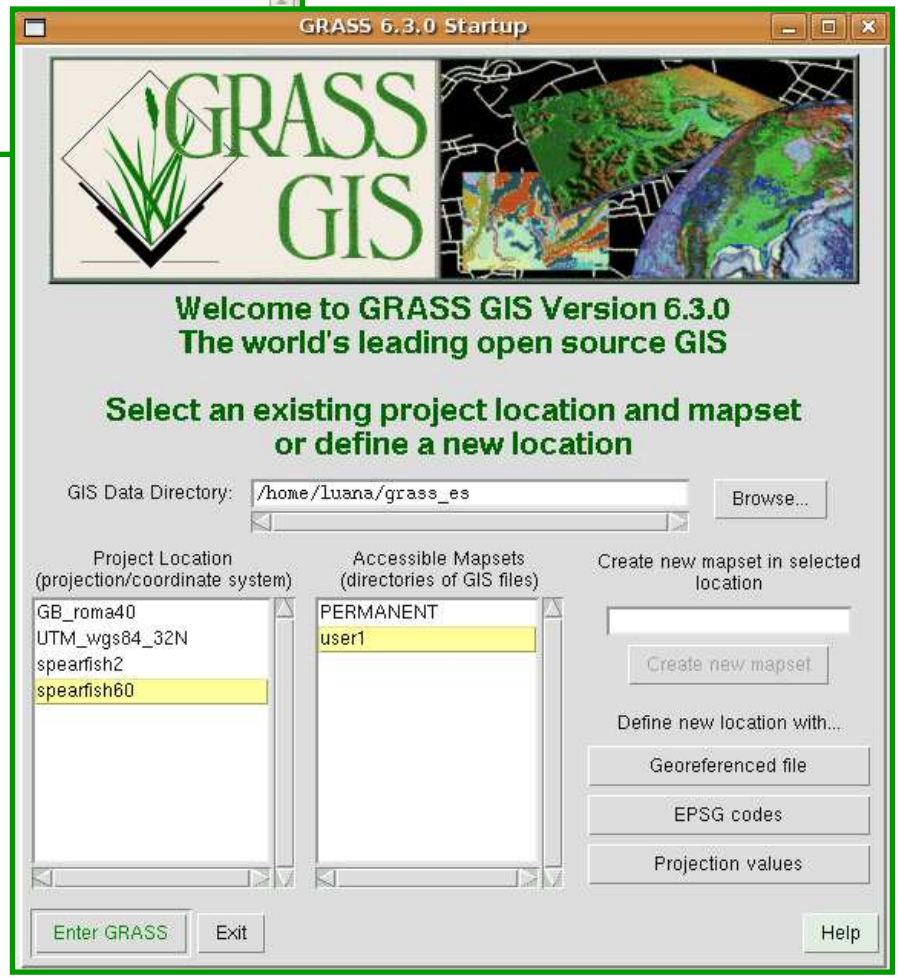
- Slower computations

How to start GRASS



You can start Grass typing '*grass63*' in the shell.

In the startup window select the mapset and press the '*Enter GRASS*' button.



Data structure

- **DATABASE**: Contains all GRASS data.

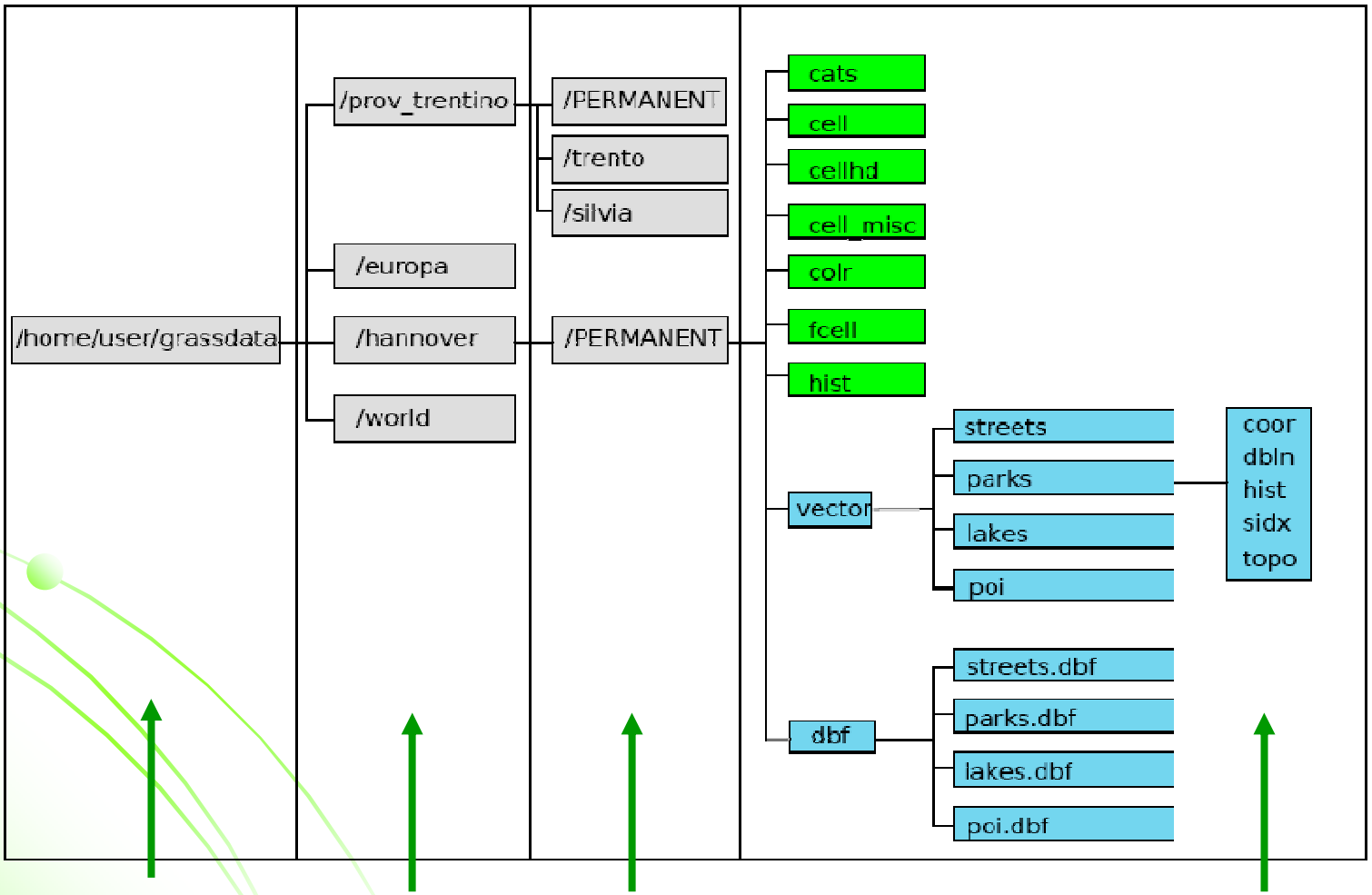
Each GRASS project is organized in a “Location” directory with subsequent “Mapset” directories

- **LOCATION**: Defines a coordinate system and a rectangular boundary for a project

- **MAPSET(s)**: Used to subdivide data by user names or subregions or access rights.
  - **PERMANENT**: is a standard mapset that contains the definitions of the location. May also contain general cartography since it is visible to all the other mapsets.

- **Multi-User** support: multiple users can work in a single location using different mapsets. Access rights can be managed per user. No user can modify/delete data of other users.

Data structure - 2



Vector maps structure

- Each vector map is contained in a single directory (inside the '*vector*' directory) with the same name of the vector map.

- Each vector directory contains the following files:

  - **cidx:** Contains the topological index
  - **coor:** Binary file that contains the features coordinates and categories
  - **dbln:** Contains information about database connection (driver, database, table, column, etc...)
  - **head:** Contains the vector header
  - **hist**: Contains information about the vector history: author, date, executed commands, etc...
  - **topo:** Binary file containing topology

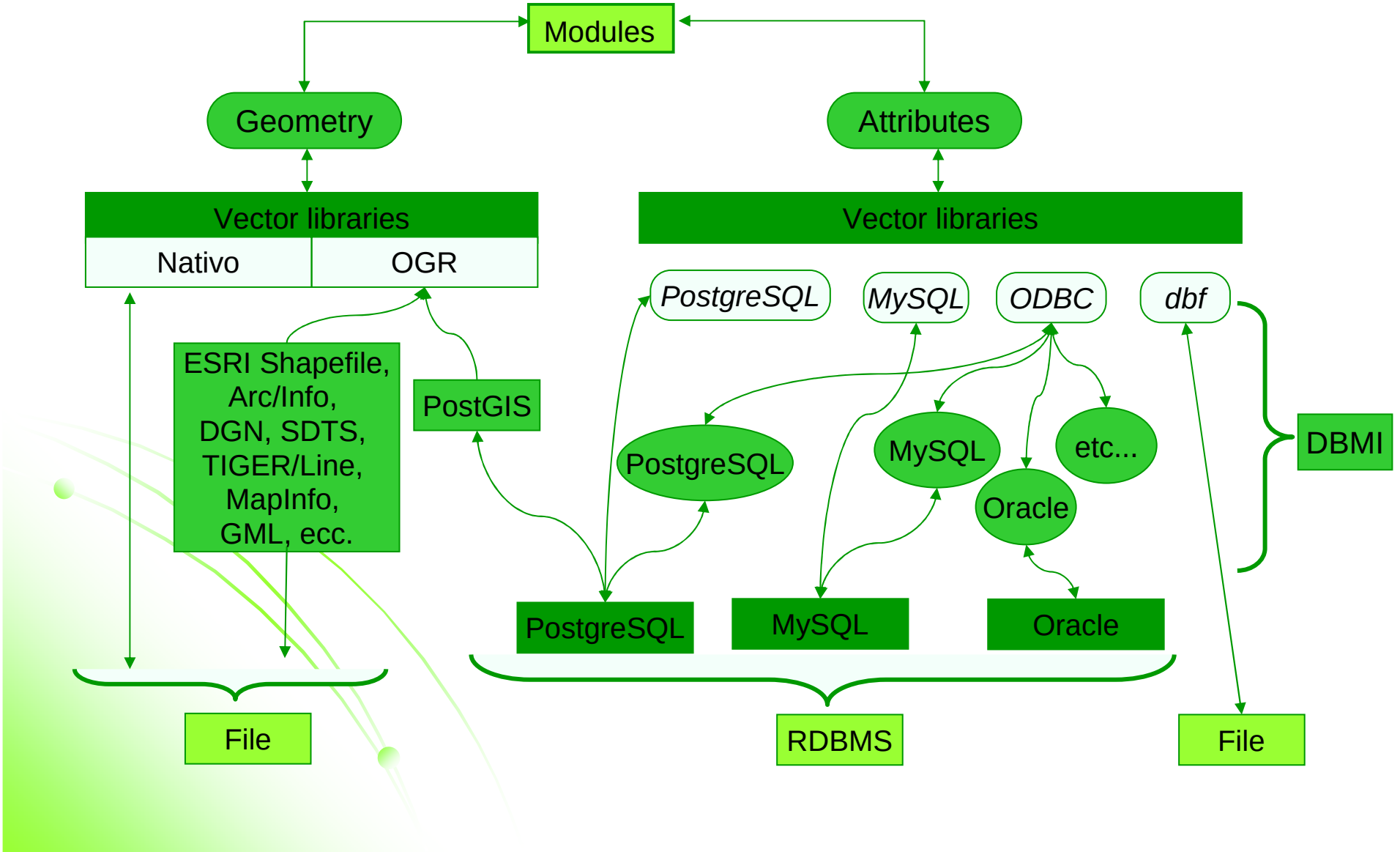
Raster maps structure

- Raster maps are recorded in a matrix where each element represents a pixel with an **integer** or floating **point value**.

- Information about the raster maps is distributed in different files inside thematical subdirectories:

  - **cat**: Contains category information
  - **cell – fcell**: Contains binary files with the numerical matrix
  - **cellhd**: Contains maps headers
  - **colr**: Contains maps color informationon
  - **hist**: Contains information about raster map history: author, date, executed commands, etc...

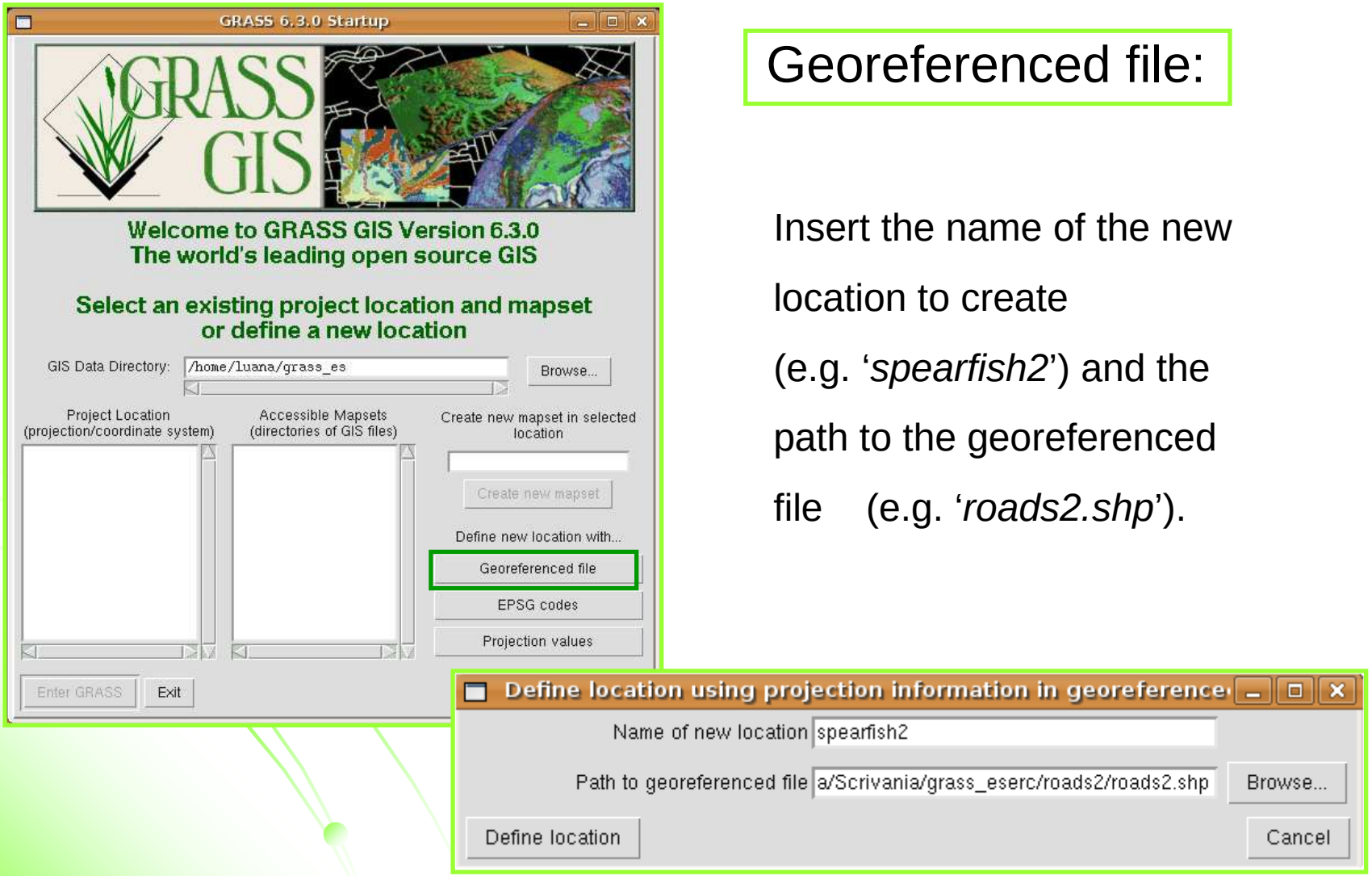
Databases with GRASS



Commands structure

Prefix	Function class	Type of command	Example
d.*	display	graphical output	d.rast/d.vector: views raster / vector maps
db.*	database	database management	db.select: select value(s) from table
g.*	general	general file operations	g.rename: renames map
i.*	imagery	image processing	i.smap: image classifier
ps.*	postscripts	map creation in Postscript	ps.map: map creation
r.*	raster	raster data processing	r.buffer: buffer around raster features
r3.*	voxel	raster voxel data processing	r3.mapcalc: volume map algebra
v.*	vector	vector data processing	v.overlay: vector map intersection

