GRASS Workshop

Open Source Geospatial '05 Conference

presented by

Markus Neteler – Kristen Perry

M. Neteler
neteler at itc it
http://mpa.itc.it

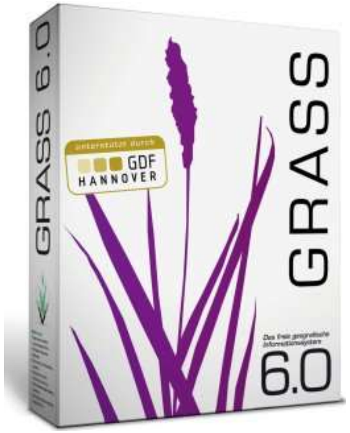
ITC-irst, Povo (Trento), Italy

K. Perry
kperry at spatialfocus com
http://www.spatialfocus.com

USA

GRASS: Geographic Resources Analysis Support System

- Free Software GIS ("software libero"):
- GRASS master Web site is in Italy:
  http://grass.itc.it
- *Portable:* Versions for GNU/Linux, MS-Windows, Mac OSX, SUN, etc
- *Programming:* Programmer's Manual on Web site (PDF, HTML), generated weekly. Code is documented in source code files (doxygen)
- Sample data
- Mailing lists in various languages
- Commercial support

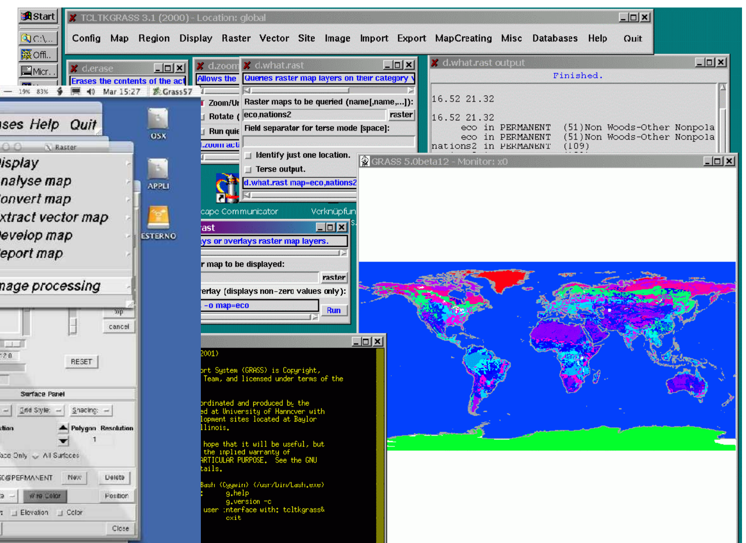


GRASS: Geographic Resources Analysis Support System

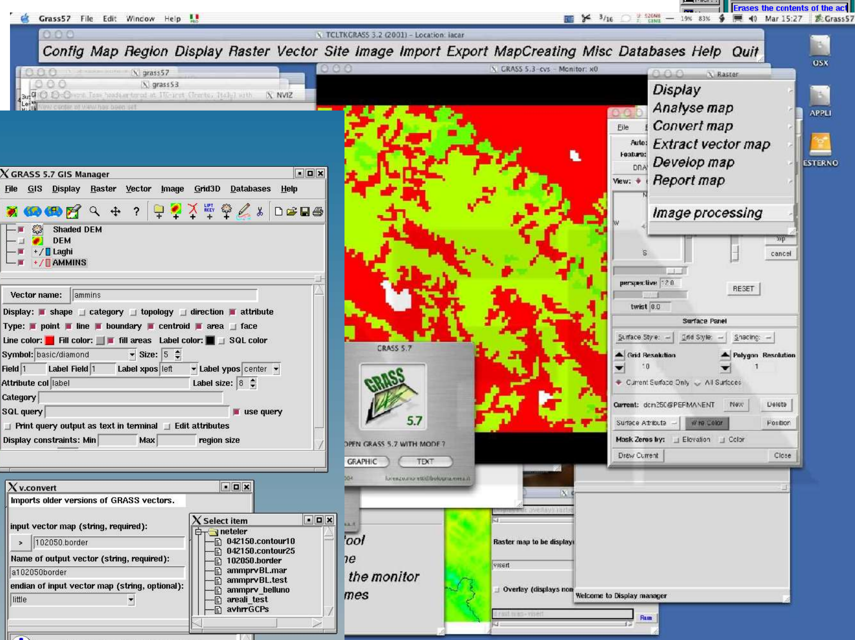
Brief Introduction – Development and System Requirements

- Developed since 1984, **always Open Source**, since 1999 under GNU General Public License
- Written in C programming language, **portable code** (32/64bit)
- **International development team**, since 2001 coordinated at ITC-irst
- Distributed as source code, precompiled binaries for various platforms, CDROM

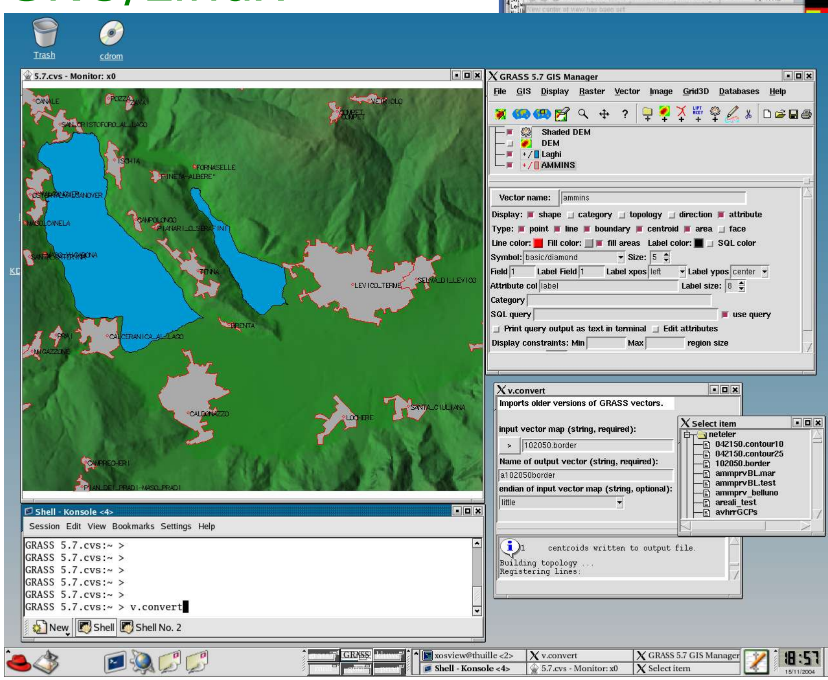
MS-Windows



MacOSX



GNU/Linux



iPAQ

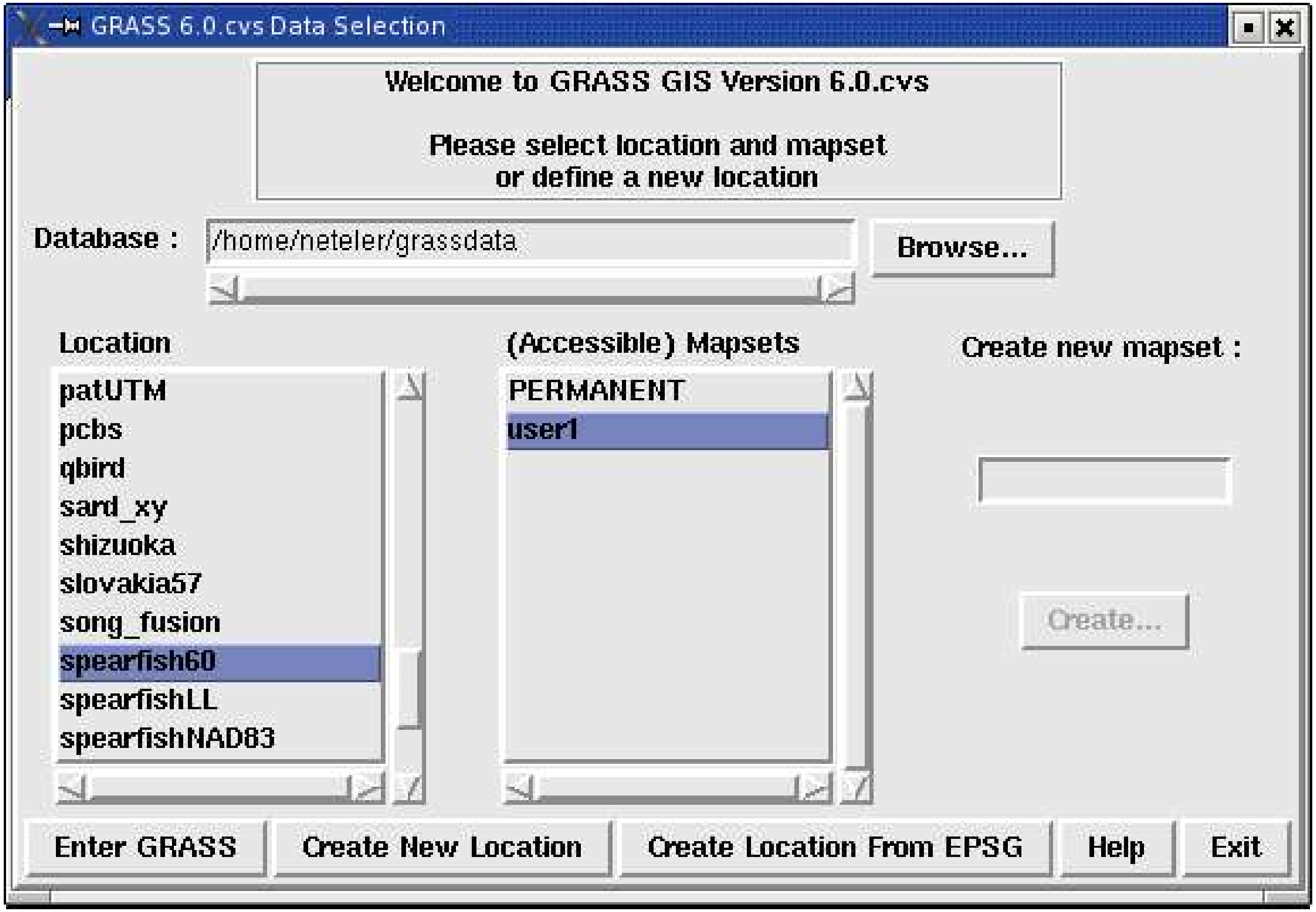
GRASS: Geographic Resources Analysis Support System

The new features of GRASS 6:

- New topological 2D/3D vector engine
- Support for vector analysis
- Attributes managed in an SQL-based DBMS
- New display manager and GRASS interface in QGIS
- Enhanced NVIZ 3D vector tool
- Integrated with GDAL/OGR libraries

Kristen Perry - Spatial Focus, Inc.