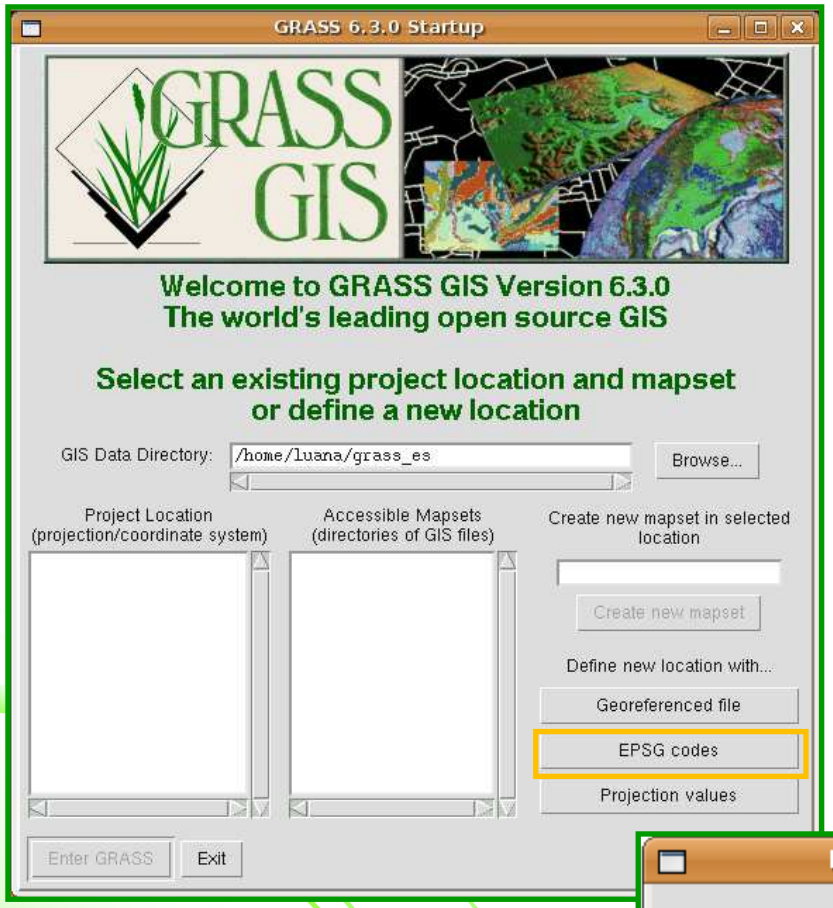
Create a new location

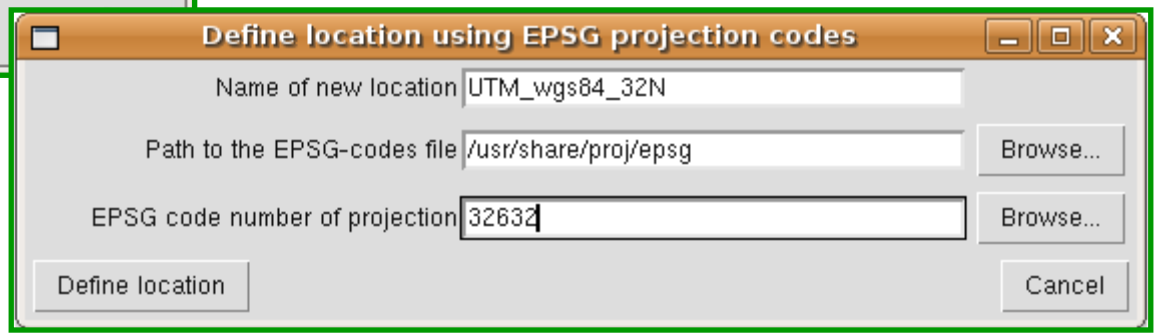


Georeferenced file:

Insert the name of the new location to create (e.g. '*spearfish2*') and the path to the georeferenced file (e.g. '*roads2.shp*').

Create a new location - 2

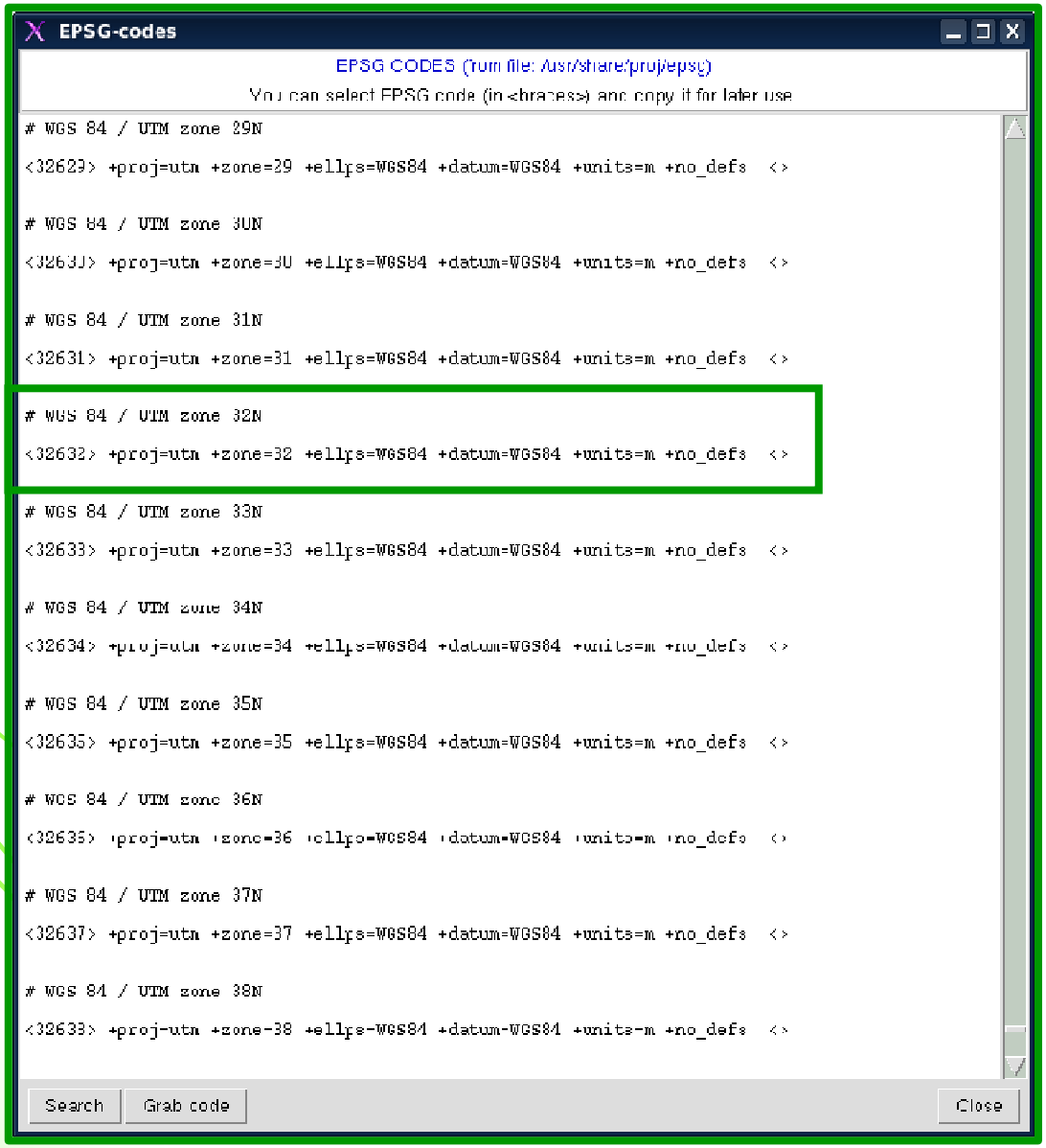




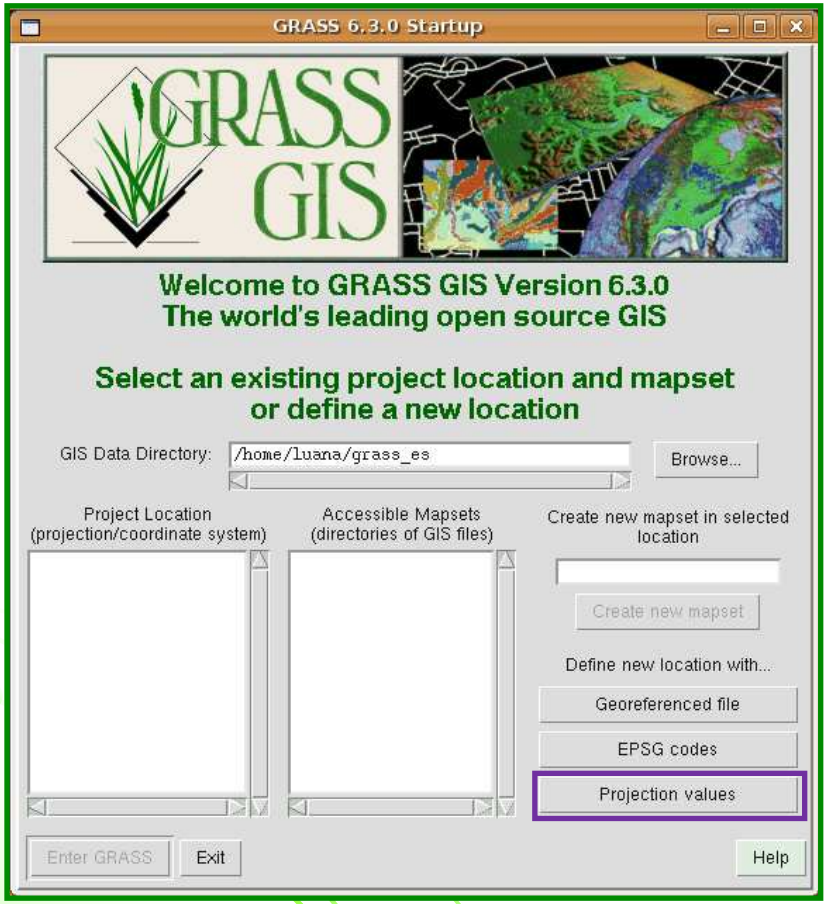
EPSG codes:

Insert the name of the new location to create (e.g. '*UTM_wgs84_32N*') , the path to the EPSG-codes file (e.g. '*proj/epsg*') and the EPSG code number of the chosen projection (e.g. *'32632*').

Create a new location - 3



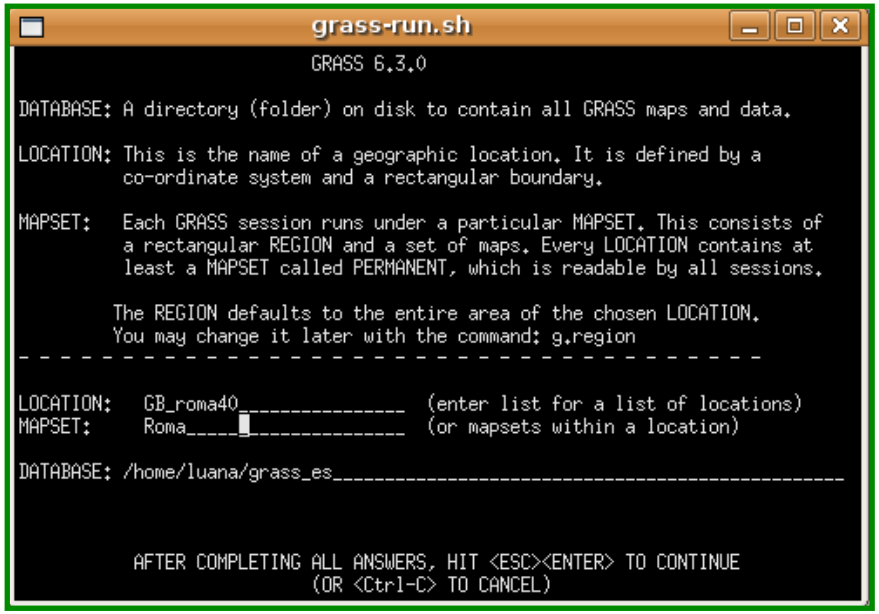
Create a new location - 4



1 Name of location, mapset and database

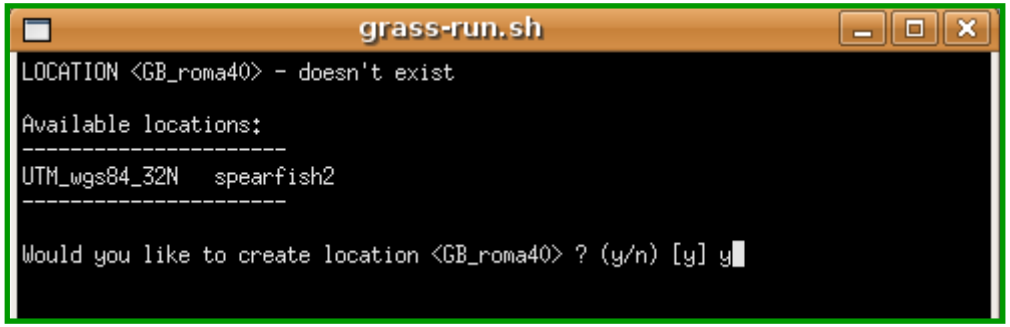
Projection values:

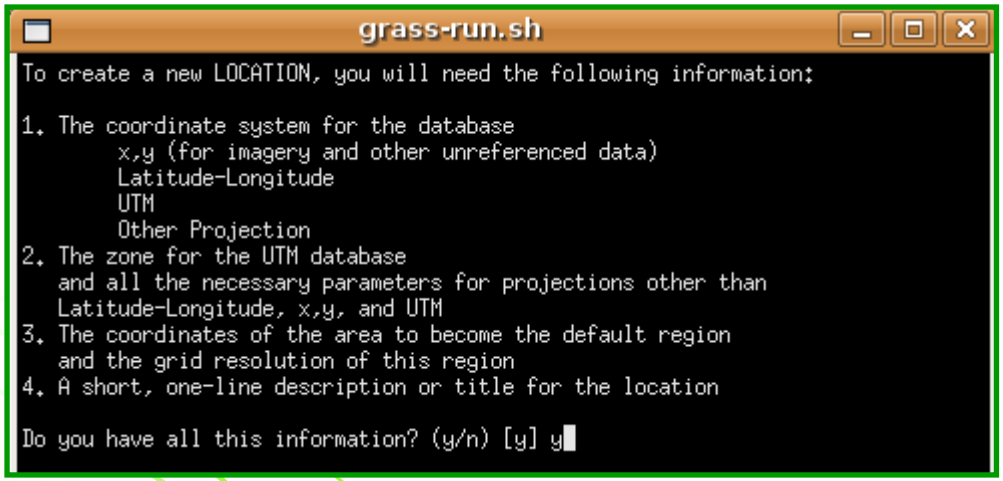
Insert the all the parameters of the chosen projection (e.g. *'Gauss-Boaga Roma40*').



Create a new location - 5

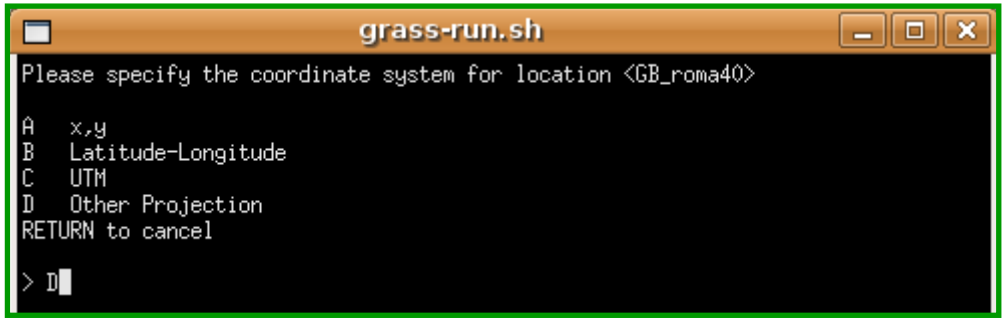
2 Creation of the new location



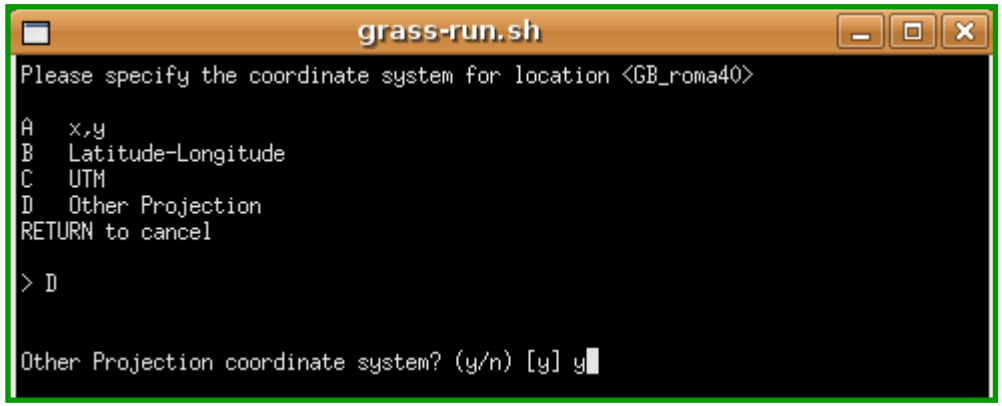


3 Checking availability of information

4 Insert type of the coordinate system

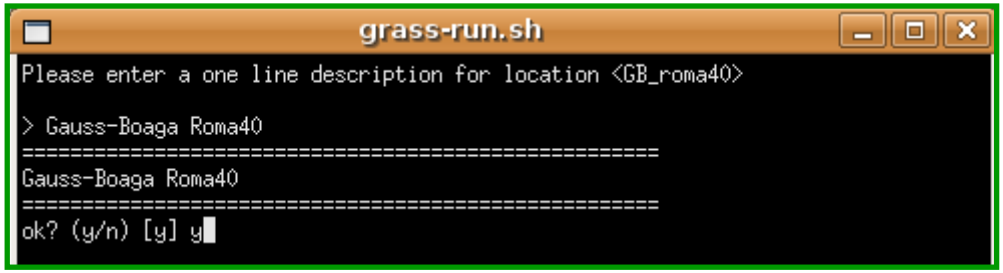


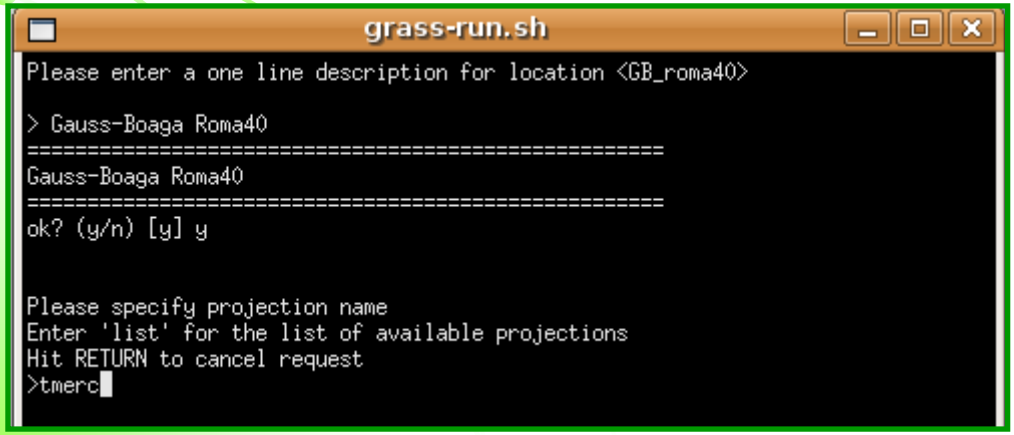
Create a new location - 6



5 Confirm the choice of 'other projection'

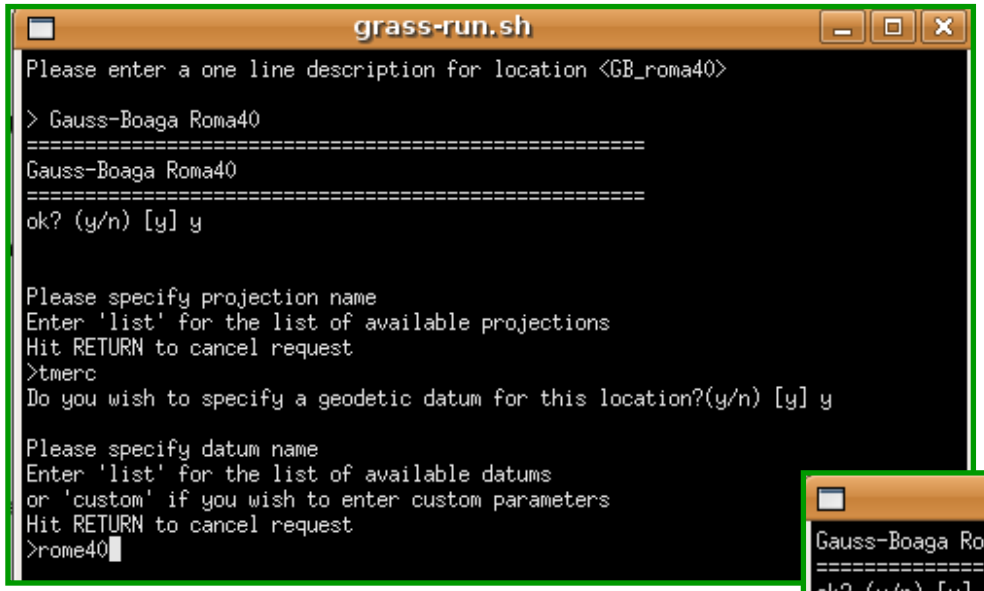
6 Insert a description of the new location



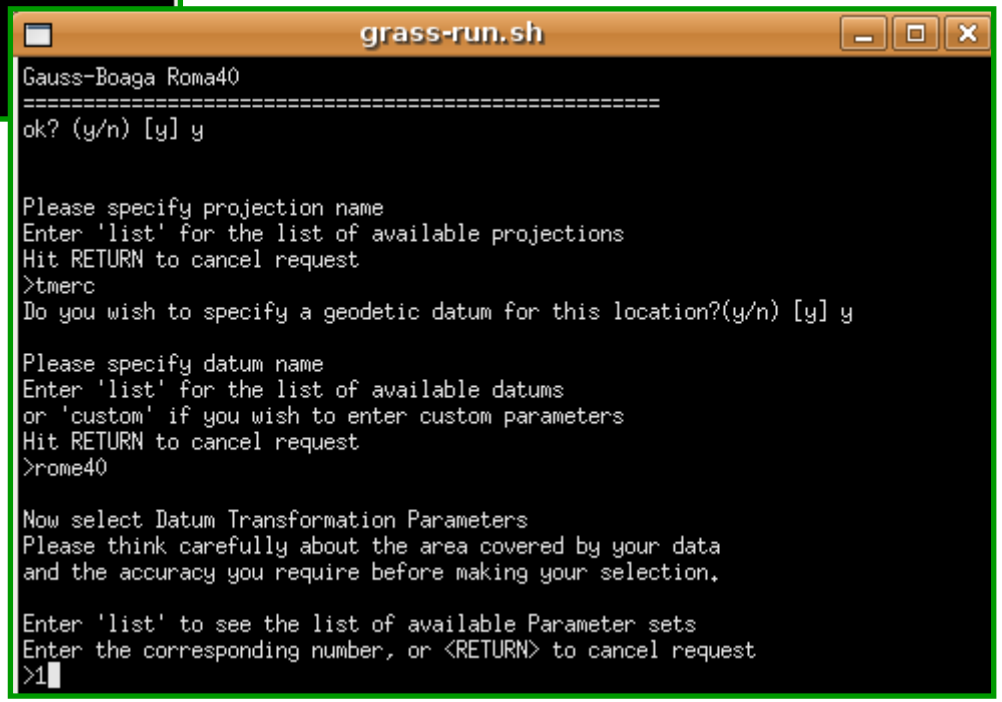


7 Specify the projection name (e.g. '*tmerc*')

Create a new location - 7

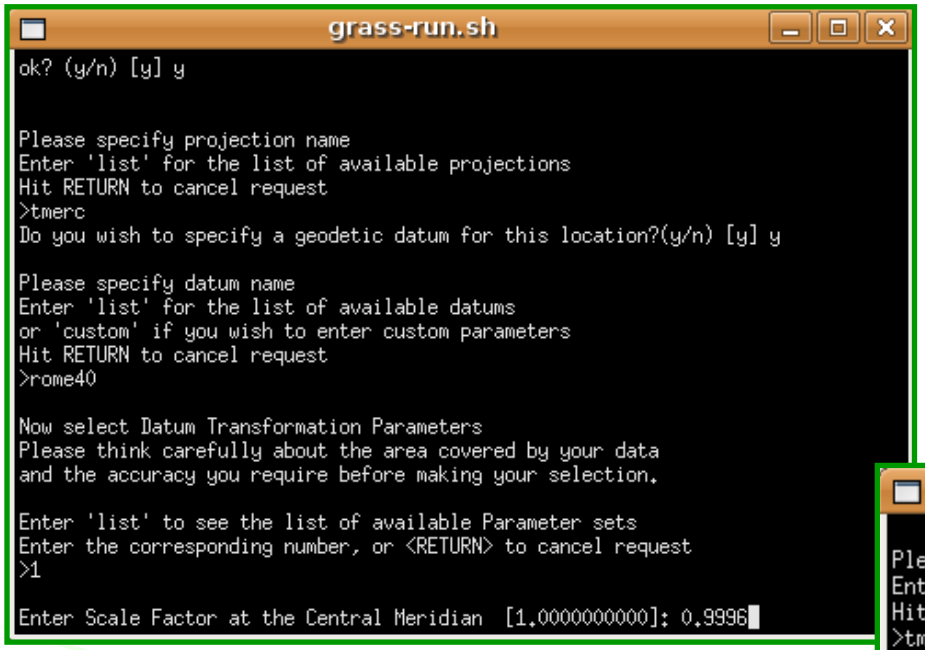


8 Specify the datum name (e.g. '*rome40*')

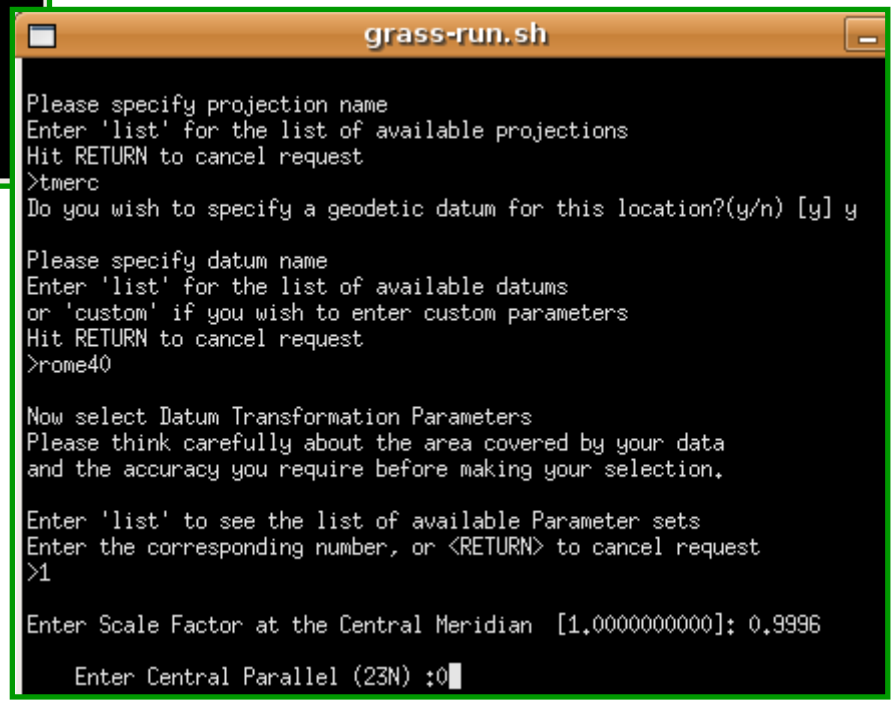


9 Select datum transformation parameters (e.g. '*1*' for Italy)

Create a new location - 8



10	Insert the scale factor at the Central Meridian (e.g. '*0.9996*')



11	Enter the Central Parallel (e.g. '*0*')

Create a new location - 9

```
Enter Scale Factor at the Central Meridian  [1.0000000000]: 0.9996
    Enter Central Parallel (23N) :0
    Enter Central Meridian (96W) :9
```