GRASS: Geographic Resources Analysis Support System

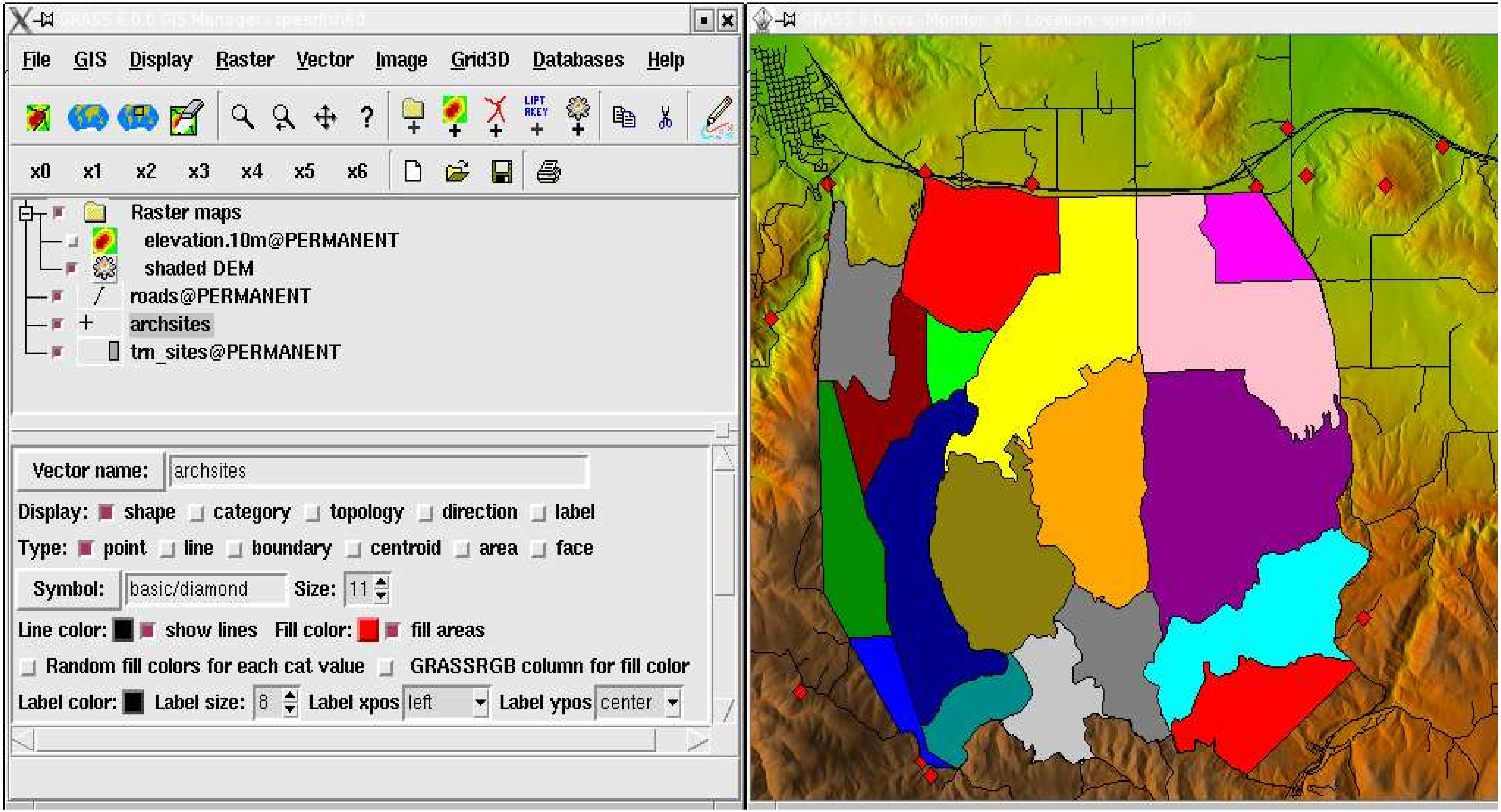


GRASS startup screen

GRASS: Geographic Resources Analysis Support System

Creating a location from an EPSG code

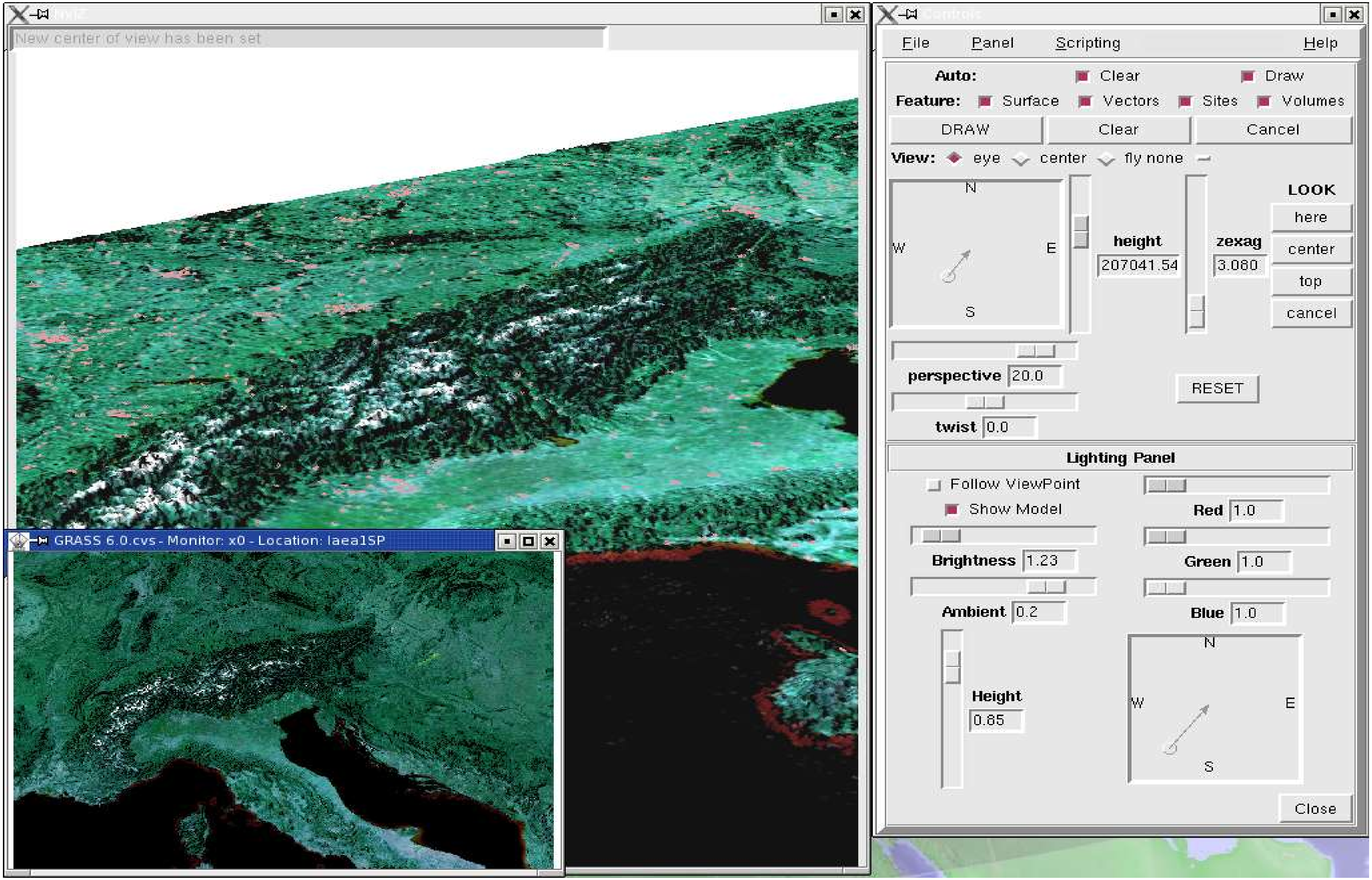
GRASS: Geographic Resources Analysis Support System



The default GUI is the GIS / Display Manager (d.m)

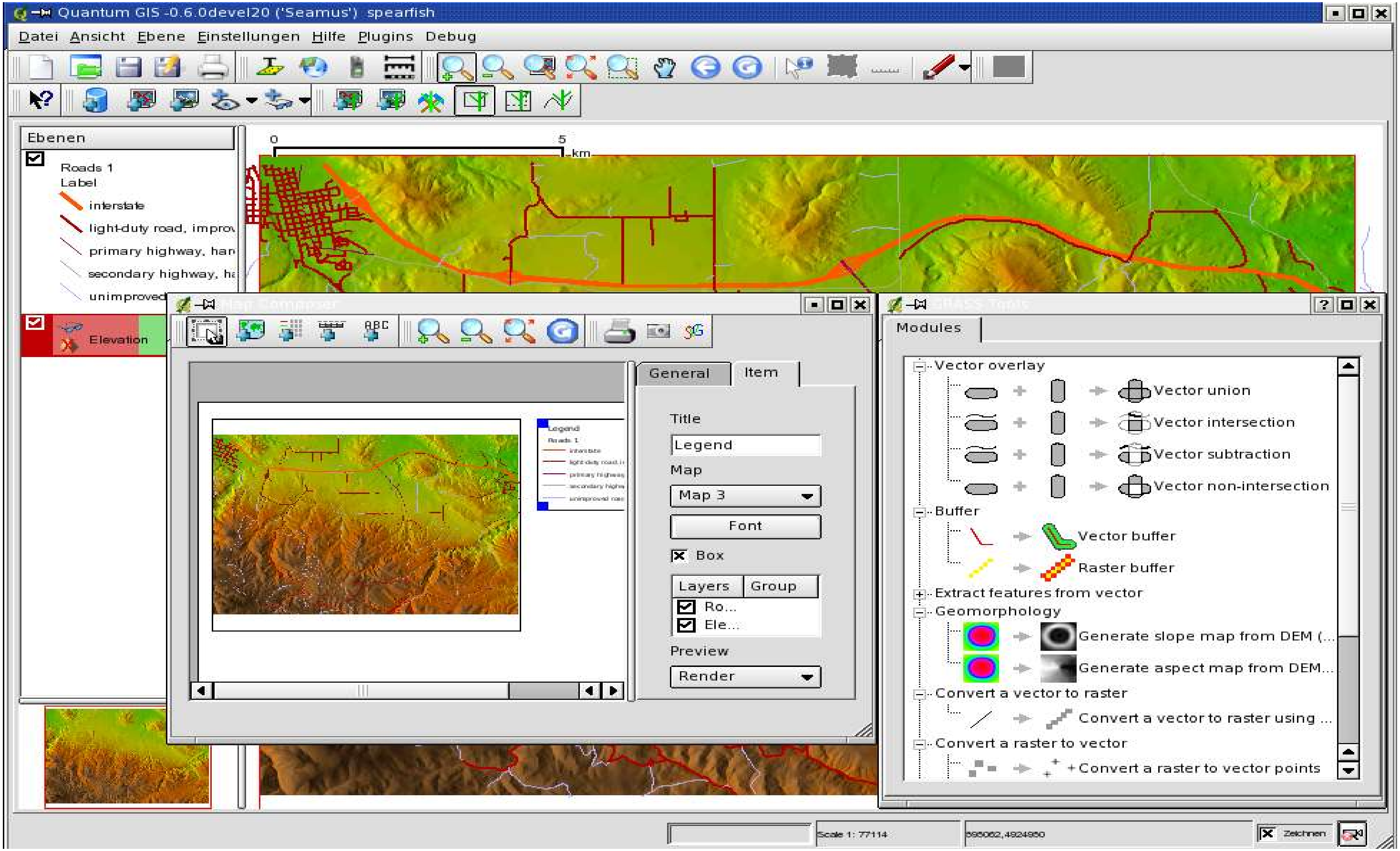
Kristen Perry - Spatial Focus, Inc.

GRASS: Geographic Resources Analysis Support System



NVIZ: Perspective 2D Maps

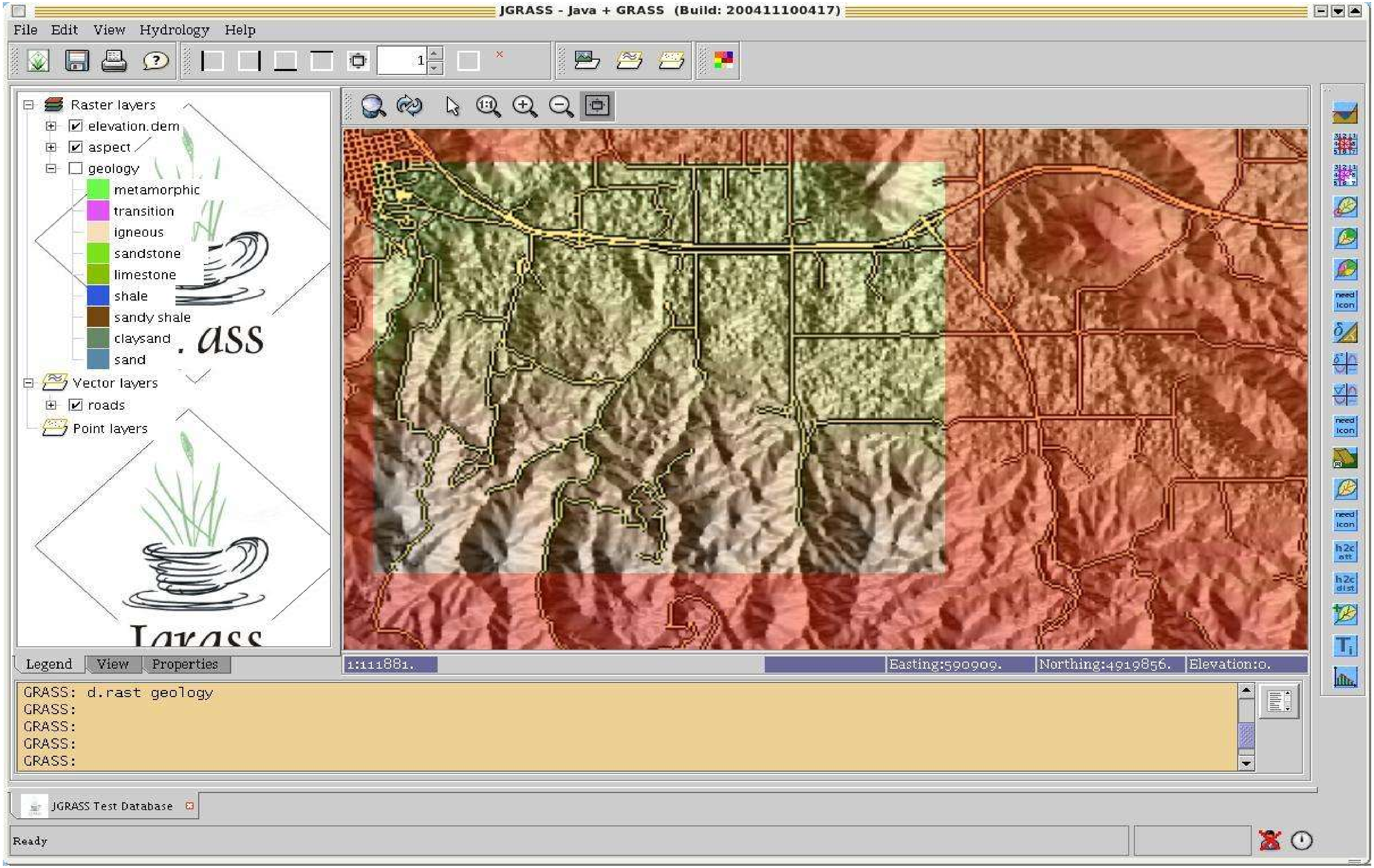
GRASS: Geographic Resources Analysis Support System



QGIS geodata browser: integrating GRASS functionality

Kristen Perry - Spatial Focus, Inc.