12	Insert the Central Meridian (e.g. '*9*' for W zone)

13	Enter False Easting (e.g. '*1500000*' for W zone)

```
Enter Scale Factor at the Central Meridian  [1.0000000000]: 0.9996
    Enter Central Parallel (23N) :0
    Enter Central Meridian (96W) :9
Enter False Easting  [0.0000000000]: 1500000
```

```
Enter Scale Factor at the Central Meridian  [1.0000000000]: 0.9996
    Enter Central Parallel (23N) :0
    Enter Central Meridian (96W) :9
Enter False Easting  [0.0000000000]: 1500000
Enter False Northing  [0.0000000000]: 0
```

14	Enter False Northing (e.g. '*0*')

Create a new location - 10

```
Enter Scale Factor at the Central Meridian  [1.0000000000]: 0.9996

    Enter Central Parallel (23N) :0

    Enter Central Meridian (96W) :9

Enter False Easting  [0.0000000000]: 1500000

Enter False Northing  [0.0000000000]: 0
Enter plural form of units [meters]: meters
```

15	Insert plural form of units (e.g. '*meters*')

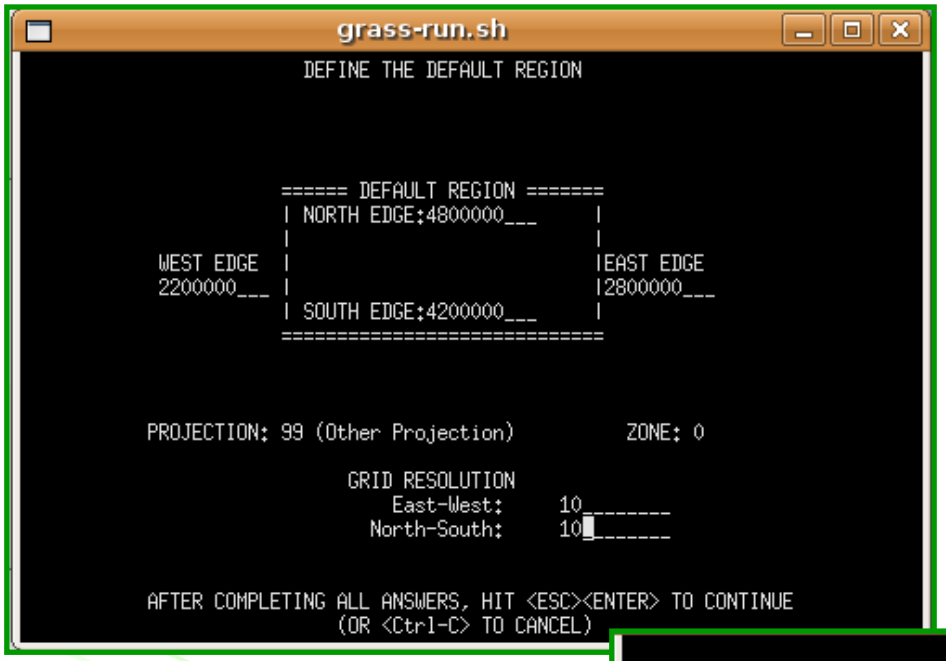
16	Enter singular for unit (e.g. '*meter*')

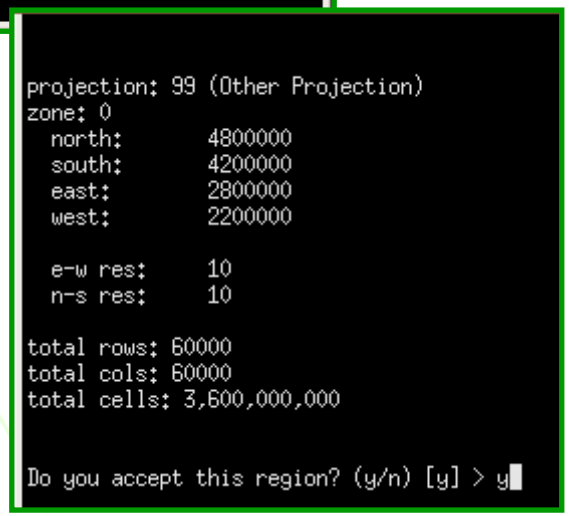
```
Enter Scale Factor at the Central Meridian  [1.0000000000]: 0.9996

    Enter Central Parallel (23N) :0

    Enter Central Meridian (96W) :9

Enter False Easting  [0.0000000000]: 1500000

Enter False Northing  [0.0000000000]: 0
Enter plural form of units [meters]: meters
Enter singular for unit: meter
```

```
Enter Scale Factor at the Central Meridian  [1.0000000000]: 0.9996

    Enter Central Parallel (23N) :0

    Enter Central Meridian (96W) :9

Enter False Easting  [0.0000000000]: 1500000

Enter False Northing  [0.0000000000]: 0
Enter plural form of units [meters]: meters
Enter singular for unit: meter
Enter conversion factor from meters to meters: 1
```

17	Enter conversion factor (e.g. '*1*')

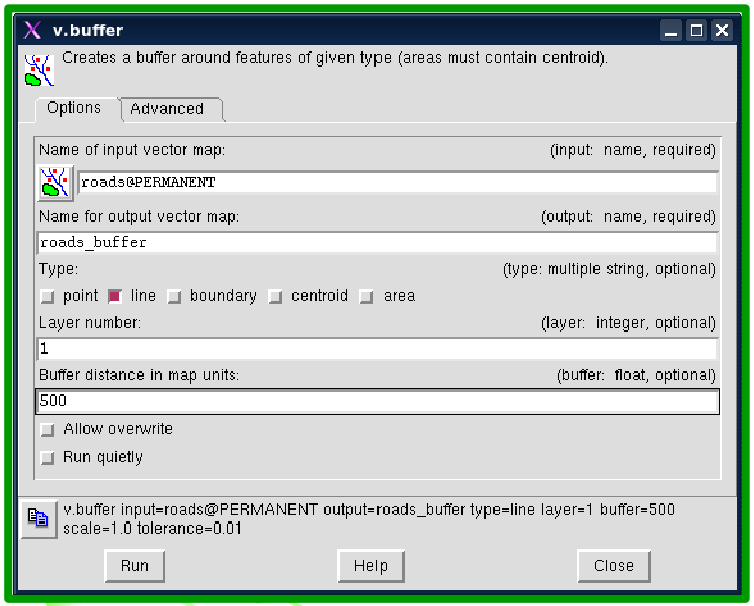
Create a new location - 11





18	Definition of the default region

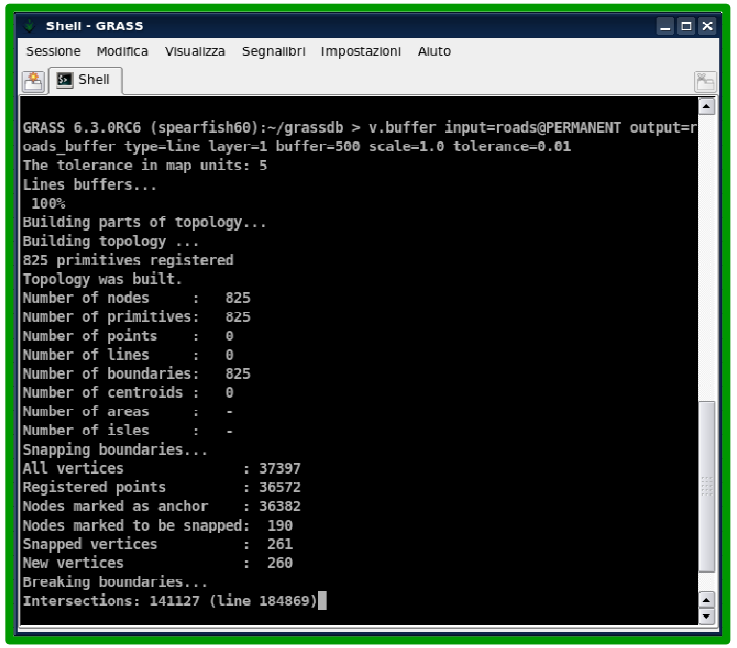
User interfaces



GIS manager
(default)

and / or

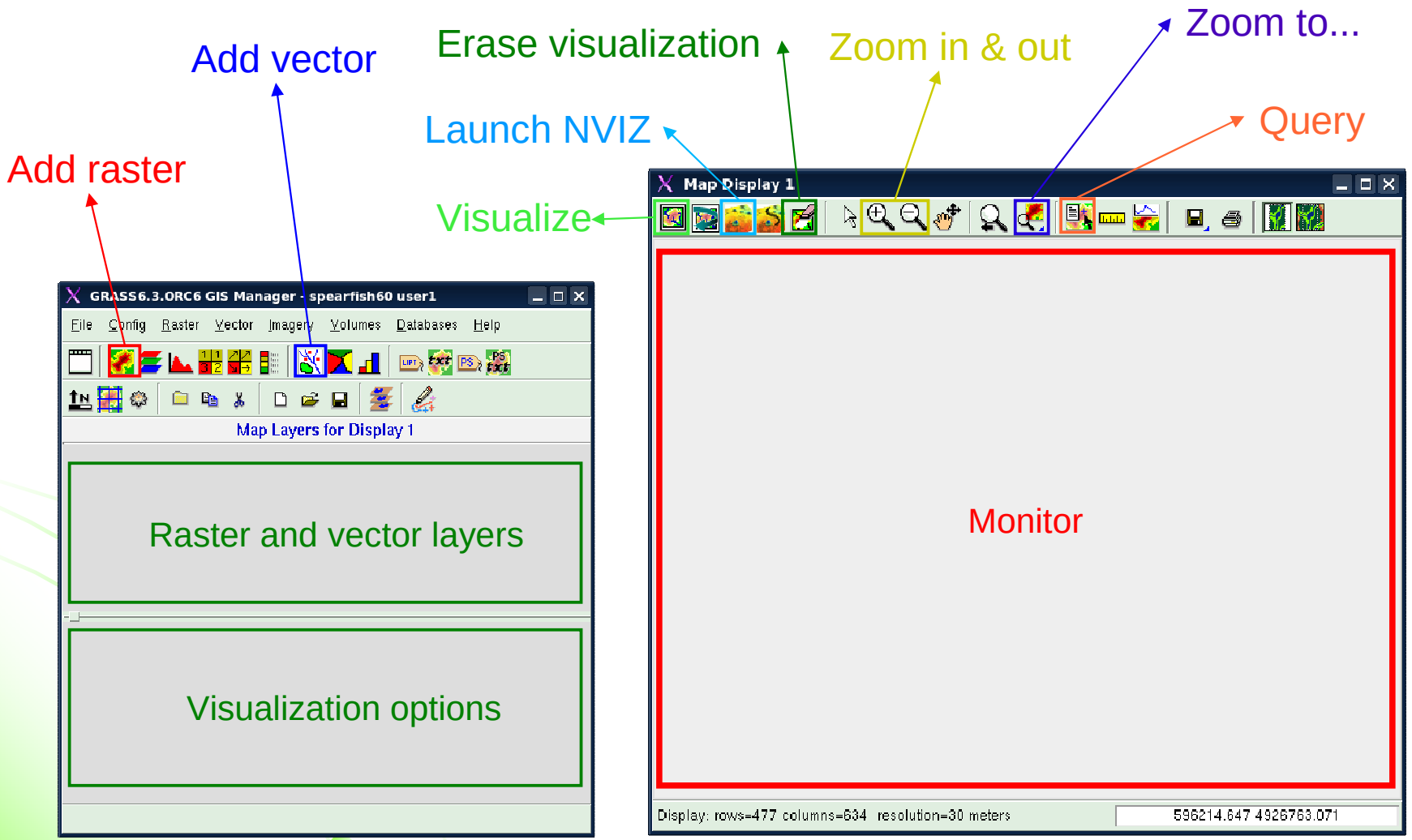
Command line



additional ways of using GRASS:

- QGIS (with GRASS plug-in)
- JavaGRASS (JGRASS)

GIS manager


Add vector
Erase visualization
Zoom in & out
Zoom to...
Launch NVIZ
Query
Add raster
Visualize
GRASS6.3.ORC6 GIS Manager - spearfish60 user1
File Config Raster Vector Imagery Volumes Databases Help
Map Layers for Display 1
Raster and vector layers
Visualization options
Map Display 1
Monitor
Display: rows=477 columns=634 resolution=30 meters
596214.647 4926763.071

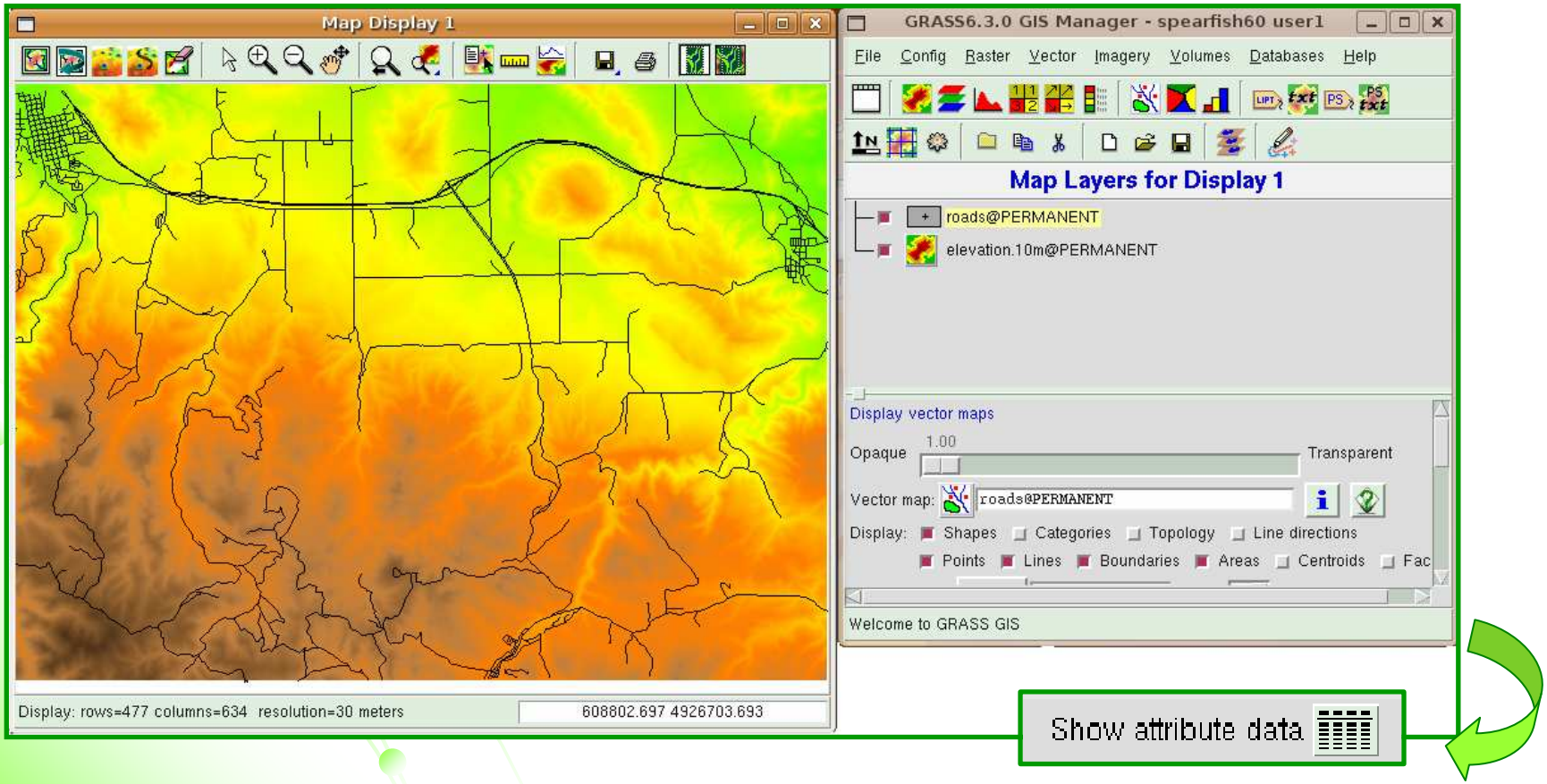
Region in GRASS

- A *region* defines the resolution used for maps and the coordinates of boundaries (map extents)
  - **Default**: Settings of the working GRASS location
  - **Current**: Is the actual working area and it can be modified by the user
- Vectorial modules work all over the vector map
- Raster modules work with the current resolution/region
- How to modify the region:
  - **g.region**: Changes the resolution and the coordinate boundaries of the current region.

[Config -> Region -> Change region settings]

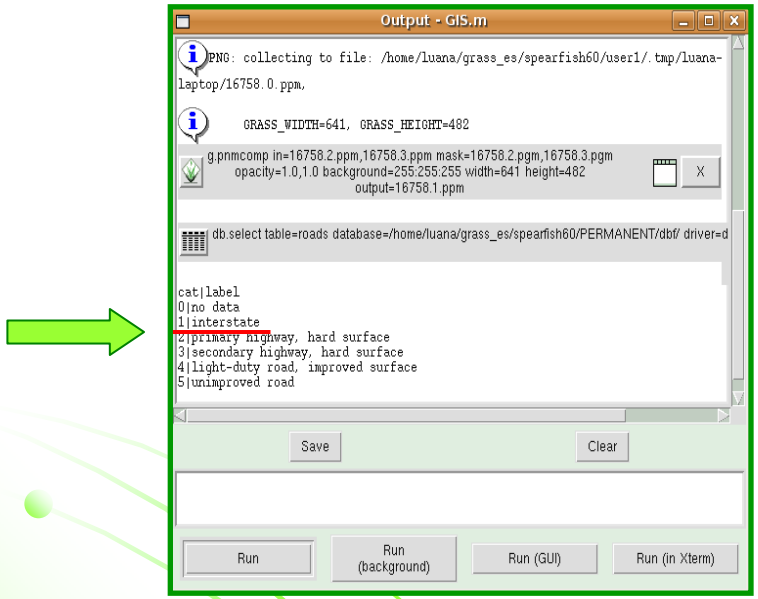
Vector - Attribute query

Extract interstate (highway) from roads vector map and buffer interstate for 3km in each direction.

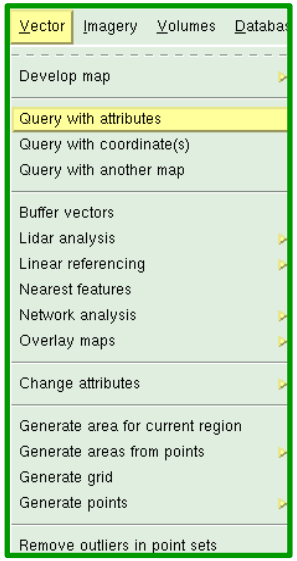


Vector - Attribute query - 2

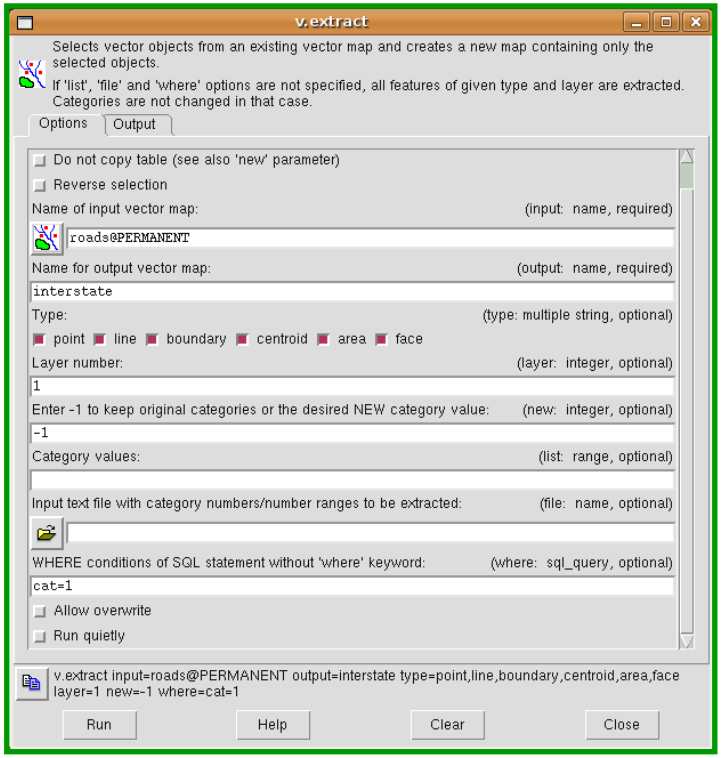
First take a look at the attribute table to get the column name and value for the interstate road:



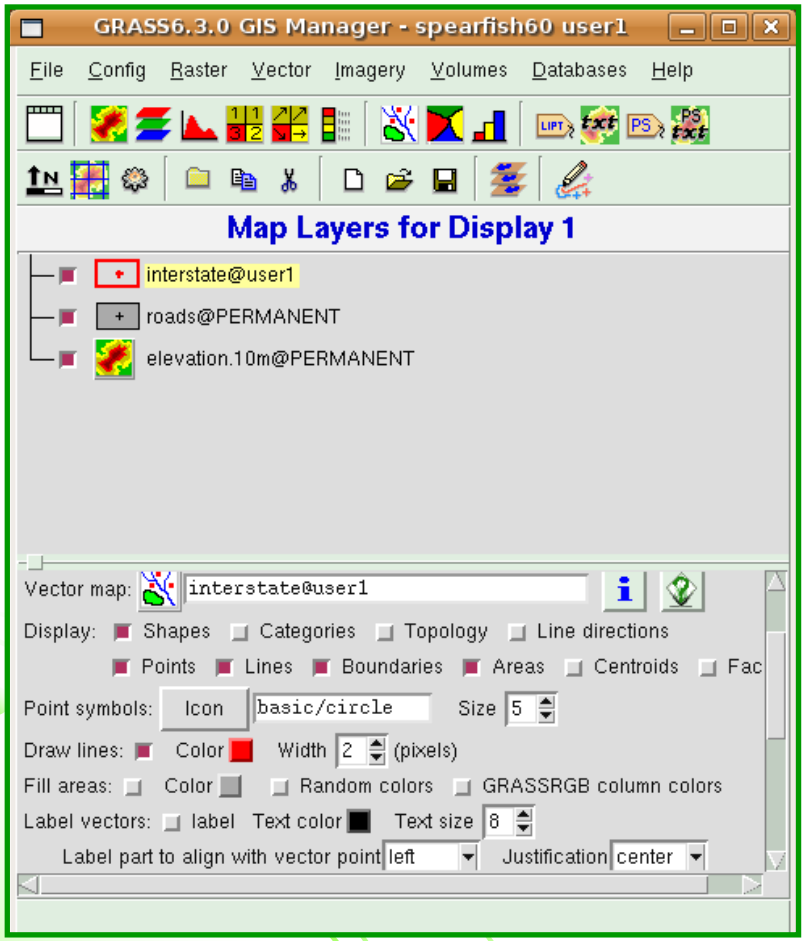
Extract only the **interstate** road:



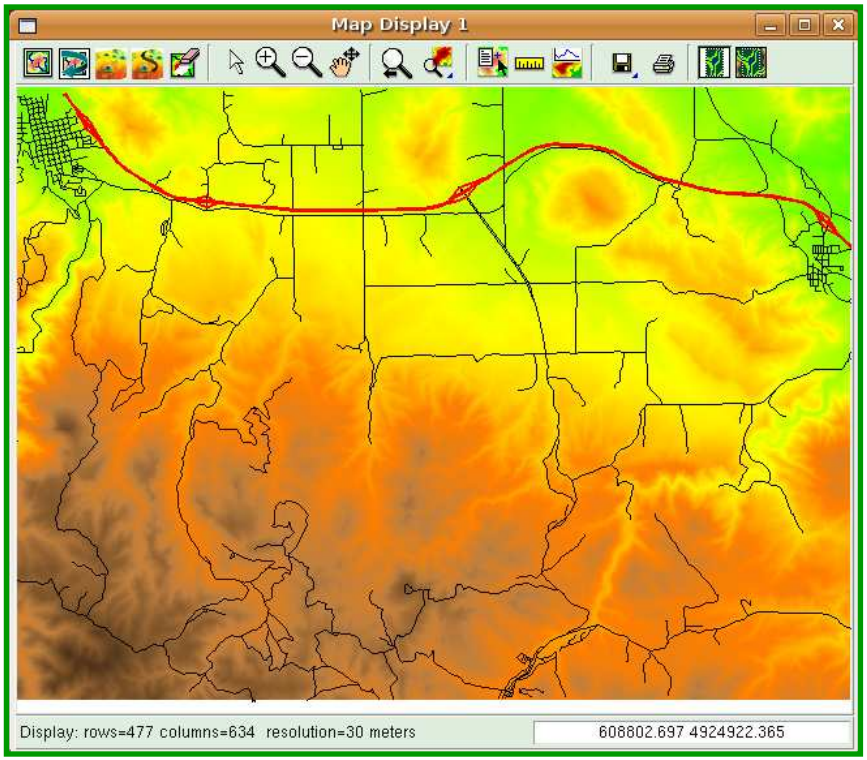
v.extract



Vector - Attribute query - 3

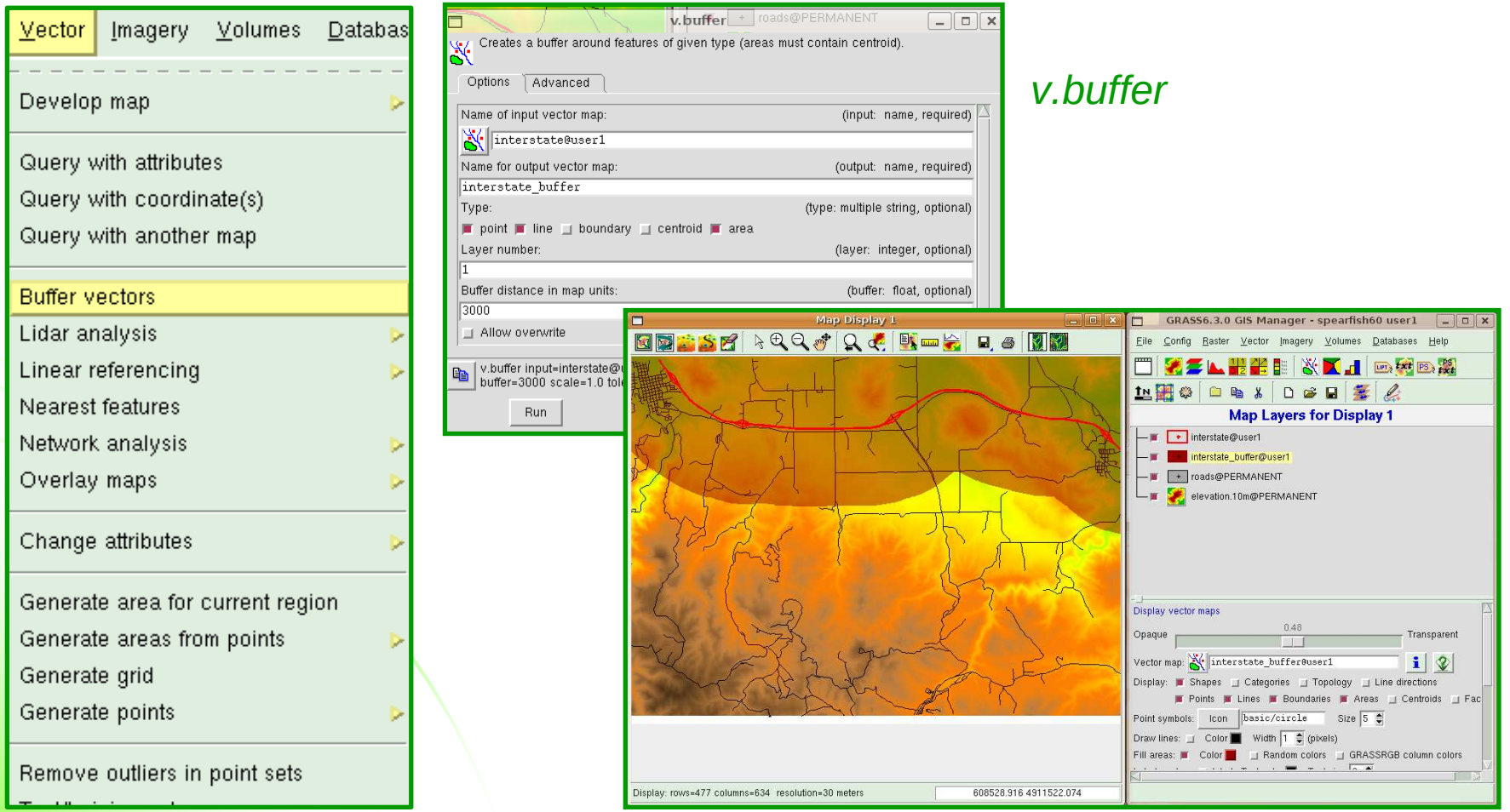


Assign a line width = 2 and a red color to the extracted road



Buffer – *noise impact example*

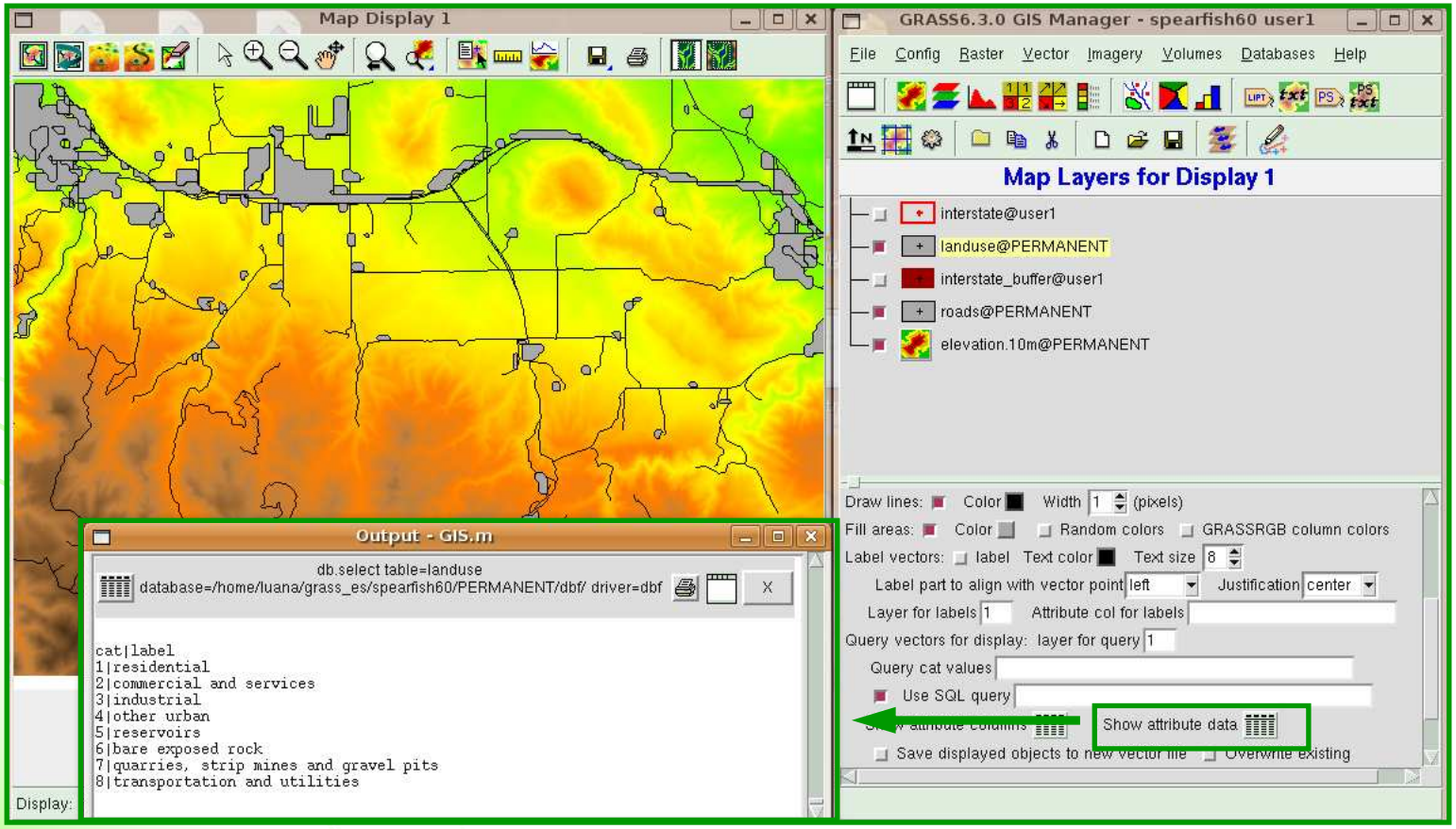
Buffer the interstate road (give buffer in map units which is meters here --> 3000)



v.buffer

From vector to raster

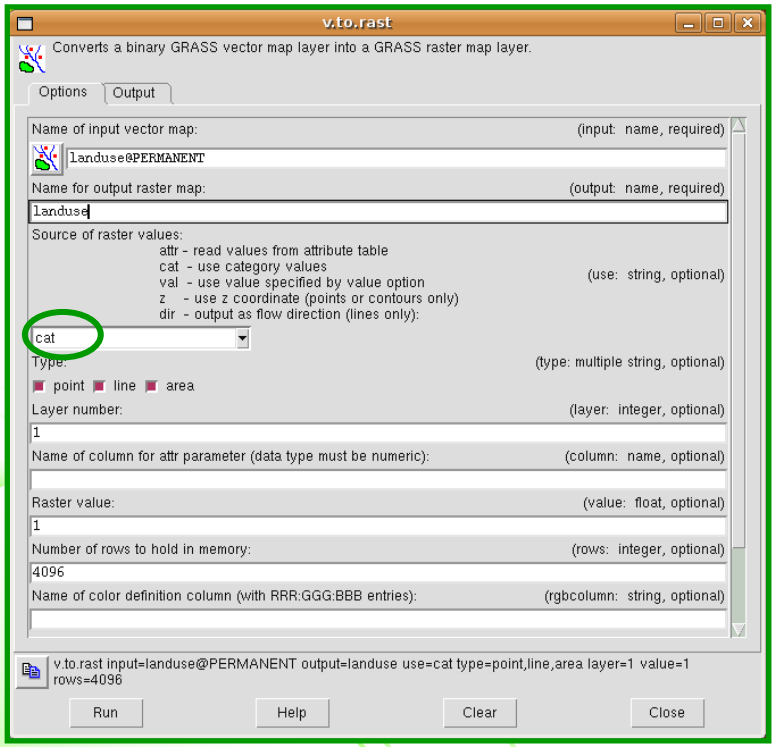
Look at the attribute table of vector map '*landuse*':



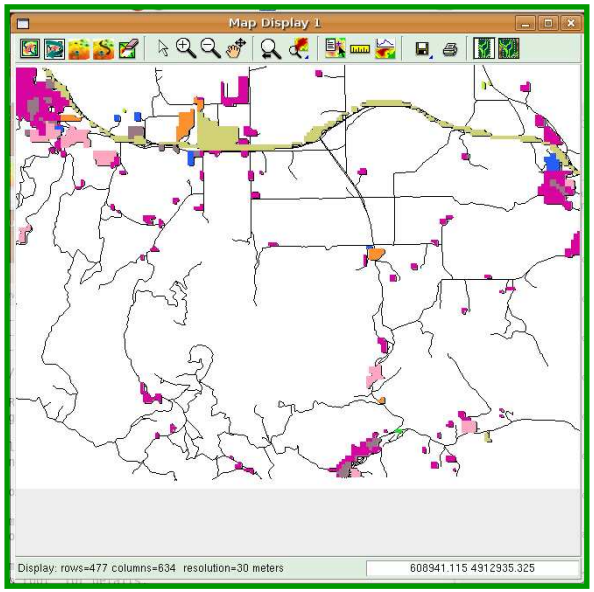
From vector to raster - 2

We can create a raster map starting from '*landuse*' vector map using the command **v.to.rast**:

v.to.rast



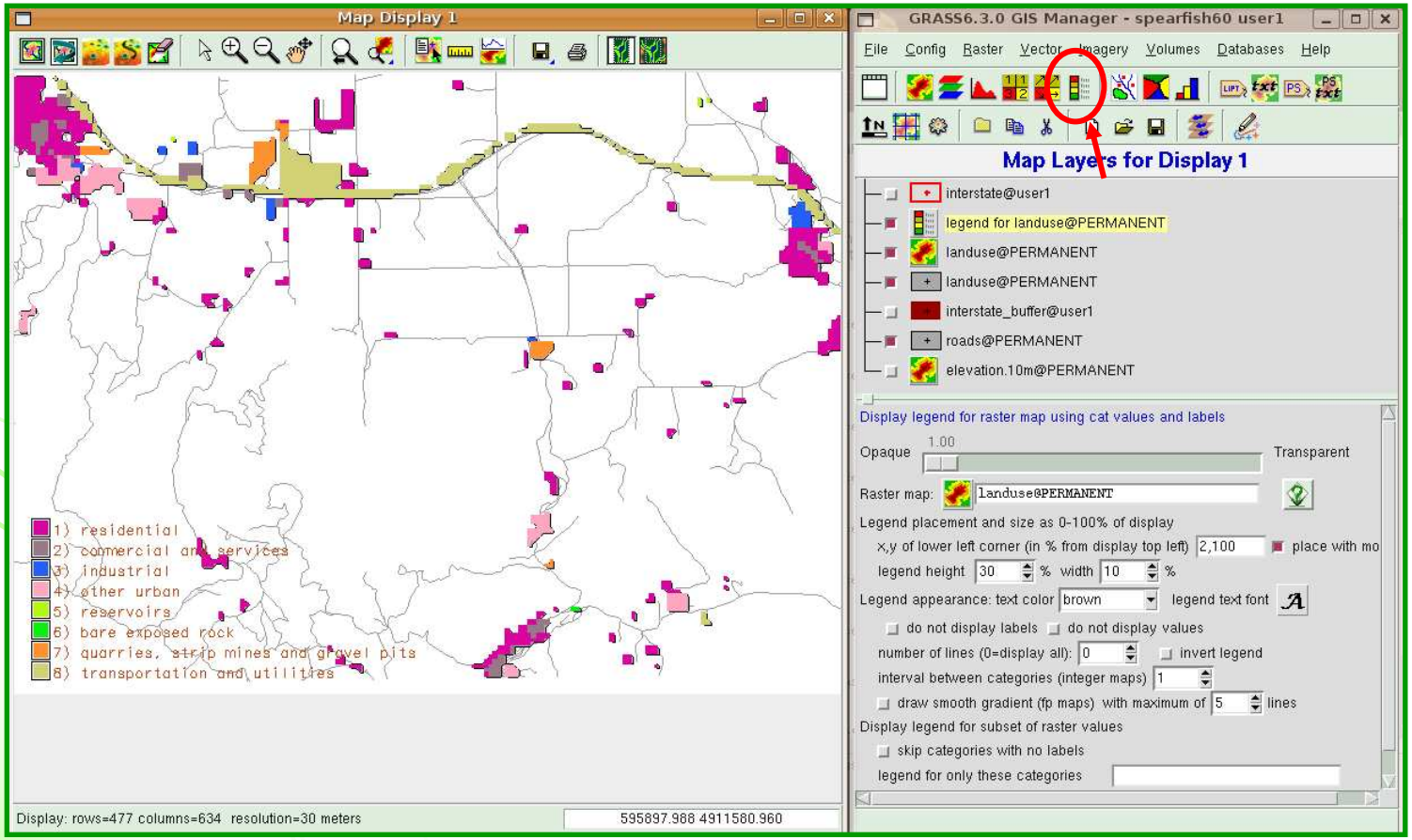
In this case we choose to use the categories values to create the new raster



Remark: the result depends on region settings, pay attention to the resolution!

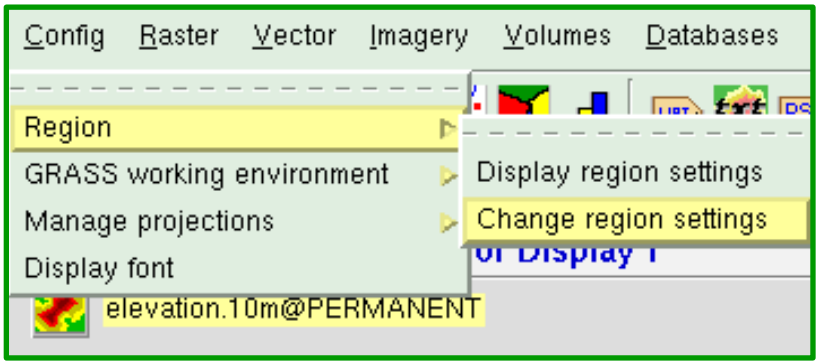
From vector to raster - 3

To understand the meaning of the different colours we can add a legend to the map:

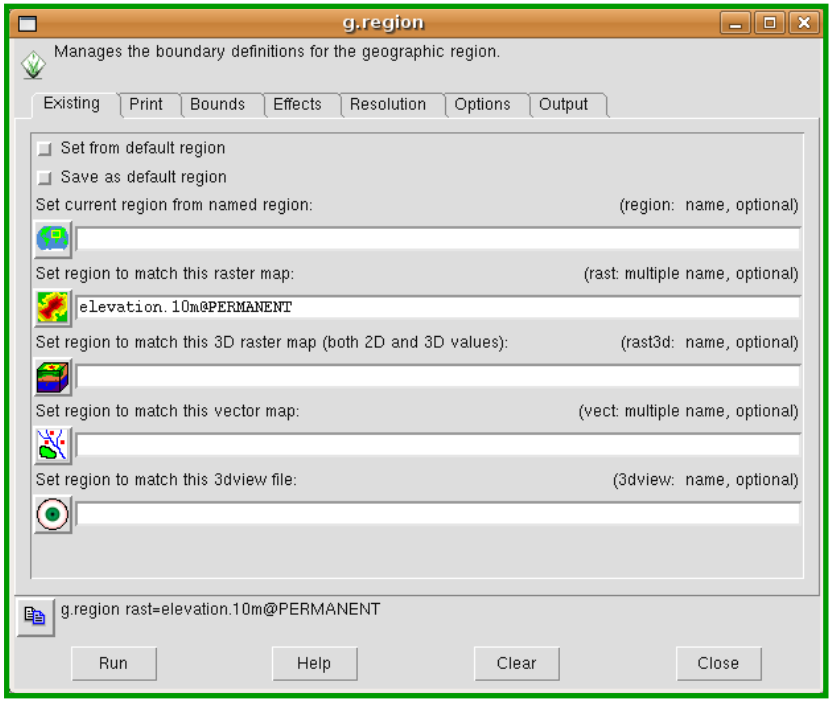


Change region settings

Set current region to match elevation.10m map:

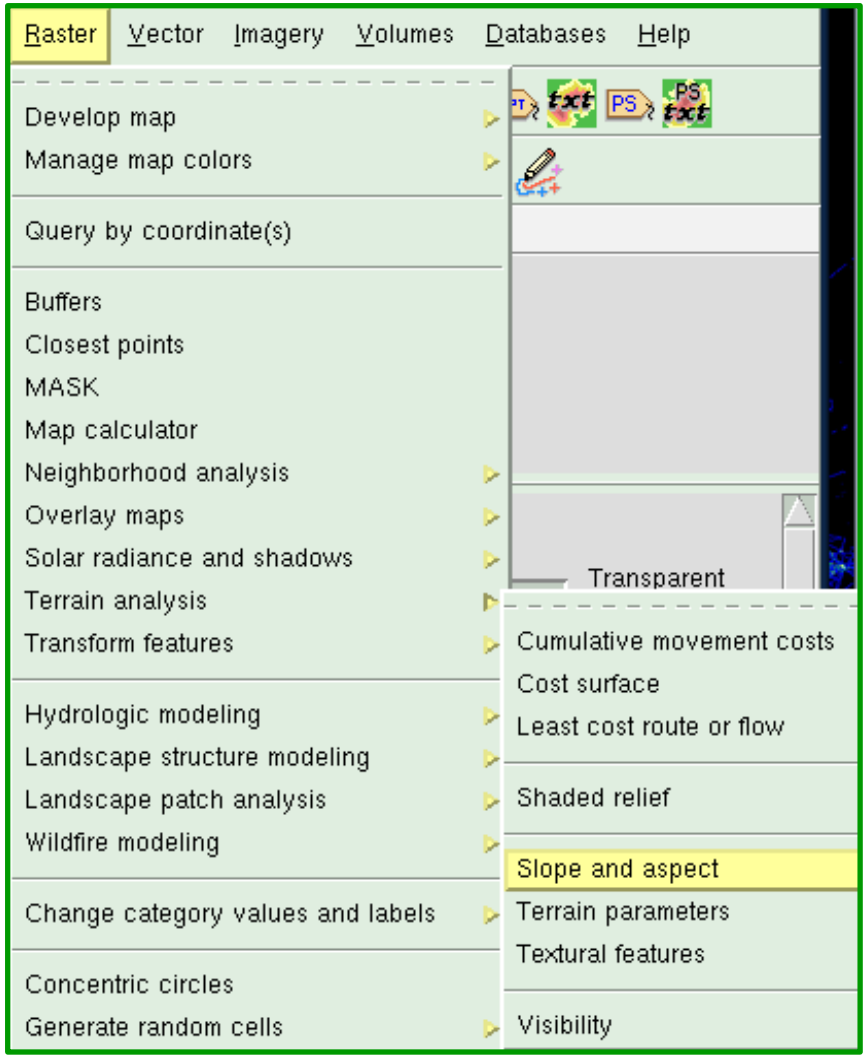


g.region

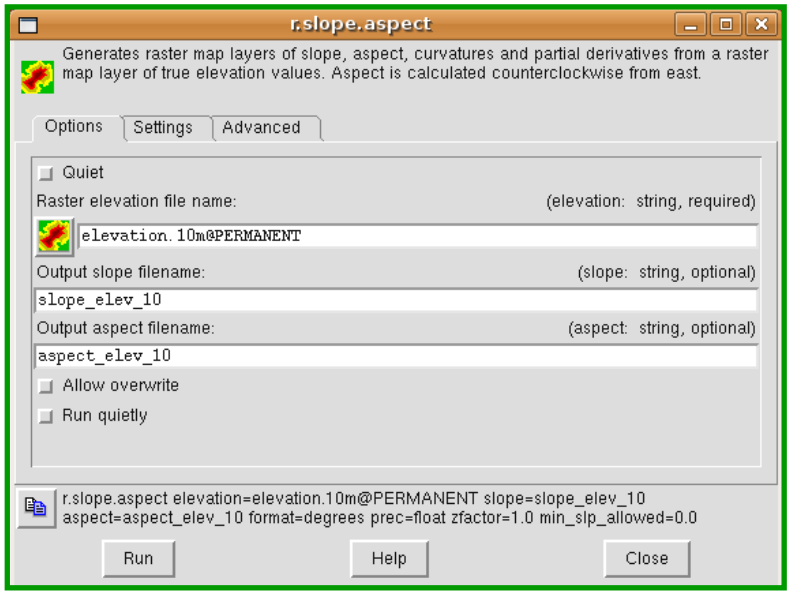


Raster – Slope & Aspect - 2

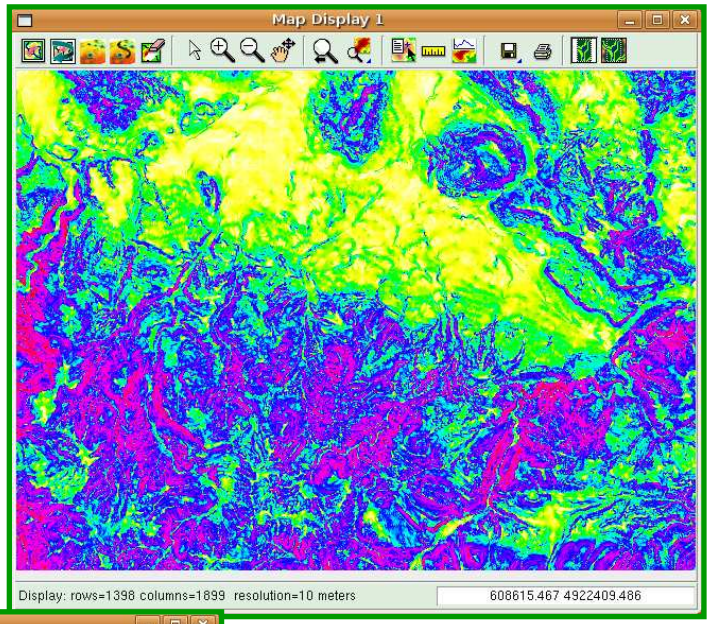
- Calculate slope and aspect from a DEM
- Horizontal angles are counted counterclockwise from the East
- Slopes are calculated by default in degrees



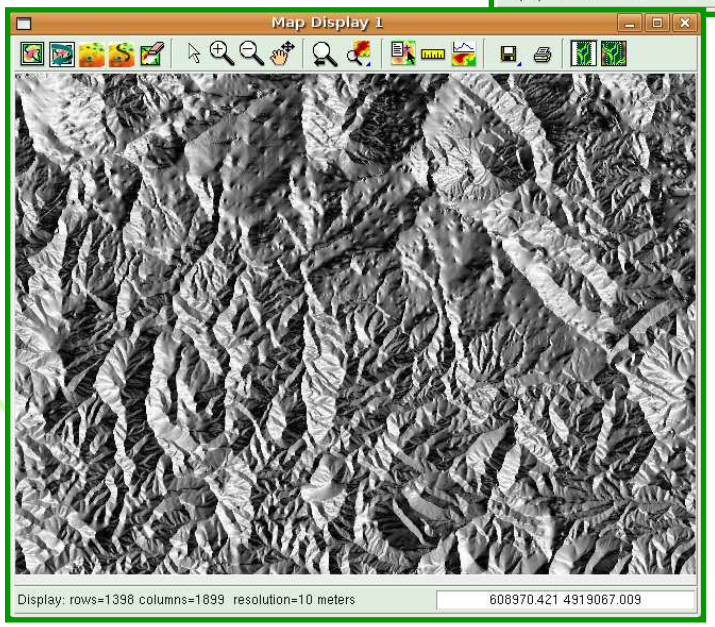
Raster – Slope & Aspect - 3



r.slope.aspect

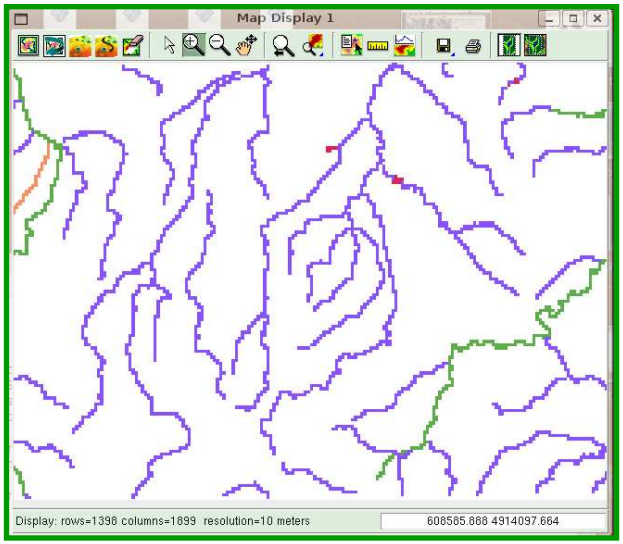


slope



aspect

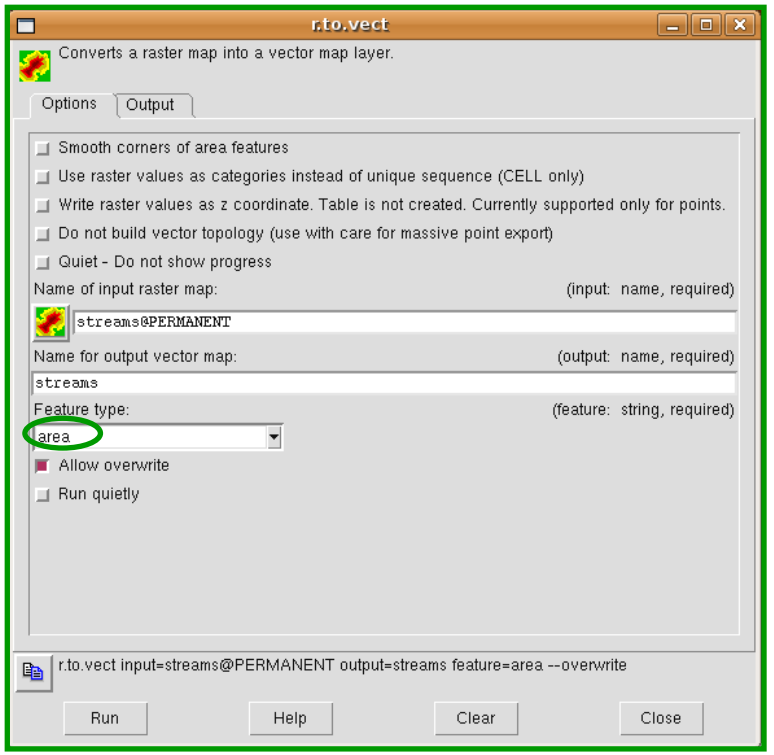
From raster to vector



streams raster map

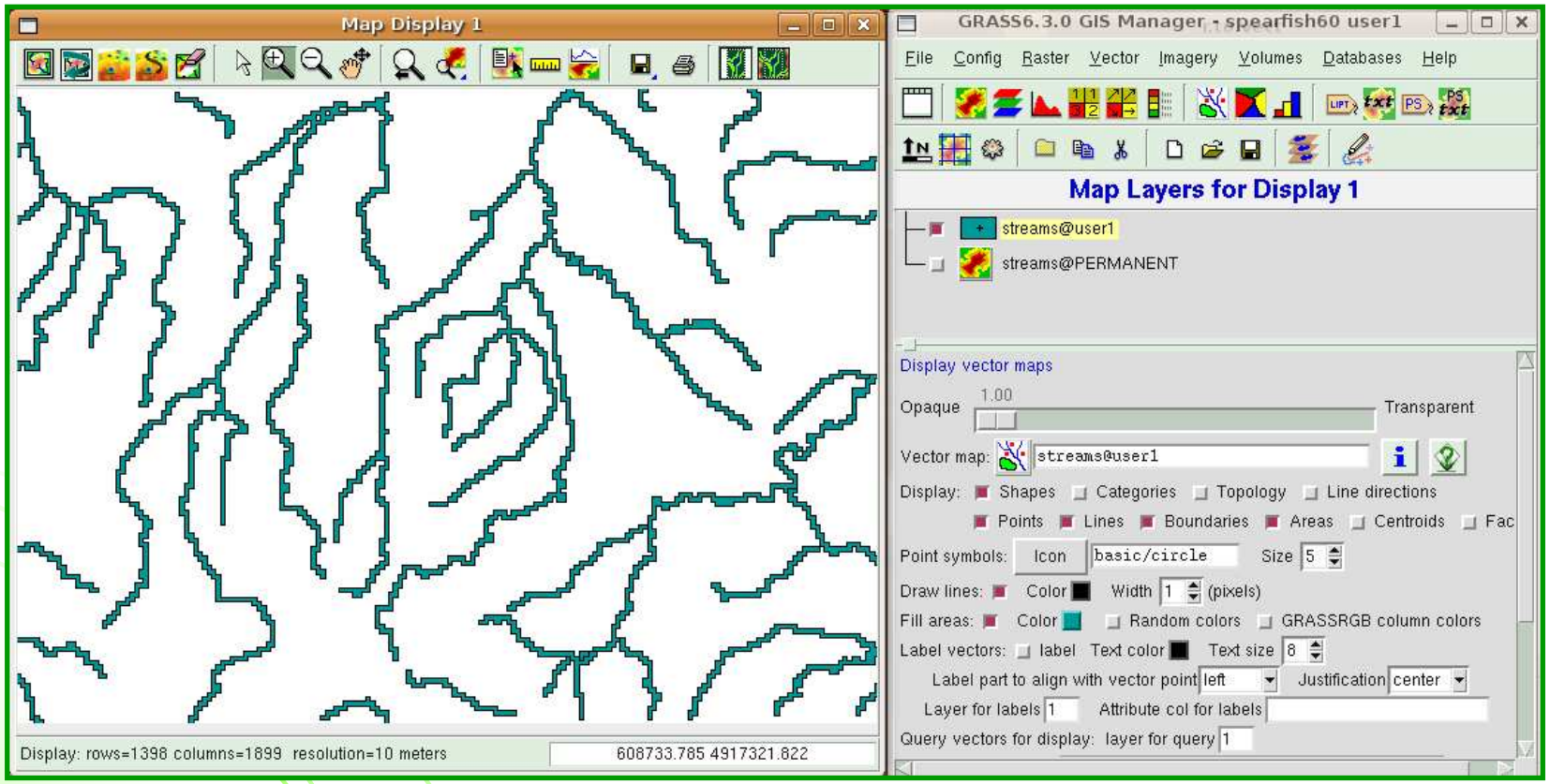
We can create a vector map starting from '*streams*' raster map using the command **r.to.vect**:

Also in this case you have to check region settings. ??