GRASS: Geographic Resources Analysis Support System



JAVAGRASS

GRASS: Geographic Resources Analysis Support System

GRASS Command Overview			
prefix	*function class*	*type of command*	*example*
d.*	display	graphical output	d.rast: views raster map d.vect: views vector map
db.*	database	database management	db.select: selects value(s) from table
g.*	general	general file operations	g.rename: renames map
i.*	imagery	image processing	i.smap: image classifier
ps.*	postscript	map creation in Postscript format	ps.map: map creation
r.*	raster	raster data processing	r.buffer: buffer around raster features r.mapcalc: map algebra
r3.*	voxel	raster voxel data processing	r3.mapcalc: volume map algebra
v.*	vector	vector data processing	v.overlay: vector map intersections

Online help: Help button and g.manual

Help can be found at different levels:

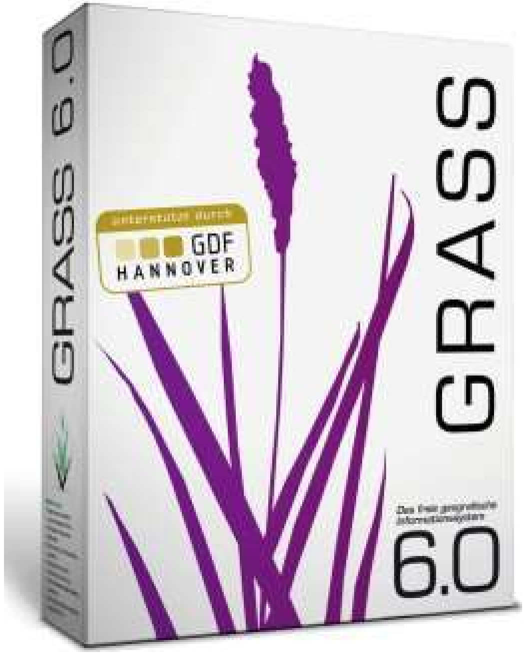
- **Launching a GRASS command without parameter (in most cases) opens a graphical window:**
  <command>   e.g., d.rast
  At the bottom a HELP button is provided.

- **Flags and parameters of a GRASS command you get with:**
  <command> -help   e.g., d.rast -help

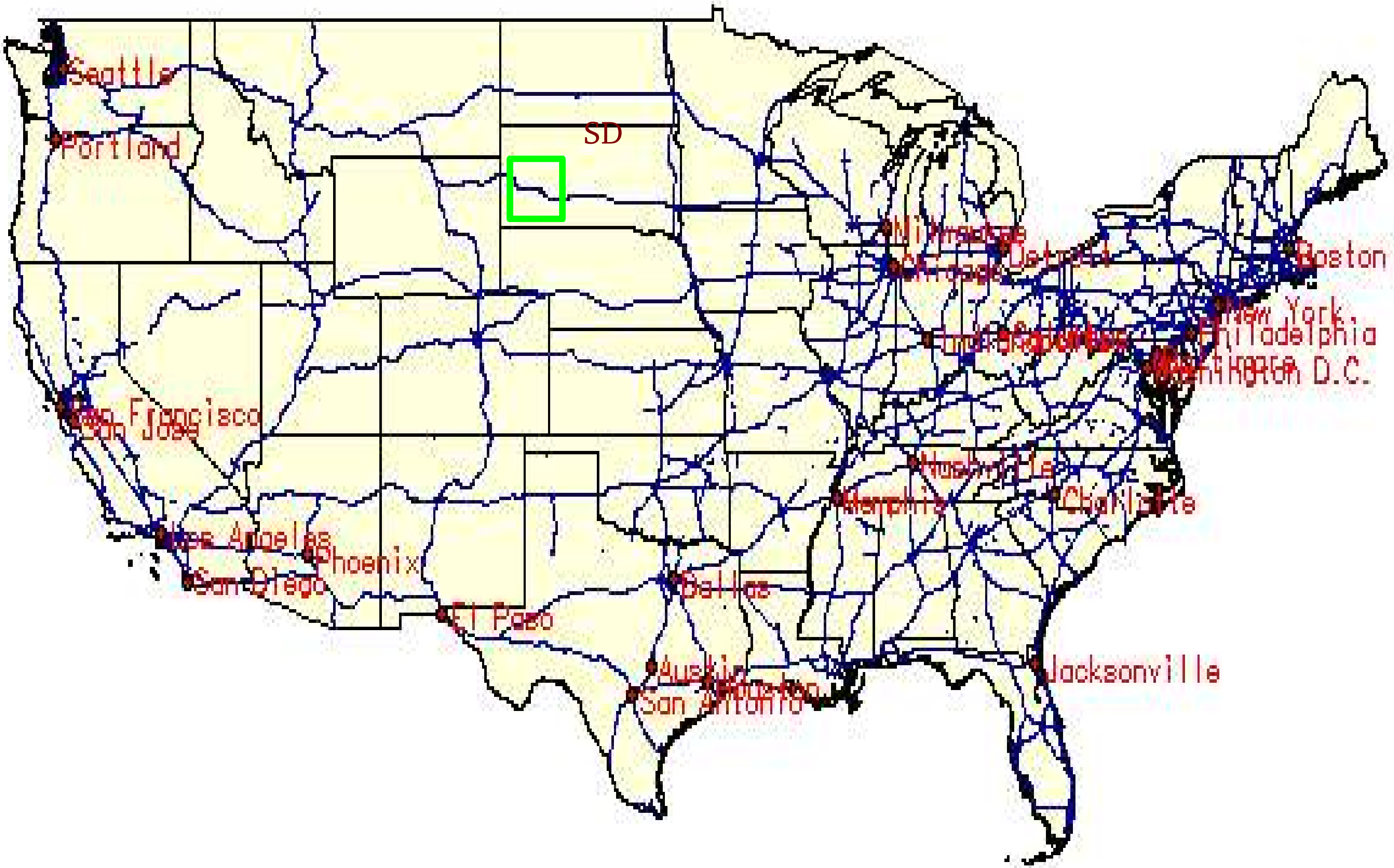
- **The manual page in a web browers you get with:**
  g.manual <command>   e.g., g.manual d.rast

- **The manual page in MAN style you get with:**
  g.manual -m <command>   e.g., g.manual -m d.rast

Section 2: Introduction to GRASS

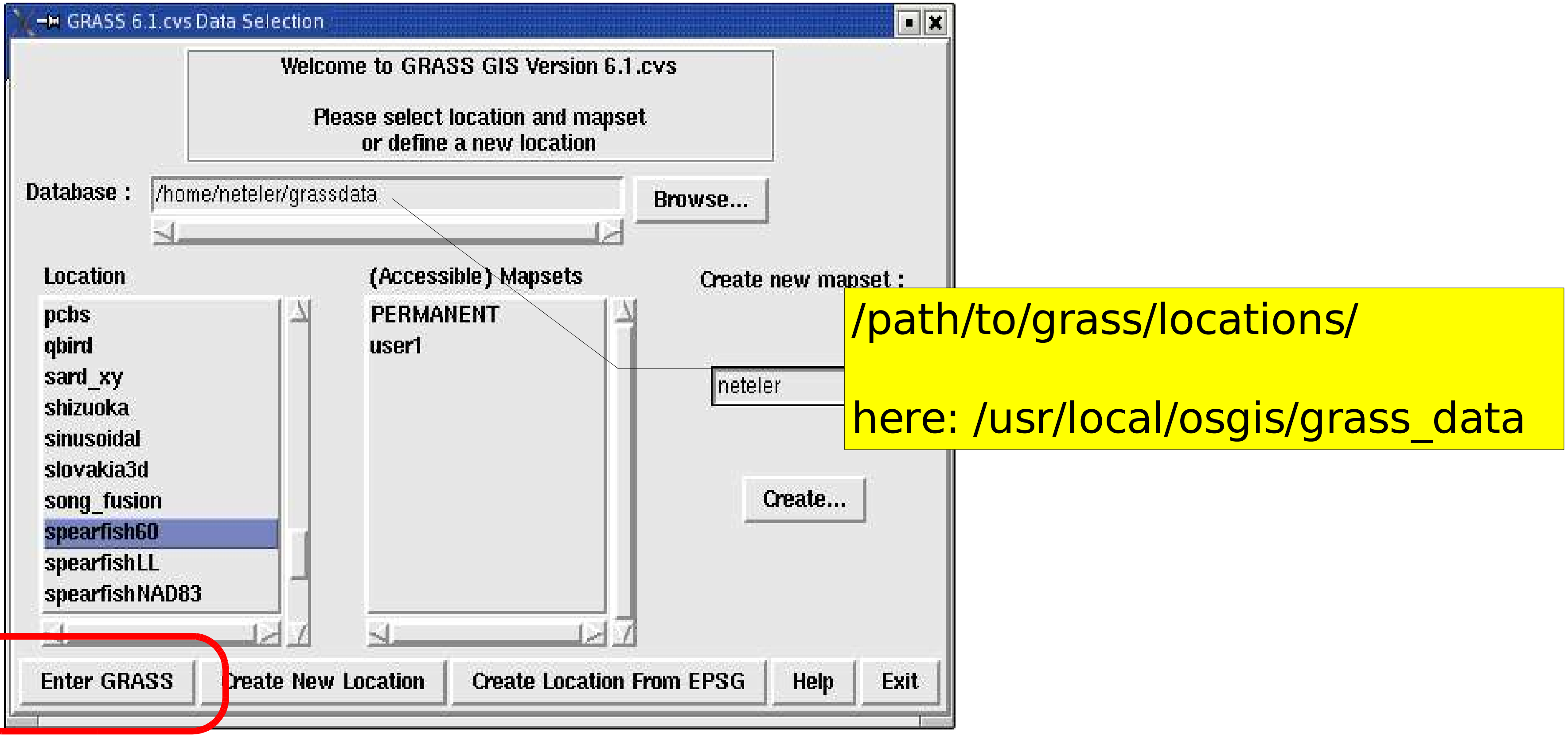


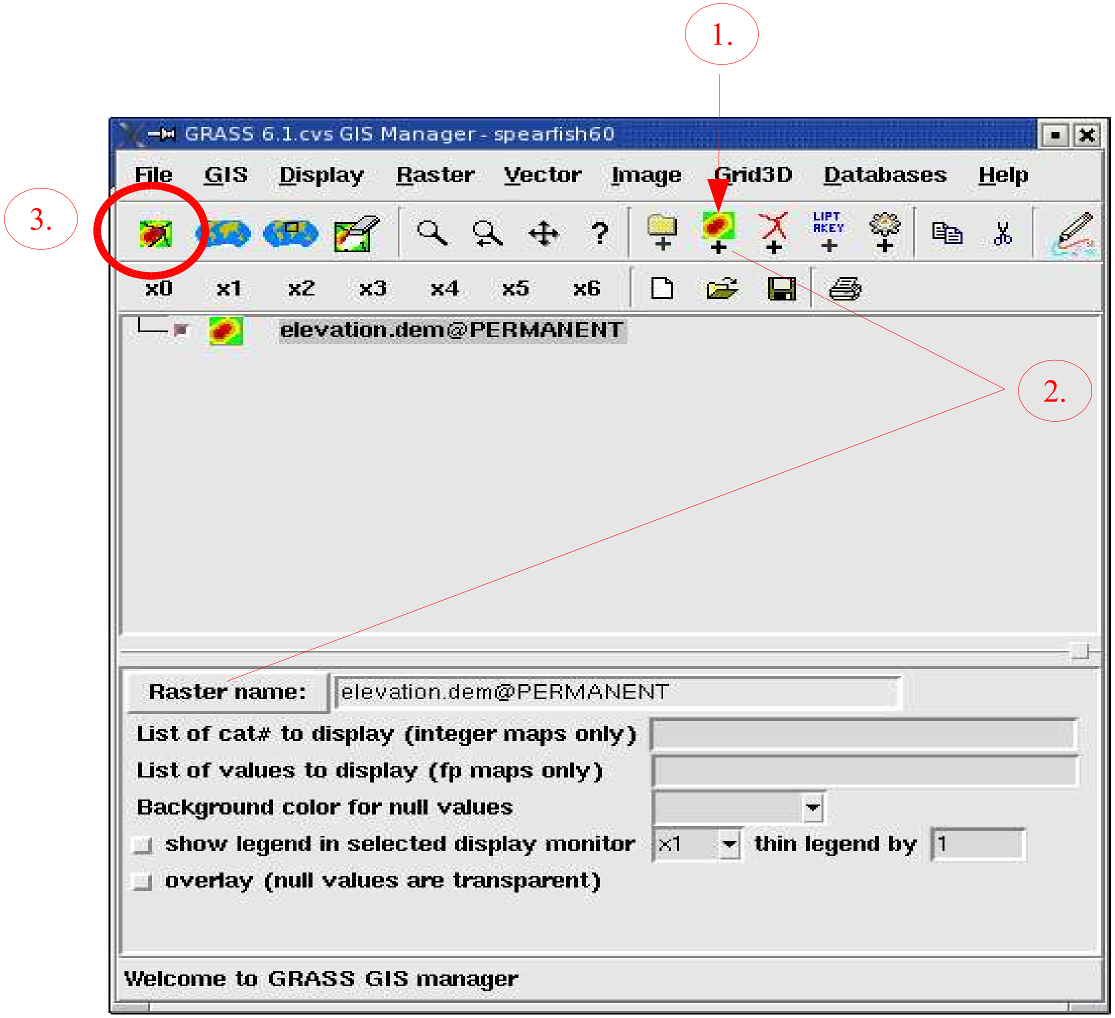
GRASS: Geographic Resources Analysis Support System