r.to.vect

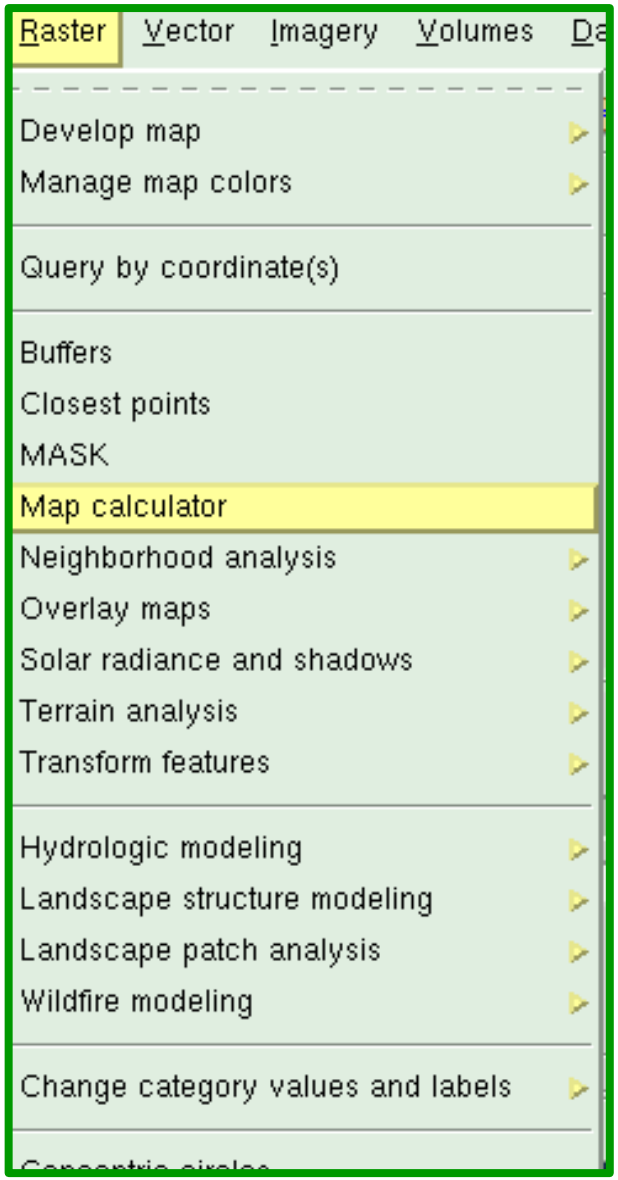
From raster to vector - 2



Raster – Map algebra

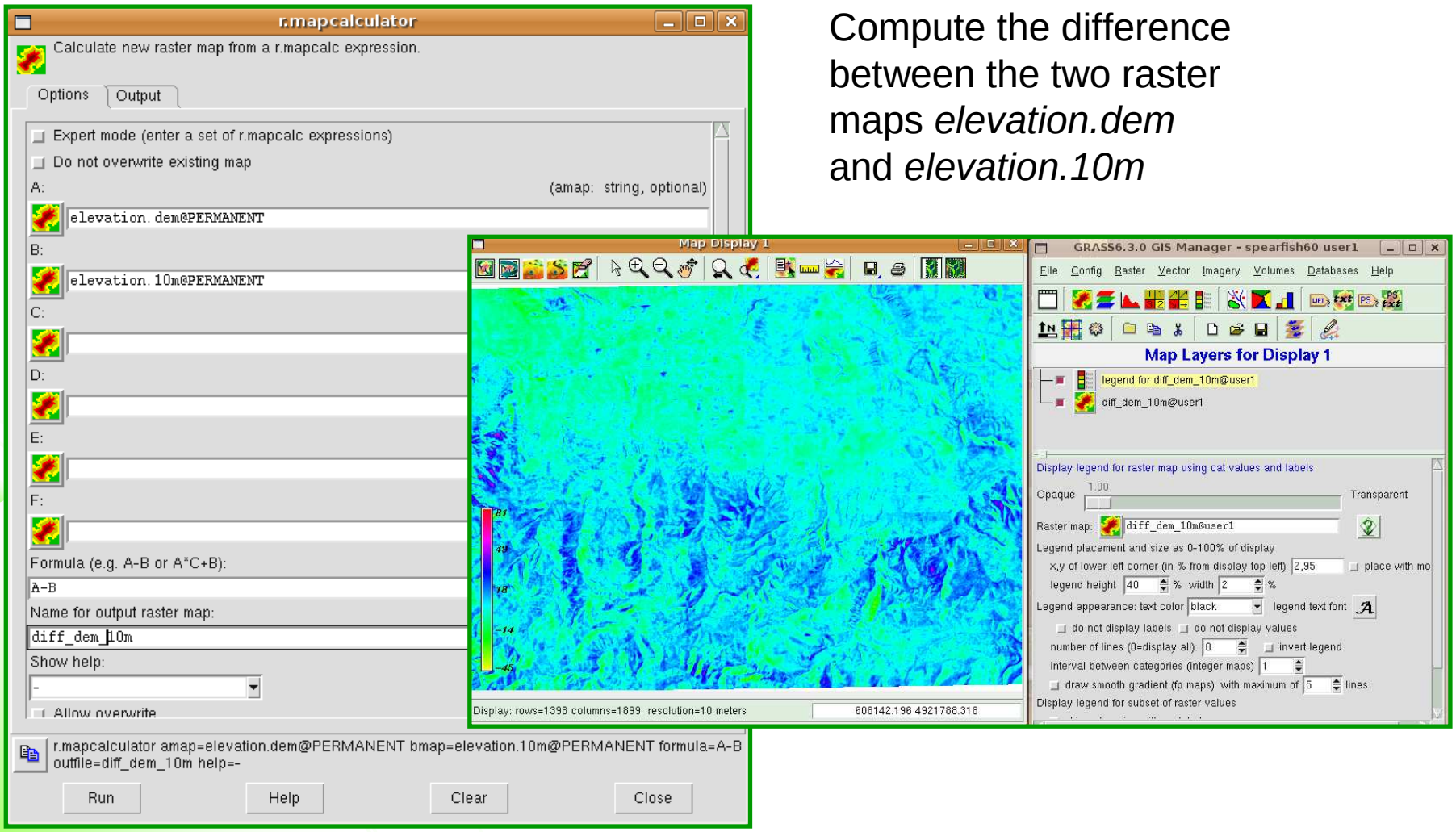
GRASS does map algebra with
`r.mapcalc`

Simplified version:
`r.mapcalculator`



Raster – Map algebra - 2

r.mapcalculator



Compute the difference between the two raster maps *elevation.dem* and *elevation.10m*

Raster – Map algebra - 3

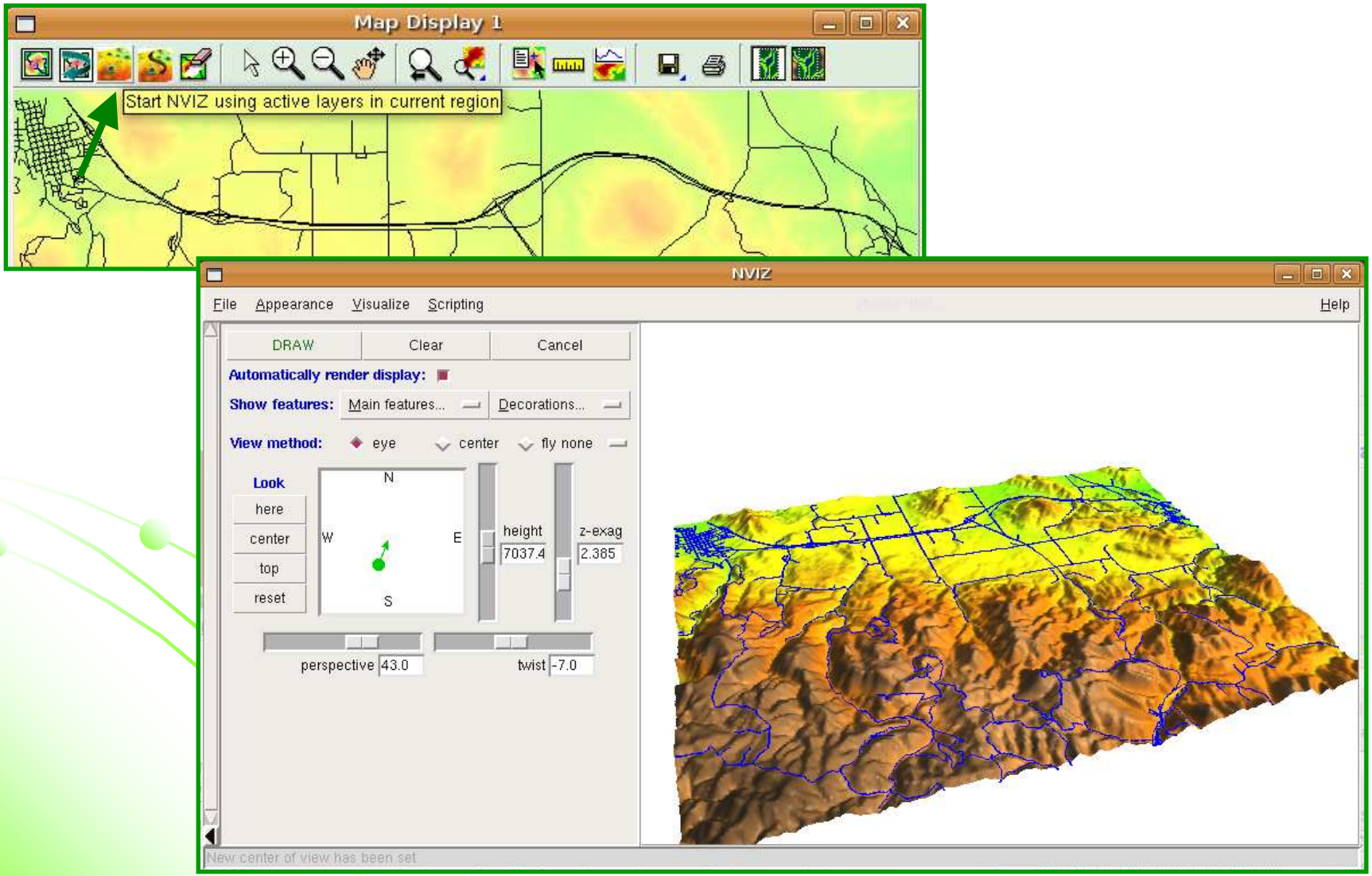
Logical operators

Operator	Meaning	Type
!	not	Logical
^	exponentiation	Arithmetic
%	modulus	Arithmetic
/	division	Arithmetic
*	multiplication	Arithmetic
+	addition	Arithmetic
-	subtraction	Arithmetic
==	equal	Logical
!=	not equal	Logical
>	greater than	Logical
>=	greater than or equal	Logical
<	less than	Logical
<=	less than or equal	Logical
&&	and	Logical
||	or	Logical
&&&	and[1]	Logical
|||	or[1]	Logical
#	color separator operator	Arithmetic

Mathematical functions

function	description
abs(x)	return absolute value of x
acos(x)	inverse cosine of x (result is in degrees)
asin(x)	inverse sine of x (result is in degrees)
atan(x)	inverse tangent of x (result is in degrees)
atan(x,y)	inverse tangent of y/x (result is in degrees)
cos(x)	cosine of x (x is in degrees)
double(x)	convert x to double-precision floating point
eval([x,y,...,]z)	evaluate values of listed expr, pass results to
exp(x)	exponential function of x
exp(x,y)	x to the power y
float(x)	convert x to single-precision floating point
graph(x,x1,y1[x2,y2..])	convert the x to a y based on points in a graph
if	decision options:
if(x)	1 if x not zero, 0 otherwise
if(x,a)	a if x not zero, 0 otherwise
if(x,a,b)	a if x not zero, b otherwise
if(x,a,b,c)	a if x > 0, b if x is zero, c if x < 0
int(x)	convert x to integer [truncates]
isnull(x)	check if x = NULL
log(x)	natural log of x
log(x,b)	log of x base b
max(x,y[,z...])	largest value of those listed
median(x,y[,z...])	median value of those listed
min(x,y[,z...])	smallest value of those listed
mode(x,y[,z...])	mode value of those listed
not(x)	1 if x is zero, 0 otherwise
pow(x,y)	x to the power y
rand(a,b)	random value x : a < x < b
round(x)	round x to nearest integer
sin(x)	sine of x (x is in degrees)
sqrt(x)	square root of x
tan(x)	tangent of x (x is in degrees)

Visualization with NVIZ



Other GRASS functionalities

- Import/Export
- Vector/raster analysis
- Rasterization / vectorization
- Overlay
- Queries
- Geometry management
- Map algebra
- Neighborhood analysis
- Interpolation
- Network analysis
- Datum shifting
- Solar radiance & shadows
- Terrain analysis
- Hydrologic modeling
- Wildfire modeling
- LiDAR analysis
- Geostatistics
- Optical image processing
- 3D analysis
- ….

Network analysis

- Methods:
  - Find shortest path along vector network – *road navigation* → `v.net.path`
  - Subdivide a network in subnetworks (iso-distances) – *how far can I go from a node in all directions* → `v.net.iso`
- Find optimal connection between nodes (Minimum Steiner tree) → `v.net.steiner`
  - Traveling salesman problem → `v.net.salesman`
  - etc...
- Allows to work with values in a attribute table

Interpolation methods

- Interpolation of vector maps in order to create rasters
  [Raster -> Interpolate surfaces]:

  - 2D inverse distance weighted → `v.surf.idw`
  - 2D regularized splines with tension (with cross-validation)→ `v.surf.rst`
  - 3D regularized splines with tension (with cross-validation) → `v.vol.rst`
  - Bilinear and Bicubic splines with Tychonov regularizator → `v.surf.bspline`
  - 2D/3D kernel densities → `v.kernel`
  - Kriging, predictive models (R-Stats)

Volume processing

GRASS is able to process and display volumes (3D voxels)

Functionality:

- 3D Import/Export
- Interpolation with 3D regularized splines with tension
- 3D Algebra
- Volumes visualization with NVIZ: Iso-surfaces and profiles

Import/Export data

- GRASS always import the complete map
- Exports raster maps within the **current region**
- Support a lot of vector and raster formats:

Vectorial (`v.in.*, v.out.*`)

- **OGR:** ESRI Shapefile, UK .NTF, SDTS, TIGER, S57, MapInfo File, DGN, VRT, AVCBin, REC, Memory, CSV, GML, KML, ODBC, Pgeo, PostgreSQL/PostGIS, OGDI, ecc...
- ArcInfo E00
- ASCII
- DXF
- ...

Raster (`r.in.*, r.out.*`)

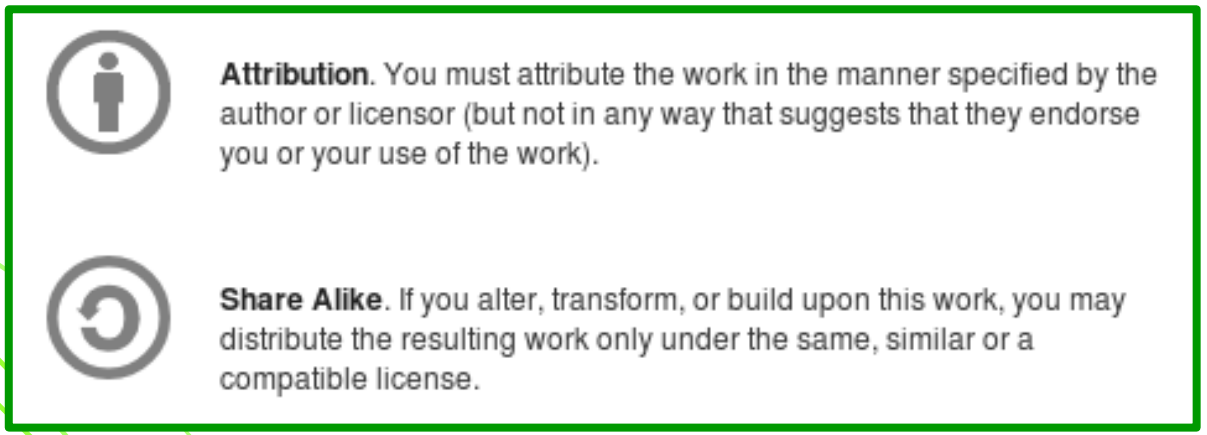
- **GDAL:** ArcInfo, CEOS, DOQ, DTED, ENVI, Envisat, Erdas Img/LAN, FAST, (Geo)TIFF, HDF4, SAR, SDTS, ecc...
- ASCII
- Bin: binary, BIL, GMT, ecc...
- Matlab file
- SRTM
- ...

Who's using GRASS?

Document license

This work is released under a 'Creative Commons License'

`http://creativecommons.org/licenses/by-sa/3.0/`



Bibliography

“Introduction to GIS GRASS and QGIS” ---- 2007 Roberto Antolín, Spain
“Introduction to GRASS GIS” ---- 2008 Eugenio Realini

GRASS tutorial (in italian)
`http://www.ing.unitn.it/~grass/docs/tutorial_62/index.html`

GRASS manual (in english)
`http://grass.itc.it/grass63/manuals/html63_user/full_index.html`