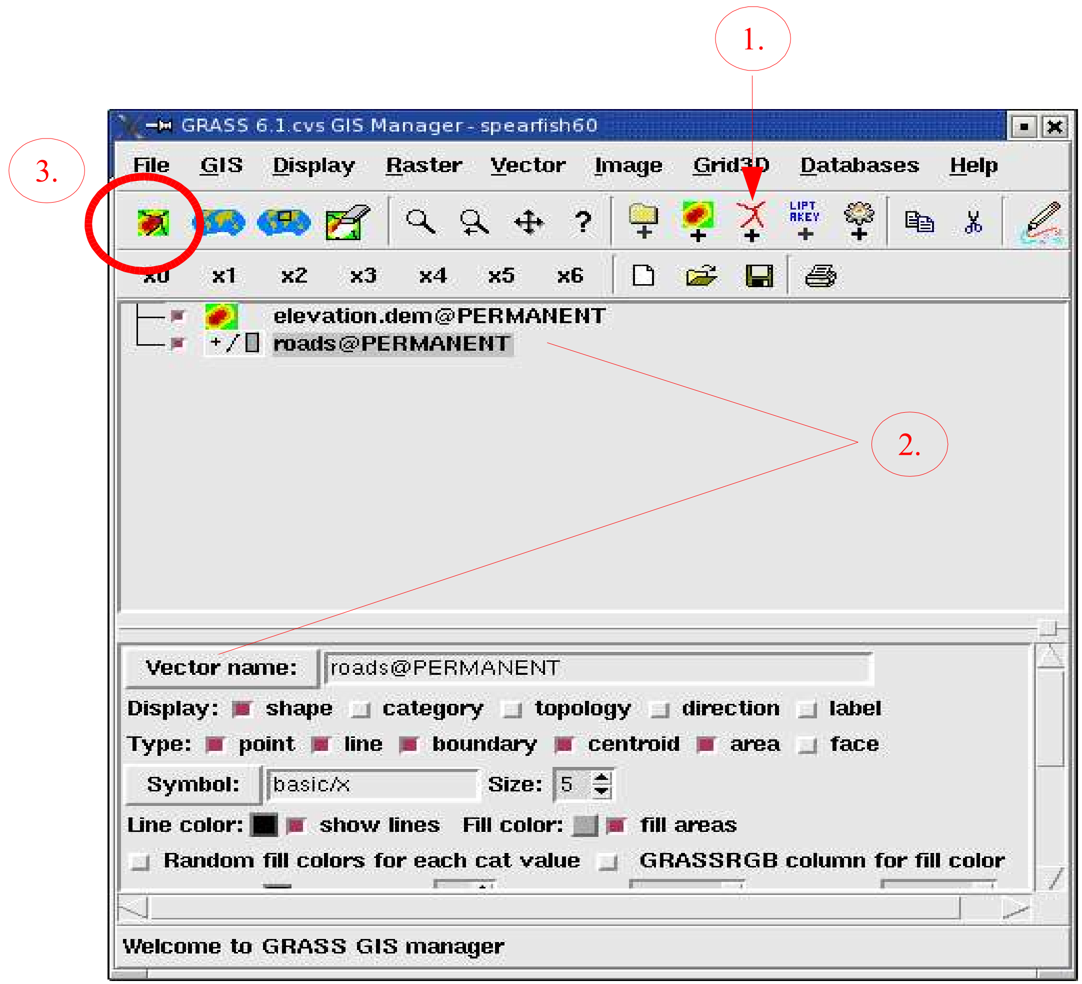
Spearfish (SD) sample data location

GRASS: Geographic Resources Analysis Support System



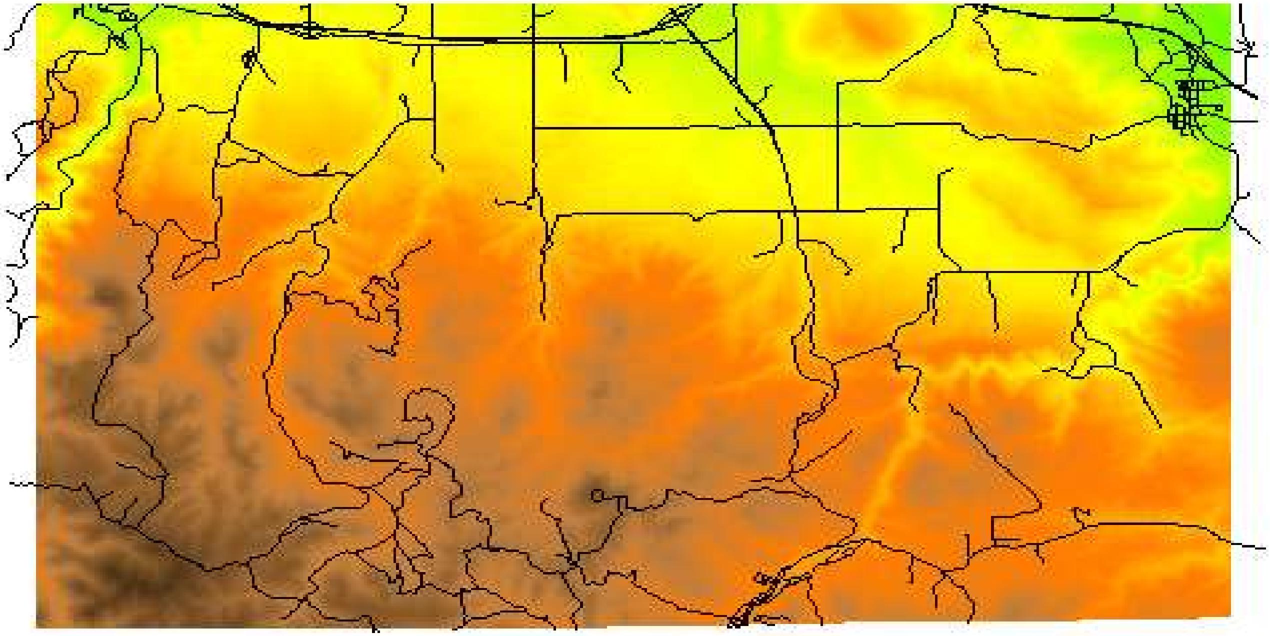


Adding the elevation.dem raster map



Adding the 'roads' vector map

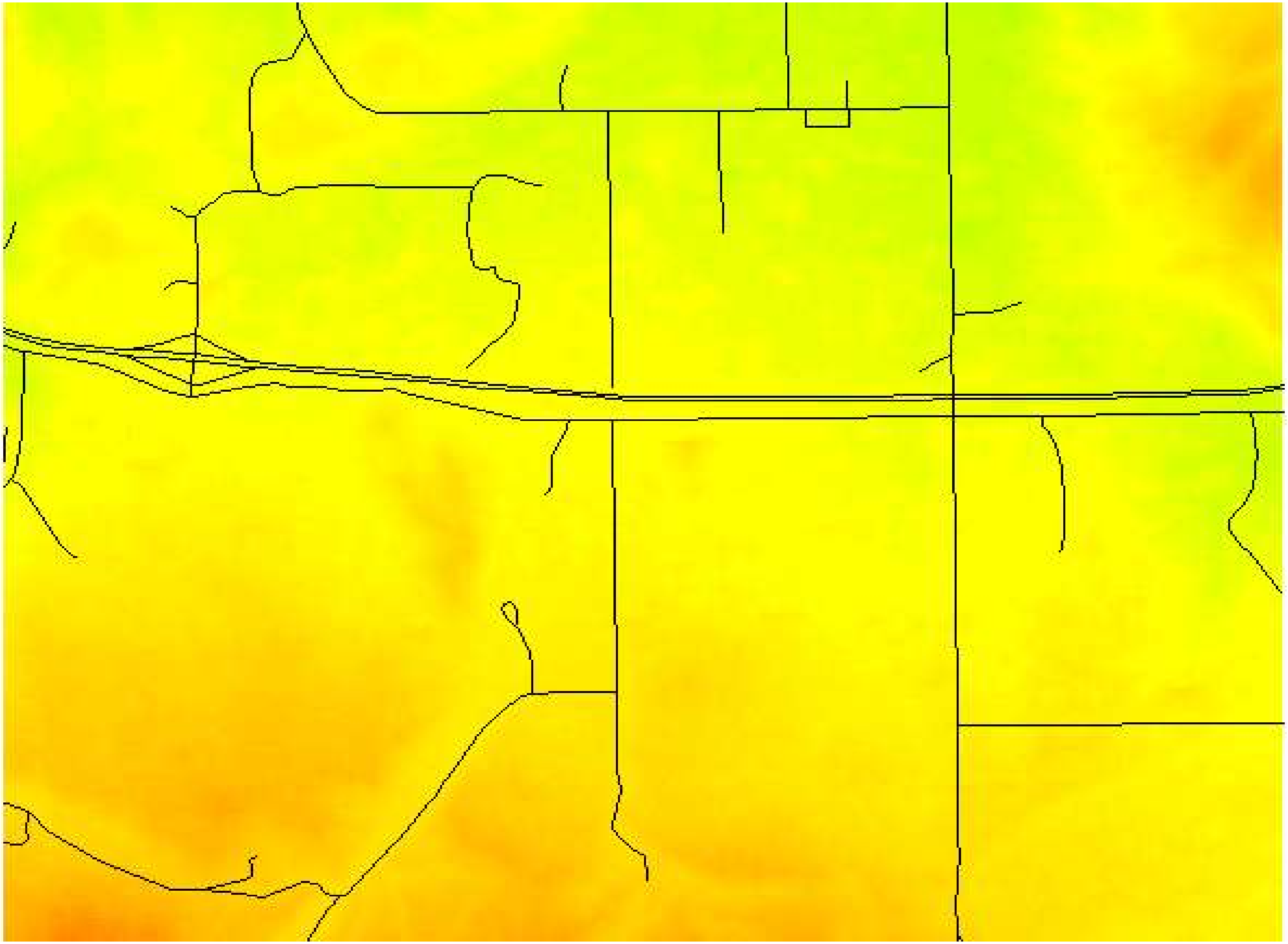
Kristen Perry - Spatial Focus, Inc.



What should display in the monitor

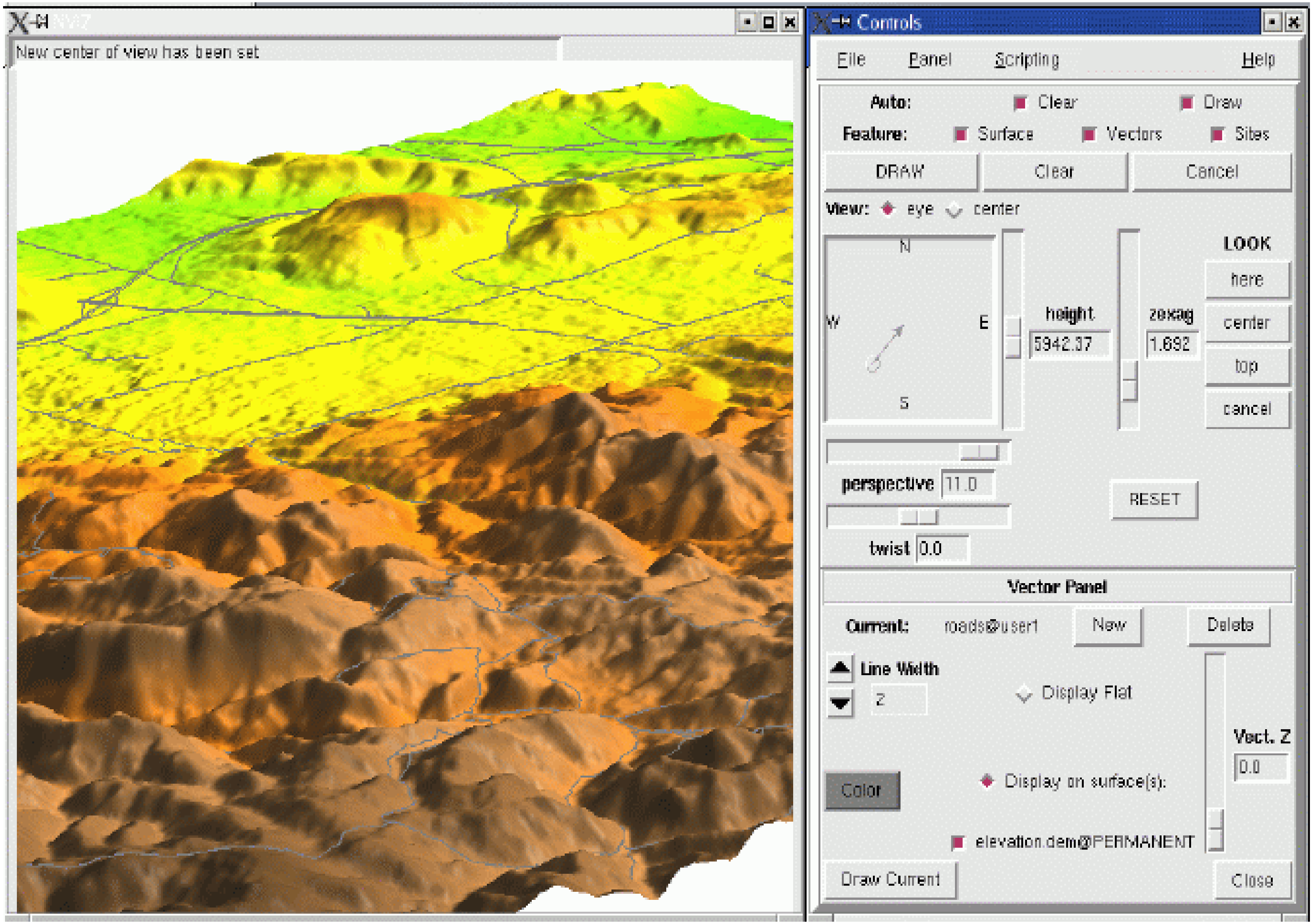
Kristen Perry - Spatial Focus, Inc.

GRASS: Geographic Resources Analysis Support System



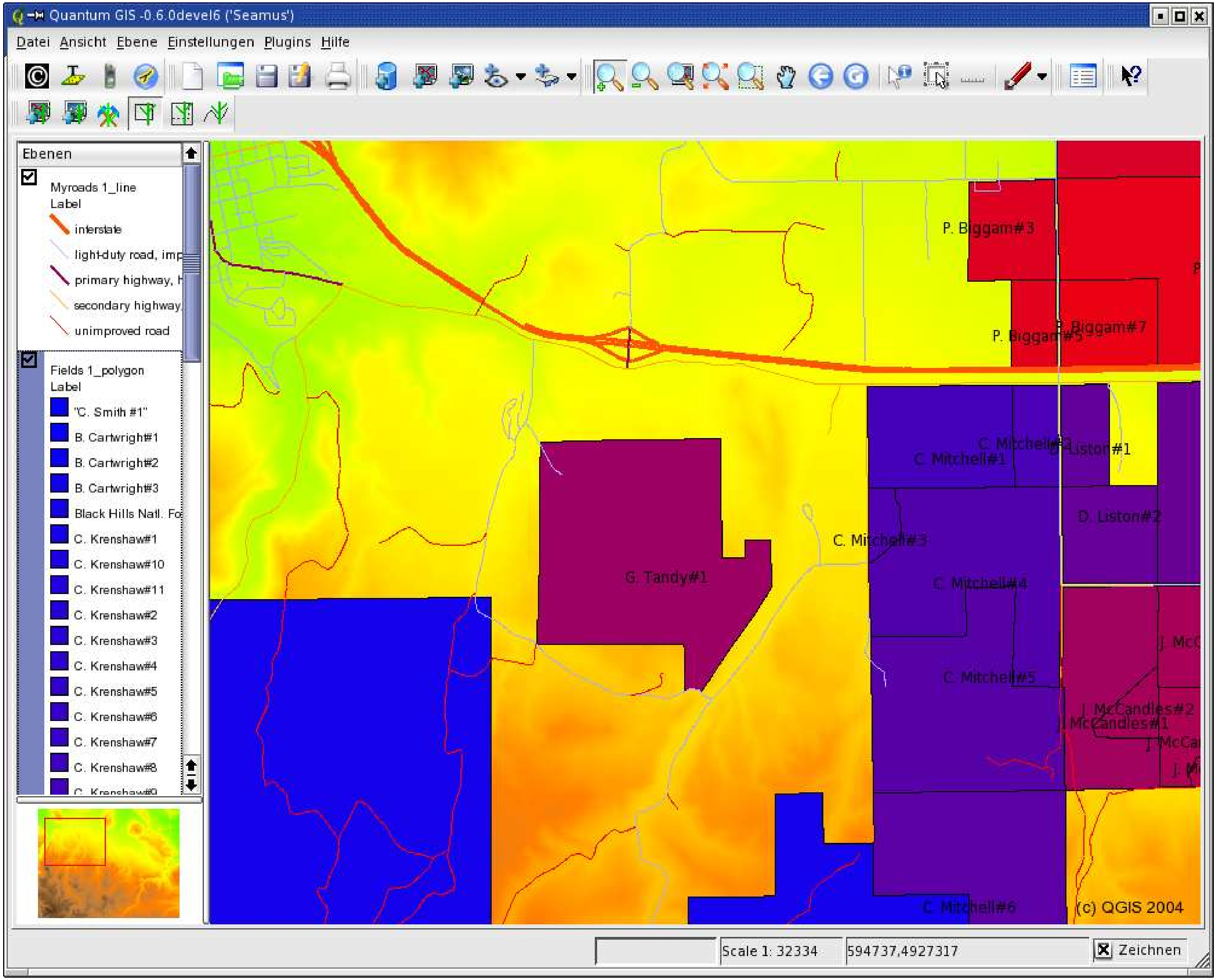
An approximate view of your saved region

GRASS: Geographic Resources Analysis Support System



NVIZ

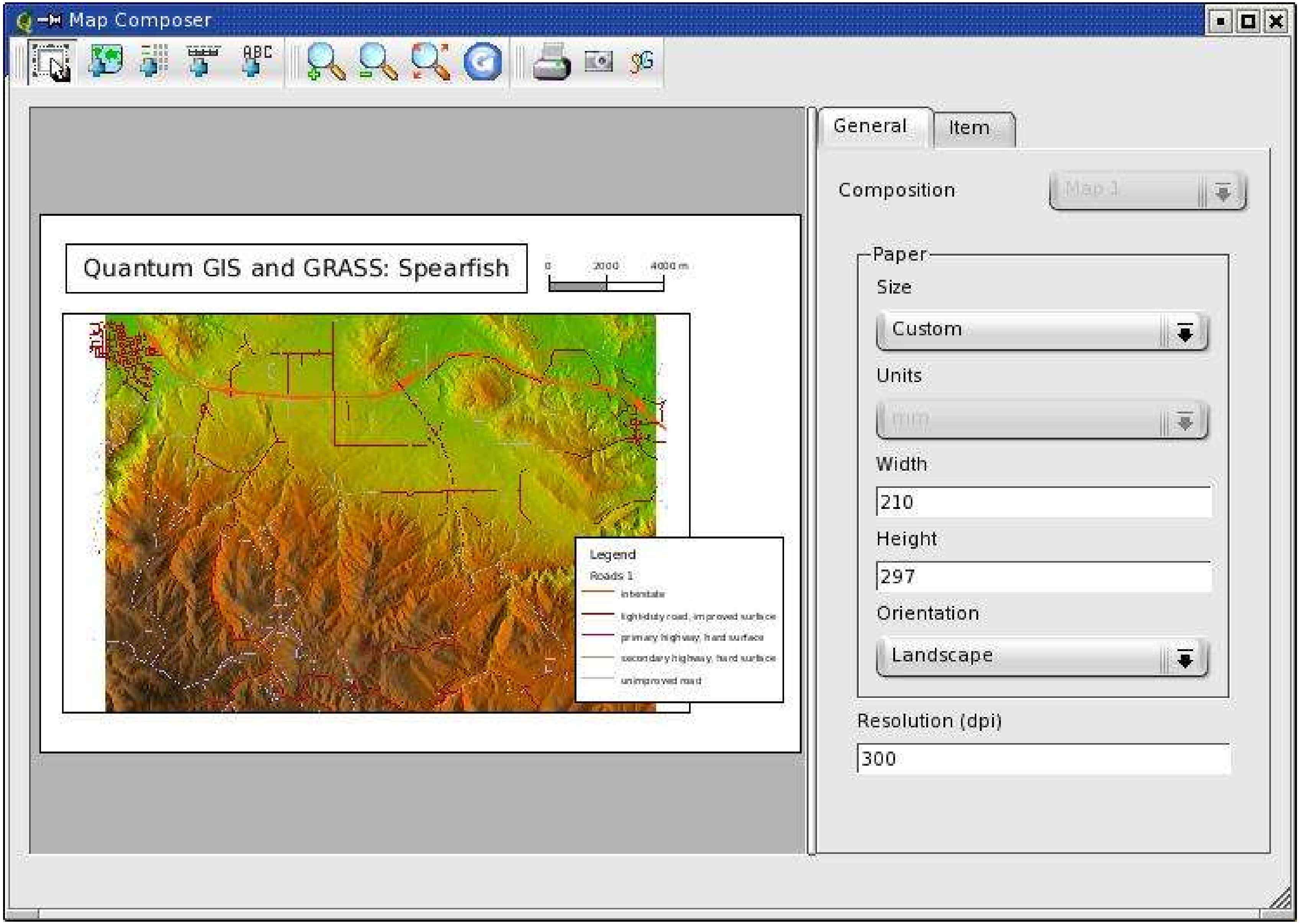
GRASS: Geographic Resources Analysis Support System



QGIS

Kristen Perry - Spatial Focus, Inc.

GRASS: Geographic Resources Analysis Support System



Creating a paper map

Kristen Perry - Spatial Focus, Inc.

To close GRASS

First close the QGIS window and display manager.

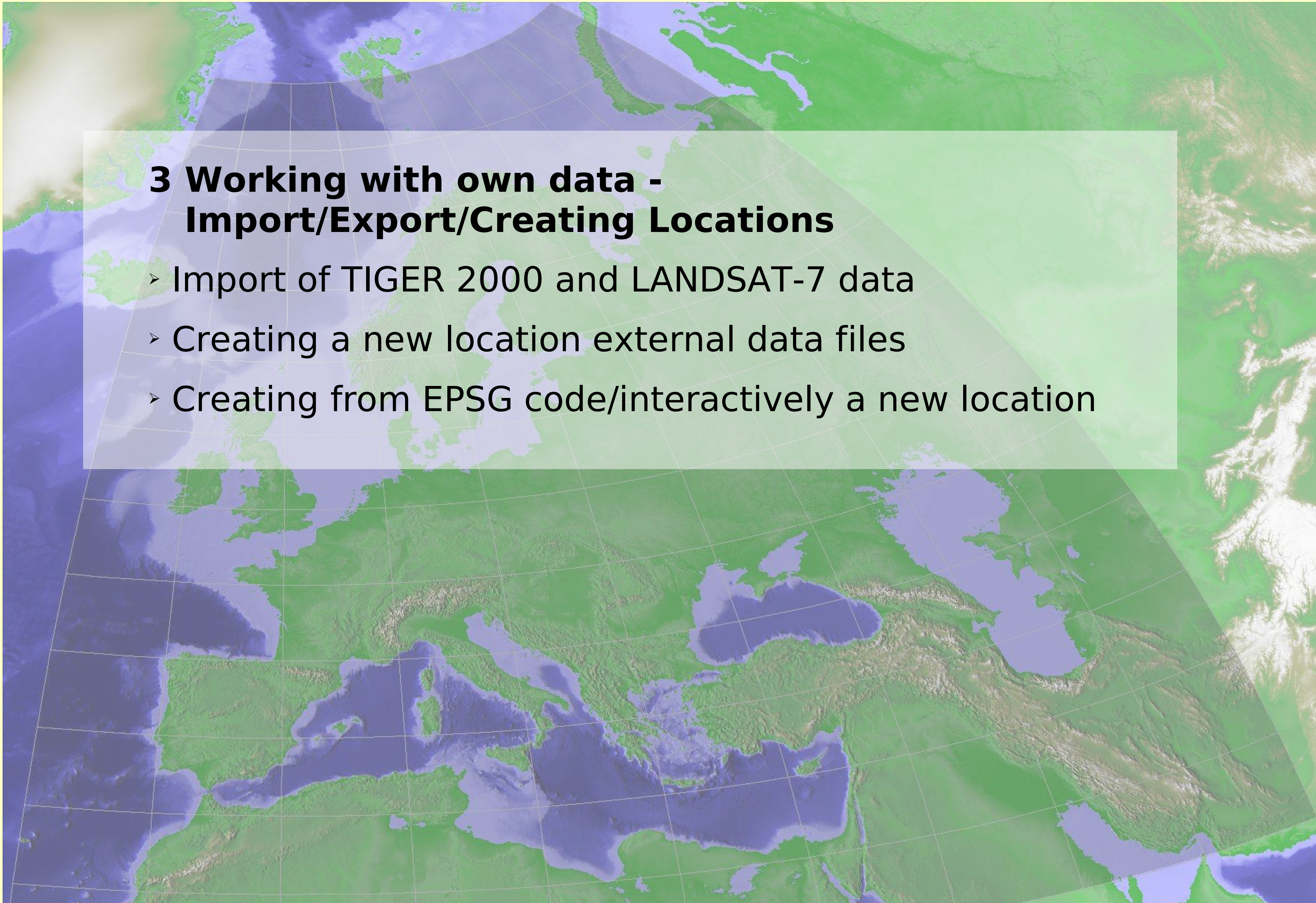
Type exit in the terminal to leave GRASS.

Monitors are closed automatically.

Outline of part II a

3 Working with own data - Import/Export/Creating Locations

- Import of TIGER 2000 and LANDSAT-7 data
- Creating a new location external data files
- Creating from EPSG code/interactively a new location



Location and Mapset: “GRASS speech”

Database: contains all GRASS data

Each GRASS project is organized in a „Location“ directory with subsequent „Mapset(s)“ subdirectories:

Location: contains all spatial/attribute data of a geographically defined region (= project area)

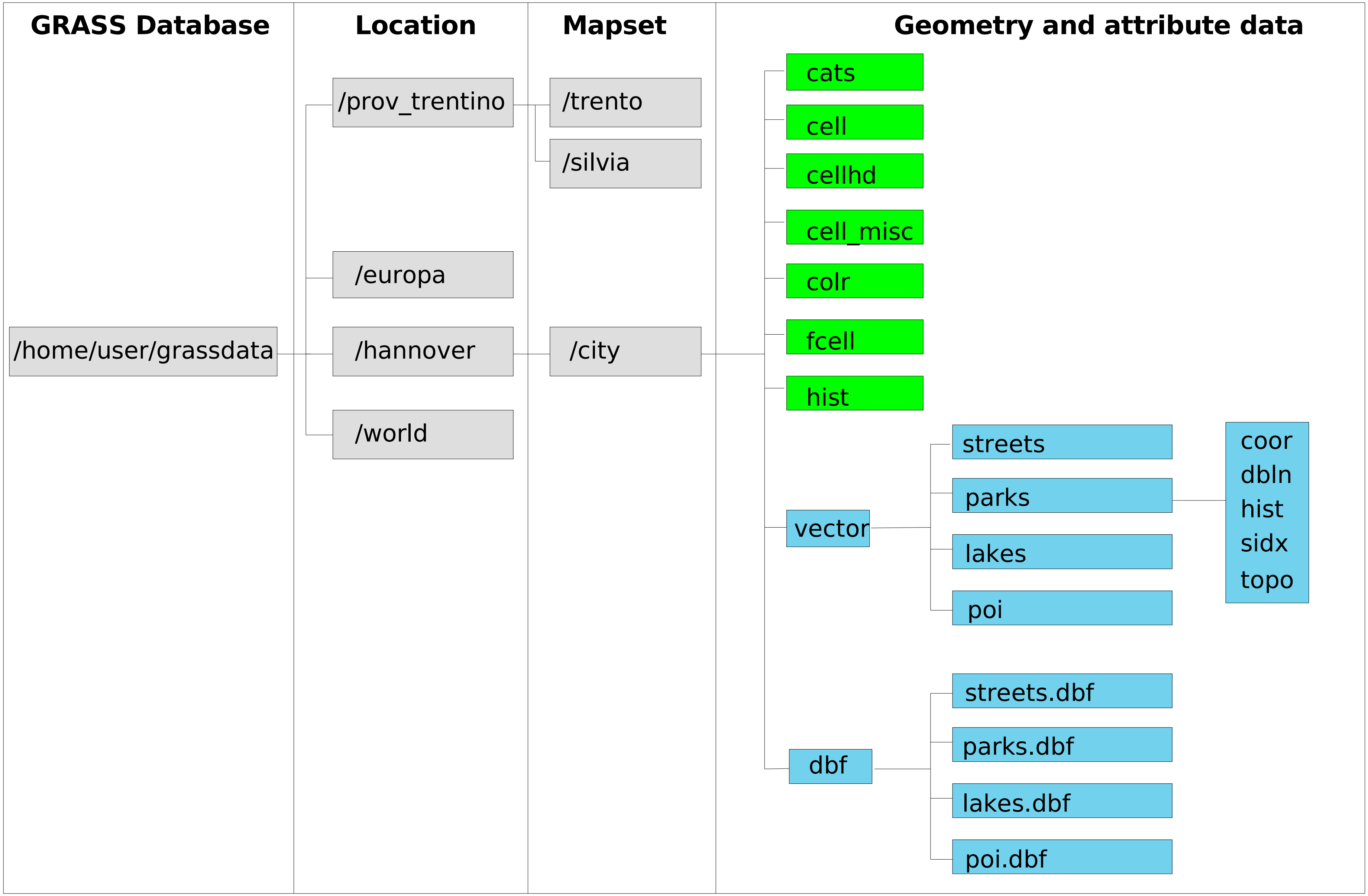
Mapset(s): used to subdivide data organization e.g. by user names, subregions or access rights (workgroups)

PERMANENT: The PERMANENT mapset is a standard mapset which contains the definitions of a location. May also contain general cartography as it is visible to all users

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.

GRASS: Geographic Resources Analysis Support System

Example for Location and Mapsets



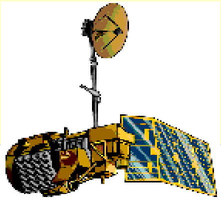
Import of TIGER 2000 vector maps

- Please unpack the **tiger2000_latlong_nad83.tar.gz** file
- To match the GRASS Spearfish sample dataset definitions (UTM zone 13N, NAD27/Clarke66; EPSG code 26713), we'll reproject them with OGR:

  ```
  ogr2ogr -s_srs epsg:4269 -t_srs epsg:26713 \
                  tgr46081lkA_UTM13_nad27.shp tgr46081lkA.shp
  ```

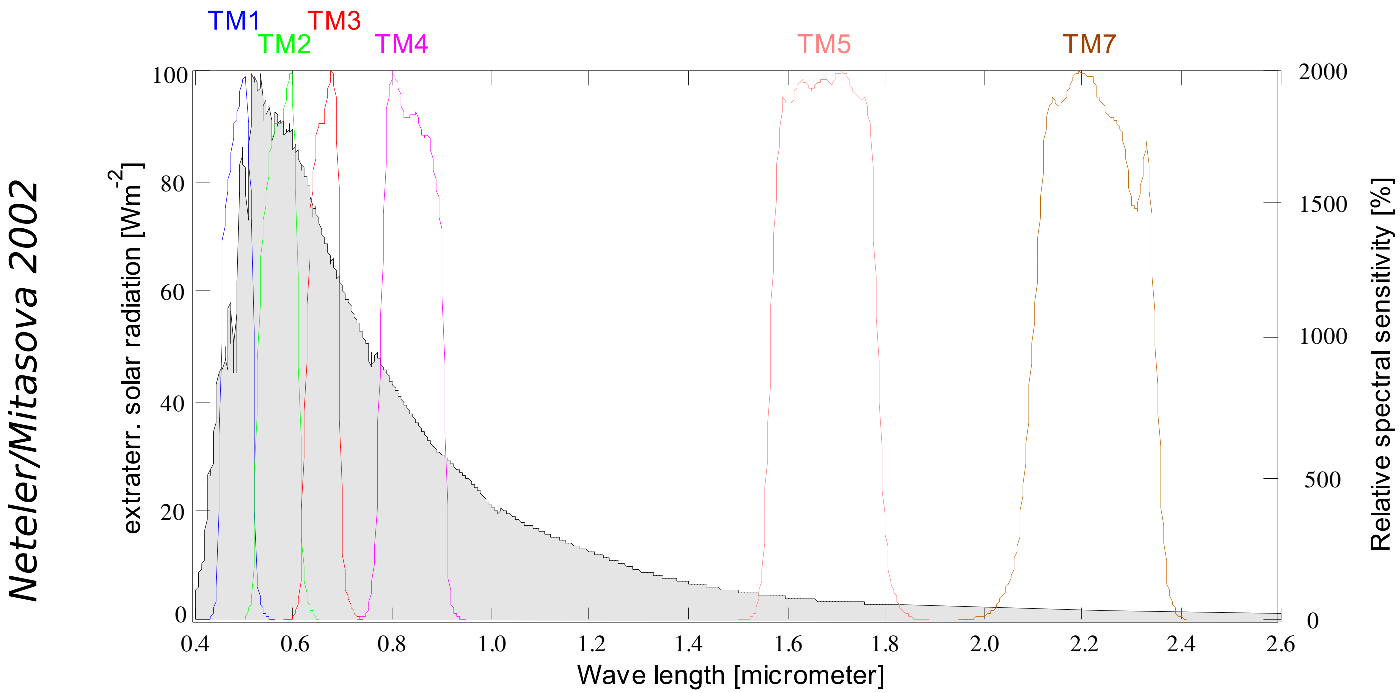
- Reproject the maps tgr46081lkA... (roads) and tgr46081lkH... (hydro)
- Now start GRASS again: **grass60**
  with Spearfish location
- The import of the roads and the hydro maps is done with **v.in.ogr**
  (dsn: data source name is the reprojected input SHAPE file)
- View the maps with **d.vect** or **qgis**

Import of LANDSAT-7 Erdas/Img raster maps 1/2



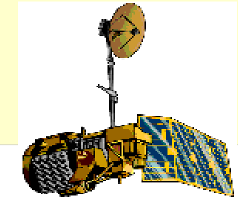
- A LANDSAT-7 scene has been prepared (reprojected, spatially subset):

  - spearfish_landsat7_NAD27_vis_ir.img:
    TM10,TM20,TM30 (blue, green, red), TM40 (NIR), TM50, TM70 (MIR)
  - spearfish_landsat7_NAD27_tir.img:
    TM62 (TIR low gain), TM62 (TIR high gain)
  - spearfish_landsat7_NAD27_pan.img:
    TM80 (panchromatic)



Solar spectrum and LANDSAT channels (thermal channel 6 not shown)

Import of LANDSAT-7 Erdas/Img raster maps 2/2



- The import is done with **r.in.gdal**:

```
r.in.gdal -e in=spearfish_landsat7_NAD27_vis_ir.img out=tm
# To keep the numbering right, we rename tm.6 to the
# correct number tm.7:
g.rename rast=tm.6,tm.7

r.in.gdal -e in=spearfish_landsat7_NAD27_tir.img out=tm6

r.in.gdal -e in=spearfish_landsat7_NAD27_pan.img out=pan
```

- Generate a RGB composite on the fly (zoom to map first):

```
g.region rast=tm.1 -p
d.rgb b=tm.1 g=tm.2 r=tm.3
```

You should see the Spearfish area in near-natural colors.

Creating new GRASS locations

- Both **r.in.gdal** and **v.in.ogr** offer a **location=** parameter to create a new location from the import dataset's metadata

  Example:
  r.in.gdal -e in=spearfish_landsat7_NAD27_tir.img out=tm6 location=utm13

- Launching GRASS (again) permits to
  - create a new location from EPSG code
  - create a new location interactively
- See the workshop handout for details

Outline of part II b

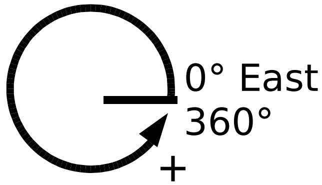
4 Raster map analysis

- DEM analysis
- Raster map algebra
- Geocoding of scanned map
- Volume data processing

Raster data analysis: Slope and aspect from DEM

- Calculating slope and aspect from a DEM

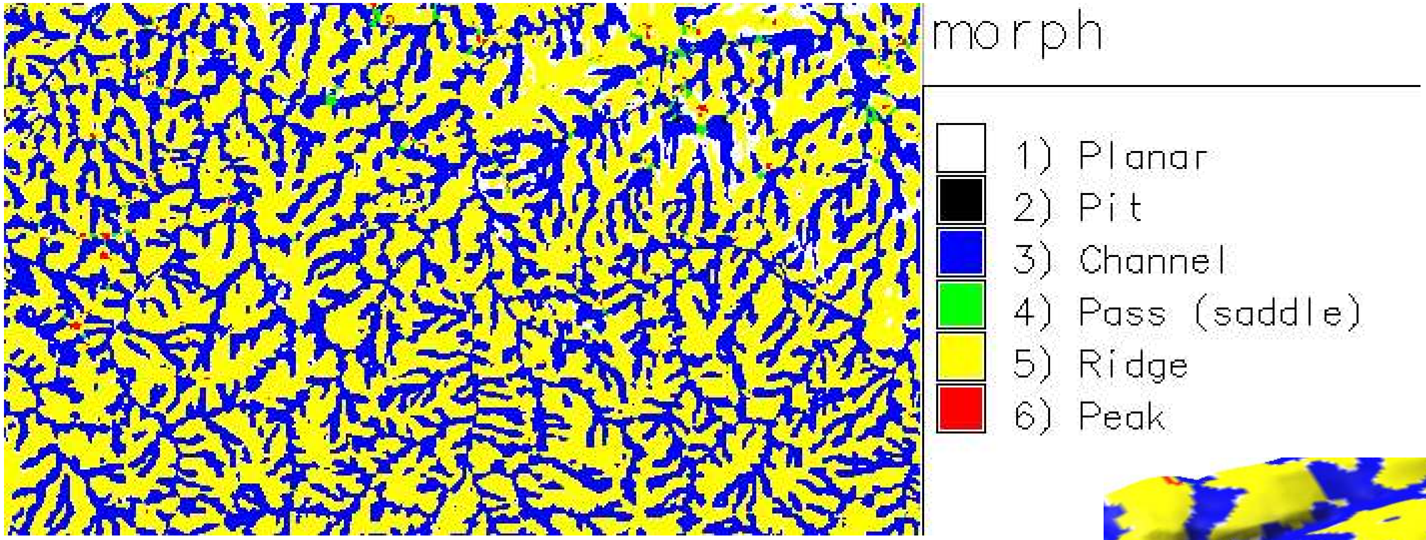
```
# First we reset the current GRASS region settings to the input map:
g.region rast=elevation.dem -p

r.slope.aspect el=elevation.dem as=aspect_30m sl=slope_30m

d.rast aspect_30m
d.rast.leg slope_30m
```



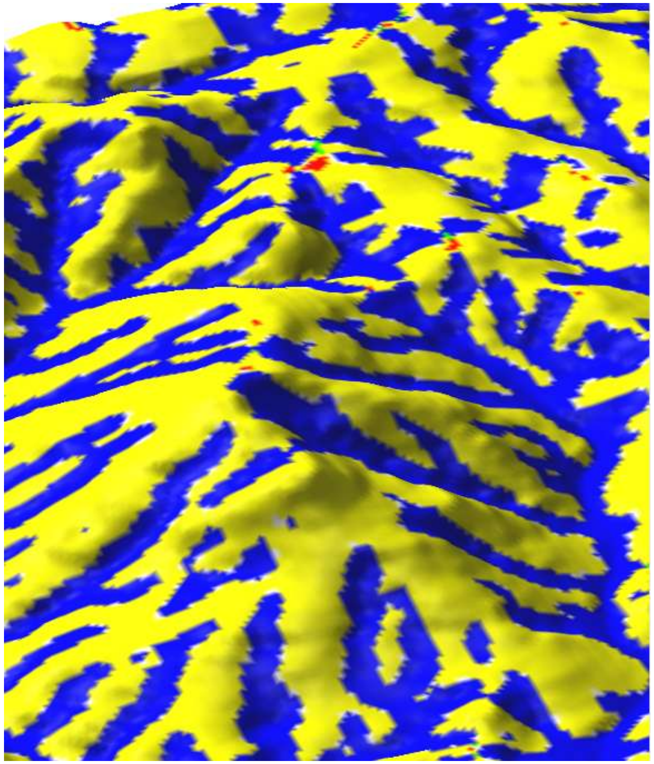
- Note: horizontal angles are counted counterclockwise from the East
- Slopes are calculated by default in degrees
- Also curvatures can be calculated

Raster data analysis: Geomorphology

DEM: r.param.scale



DEM: 30m
Size: 7x7



zoom to the input map:
```
g.region rast=elevation.dem -p
```

generalize with size parameter
```
r.param.scale elevation.dem out=morph \
                    param=feature size=21
```

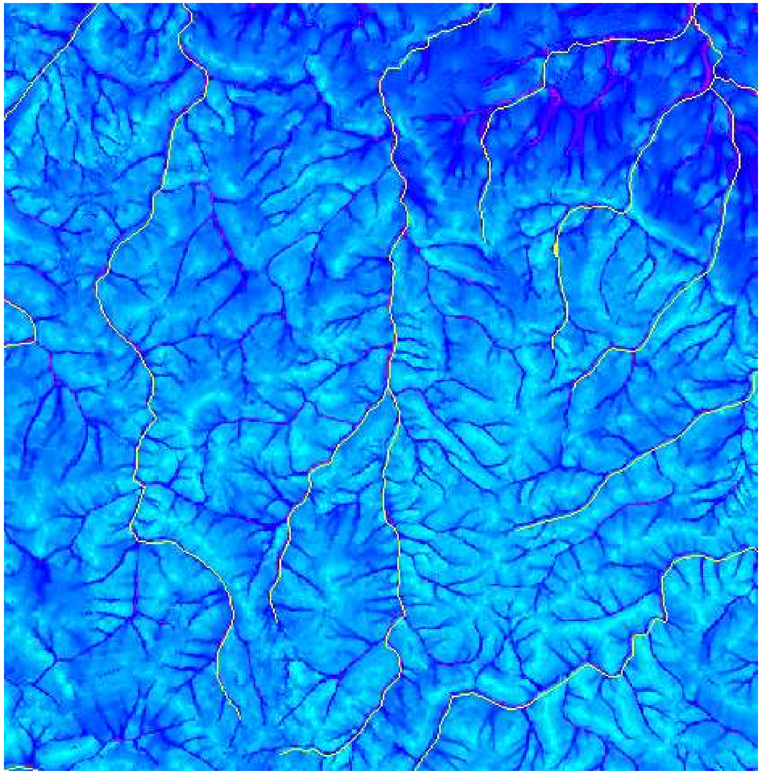
with legend
```
d.rast.leg morph
```

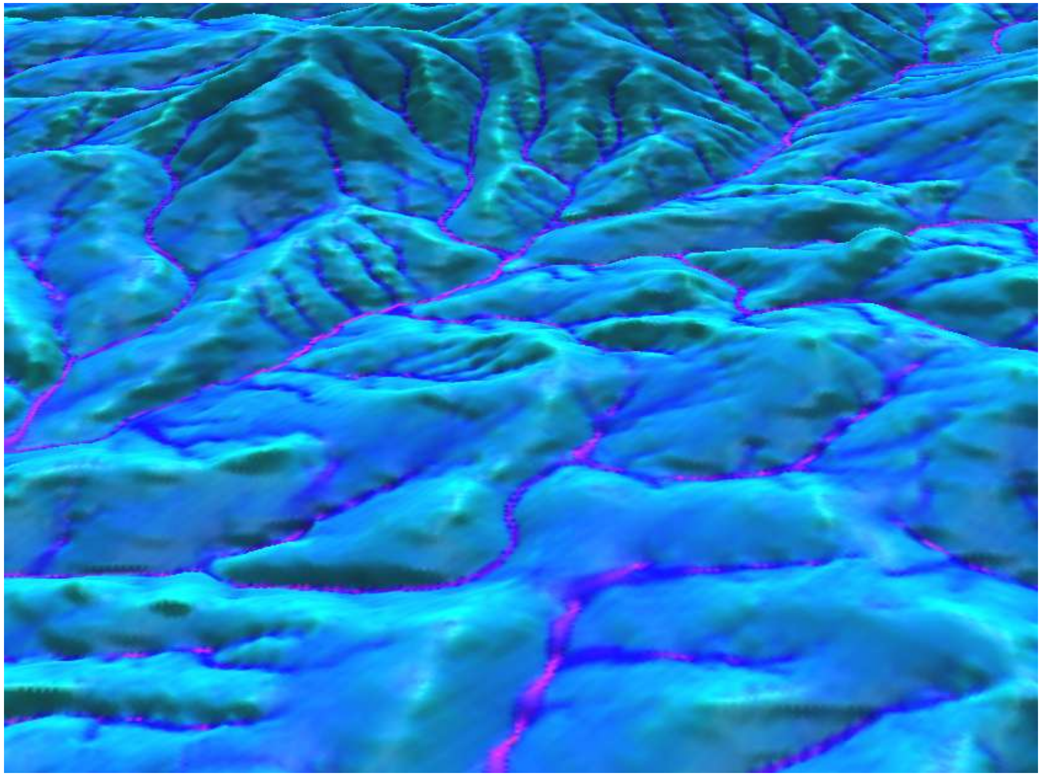
view with aspect/shade map (or QGIS)
```
d.his h=morph i=aspect.10m
```

Raster data analysis: Water flows - Contributing area

Topographic Index: ln(a/tan(beta))

```
 g.region rast=elevation.10m -p
# zoom to smaller area
 d.zoom

 r.topidx in=elevation.10m out=ln_a_tanB
 d.rast ln_a_tanB
 d.vect streams col=yellow
 nviz elevation.10m col=ln_a_tanB
```





Raster data analysis: further methods

- Additional DEM analysis modules:
  - depression areas can be filled with **r.fill.dir**
  - flowlines can be calculated with **r.flow**
  - trace a flow through a DEM: **r.drain**
  - watershed analysis can be done with **r.watershed** and **r.terraflow**
  - cost surfaces: **r.cost**

- Energy:
  - cast shadows, astronomical calculations of sun position: **r.sunmask**
  - energy budget: **r.sun**

- Line of sight:
  - viewsheds can be generated with: **r.los**

- Interpolation methods
  - 2D inverse distance weighted: **v.surf.idw**
  - 2D from contour lines: **r.surf.contour**
  - 2D bilinear: **r.bilinear**
  - 2D regularized splines with tension (with cross validation): **v.surf.rst**
  - 3D regularized splines with tension (with cross validation): **v.vol.rst**
  - 2D/3D kernel densities: **v.kernel**

  - via R-stats: kriging, predictive models etc

Raster map algebra

- A powerful raster map algebra calculator is **r.mapcalc**
  See for functionality:

  ```
  g.manual r.mapcalc &
  ```

- With a simple formula we filter all pixels with elevation higher than 1500m from the Spearfish DEM:

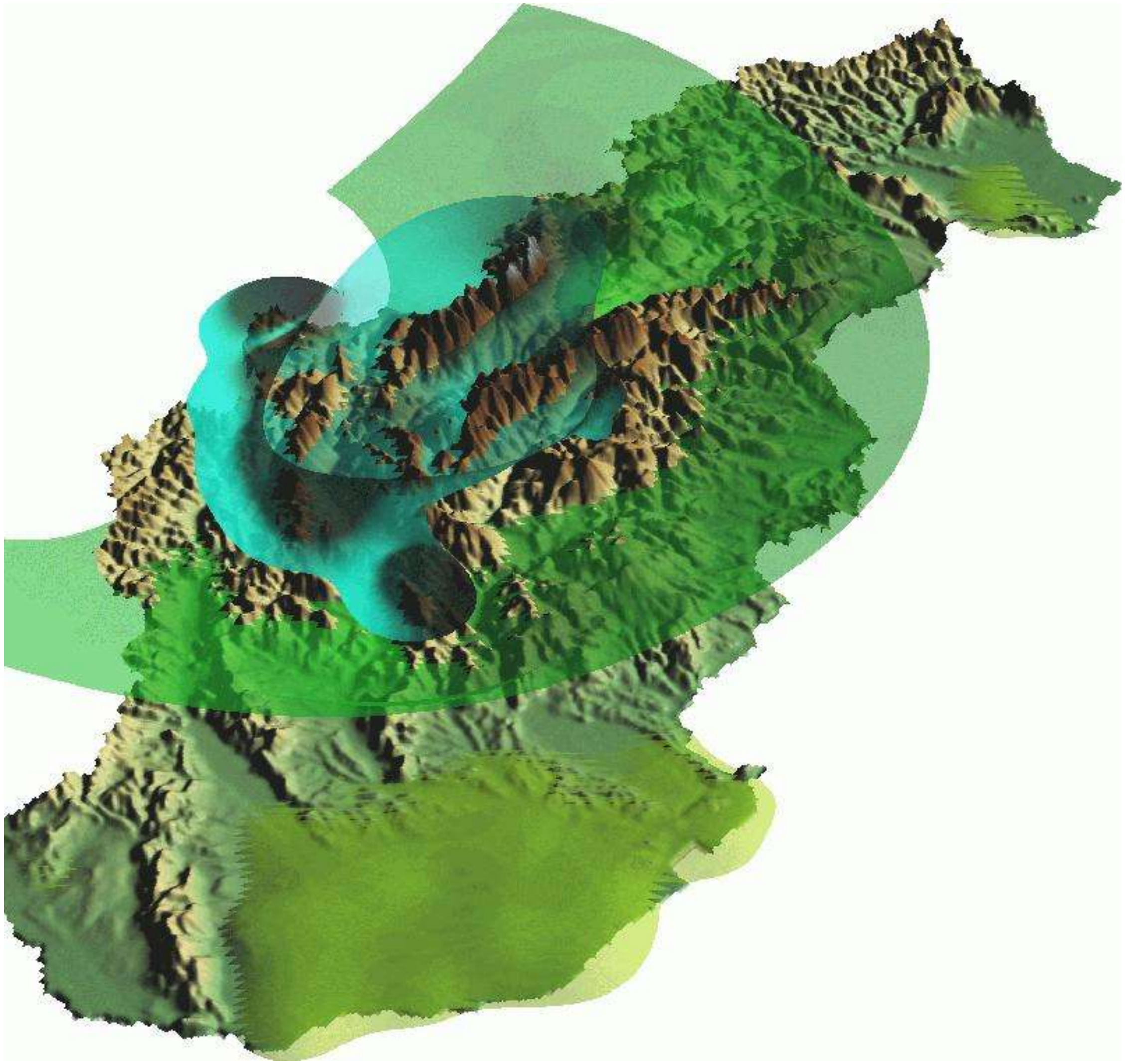
  ```
  r.mapcalc "elev_1500 = if(elevation.dem > 1500.0, elevation.dem, null())”
  d.rast elev_1500

  d.rast aspect
  d.rast -o elev_1500
  ```

Volume map processing: Demo

GRASS was enhanced to process and visualize Volumes (consisting of 3D voxels)

Functionality:

- 3D import/export
- 3D Regularized Splines with Tension interpolation
- 3D map algebra
- NVIZ volume visualization: Isosurfaces and Profiles



Outline of part II d

5 Image processing

- Image classification
- Image fusion with Brovey transform
- Calculating a degree Celsius map from the LANDSAT thermal channel

Import of LANDSAT-7 Erdas/Img

Image Classification

Unsupervised & Supervised Image Classification

- classification methods in GRASS:

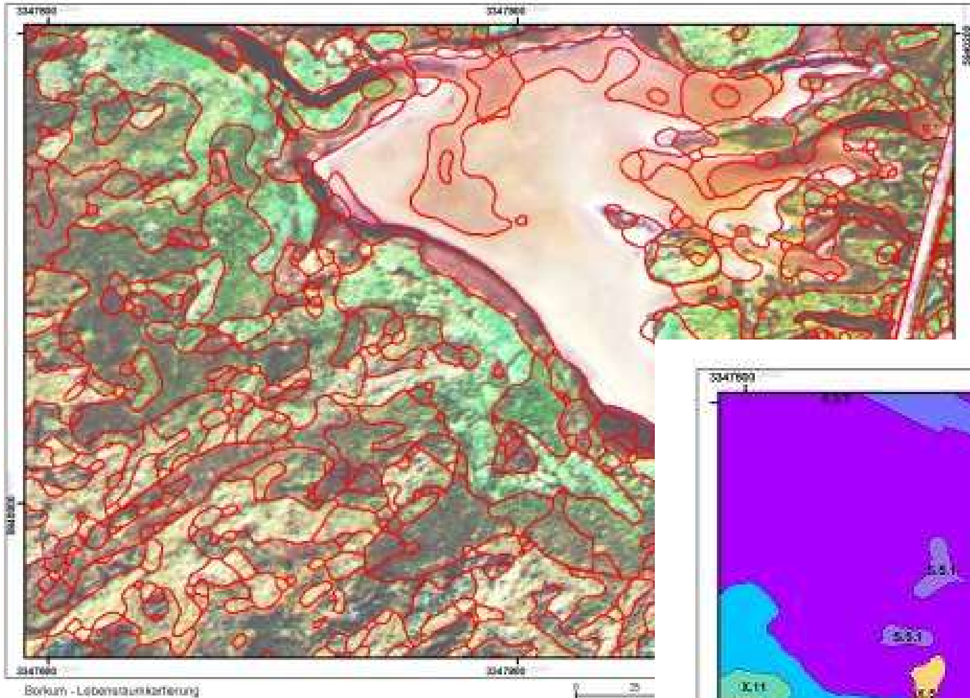
	radiometric, unsupervised	radiometric, supervised		radio- and geometric supervised
Preprocessing	i.cluster	i.class (monitor)	i.gensig (maps)	i.gensigset (maps)
Computation	i.maxlik	i.maxlik	i.maxlik	i.smap

- all image data must be first listed in a group (i.group)

- See handout for unsupervised classification example

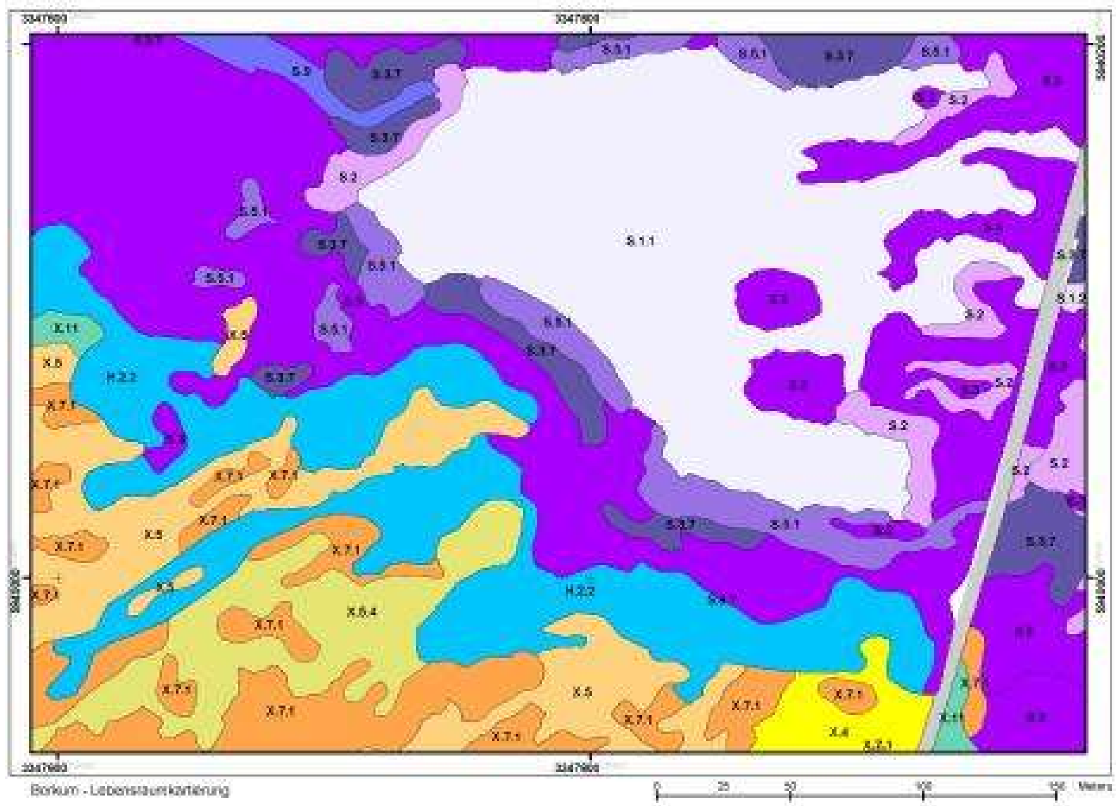
GRASS: Geographic Resources Analysis Support System

Image classification



GRASS supports

- Image geocoding and ortho-rectification
- Analysis of aerial and satellite data
- Time series analysis



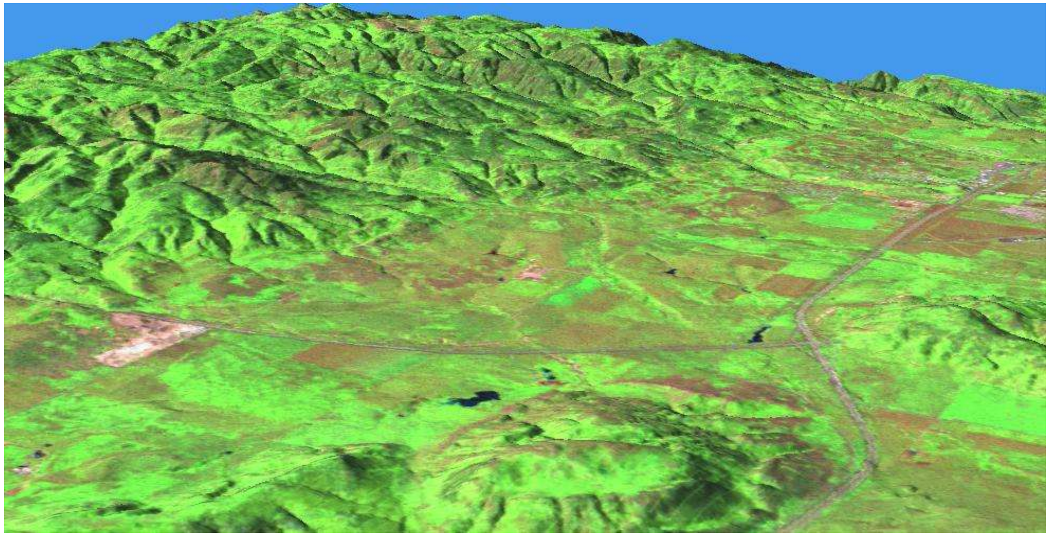
- Biotope monitoring from digital aerial cameras (HRSC-X and DMC)

- SMAP Classifier of GRASS

Image fusion: Brovey transform

The earlier imported LANDSAT-7 scene wil be used to perform image fusion of the channels 2 (red), 4 (NIR), and 5 (MIR):

```
g.region -dp
i.fusion.brovey -l ms1=tm.2 ms2=tm.4 ms3=tm.5 pan=pan out=brovey

# zoom to fused channel
g.region -p rast=brovey.red

# color composite:
r.composite r=brovey.red g=brovey.green b=brovey.blue n out=tm.brovey

d.rast tm.brovey

nviz elevation.10m col=tm.brovey

# Increase visual resolution in NVIZ
# with Panel -> Surface
#               -> Polygon resolution
#                   (lower! the value)
```



Recalibrating the LANDSAT-7 thermal channel 1/2

TM61: Conversion of temperature first to Kelvin, then to degree Celsius

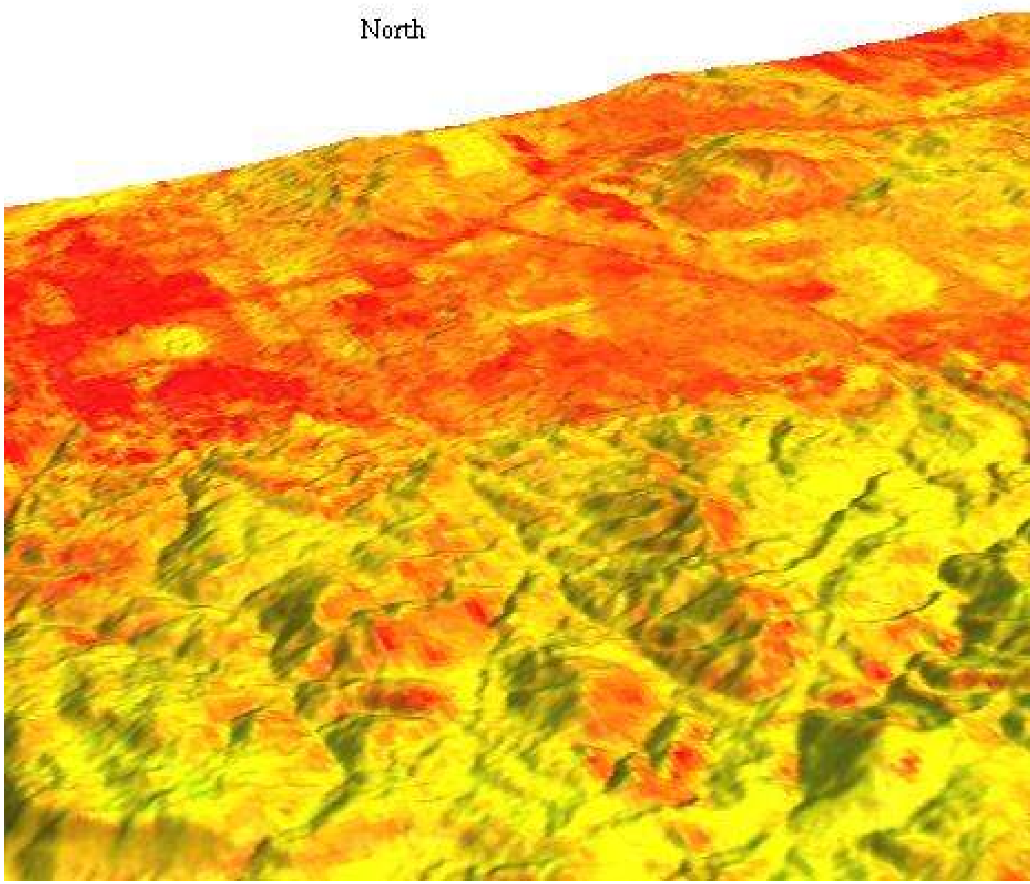
```
g.region rast=tm6.1 -p

#DN: digital numbers (coded temperatures)
 r.info -r tm6.1
   min=131
   max=175

# Conversion of DN to spectral radiances:
 r.mapcalc "tm61rad=((17.04 - 0.)/(255. - 1.))*(tm6.1 - 1.) + 0."
 r.info -r tm61rad
   min=8.721260
   max=11.673071

# Conversion of spectral radiances to absolute temperatures (Kelvin):
# T = K2/ln(K1/L_l + 1))
 r.mapcalc "temp_kelvin=1260.56/(log (607.76/tm61rad + 1.0))"
 r.info -r temp_kelvin
   min=296.026722
   max=317.399879
```

Recalibrating the LANDSAT-7 thermal channel 2/2

Note: Land surface temperatures are not air temperatures!
LANDSAT passes at around 9:30 local time

TM61: ... conversion to degree Celsius

```
# We currently have the land surface temperature map in Kelvin.
# Conversion to degree Celsius:
 r.mapcalc "temp_celsius=temp_kelvin - 273.15"
 r.info -r temp_celsius
   min=22.876722
   max=44.249879

# New color table:
r.colors temp_celsius col=rules << EOF
-10 blue
15 green
25 yellow
35 red
50 brown
EOF

d.rast.leg temp_celsius

g.region rast=elevation.dem -p
nviz elevation.dem col=temp_celsius
```



Outline of part II e

6 Working with vector data

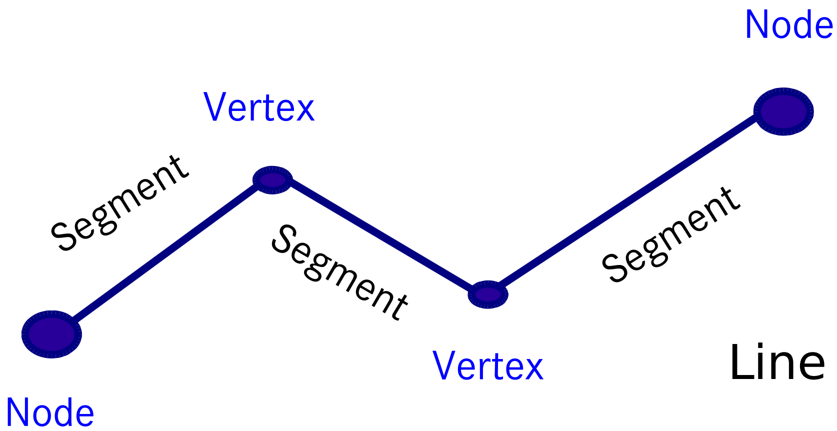
- Vector map import
- Attribute management
- Buffering
- Extractions, selections, clipping, unions, intersections
- Conversion raster-vector and vice verse
- Digitizing in GRASS and QGIS
- Working with vector geometry

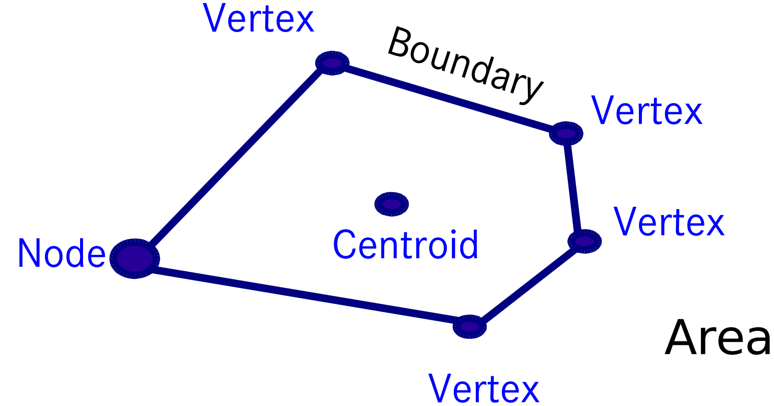
GRASS 6 Vector data

Vector geometry types

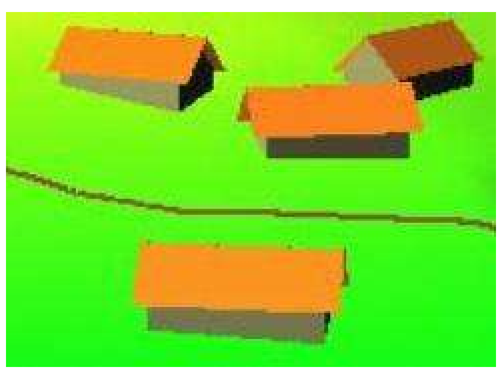
- Point
- Centroid
- Line
- Boundary
- Area (boundary + centroid)
- face (3D area)
- [kernel (3D centroid)]
- [volumes (faces + kernel)]

Geometry is **true** 3D: x, y, z





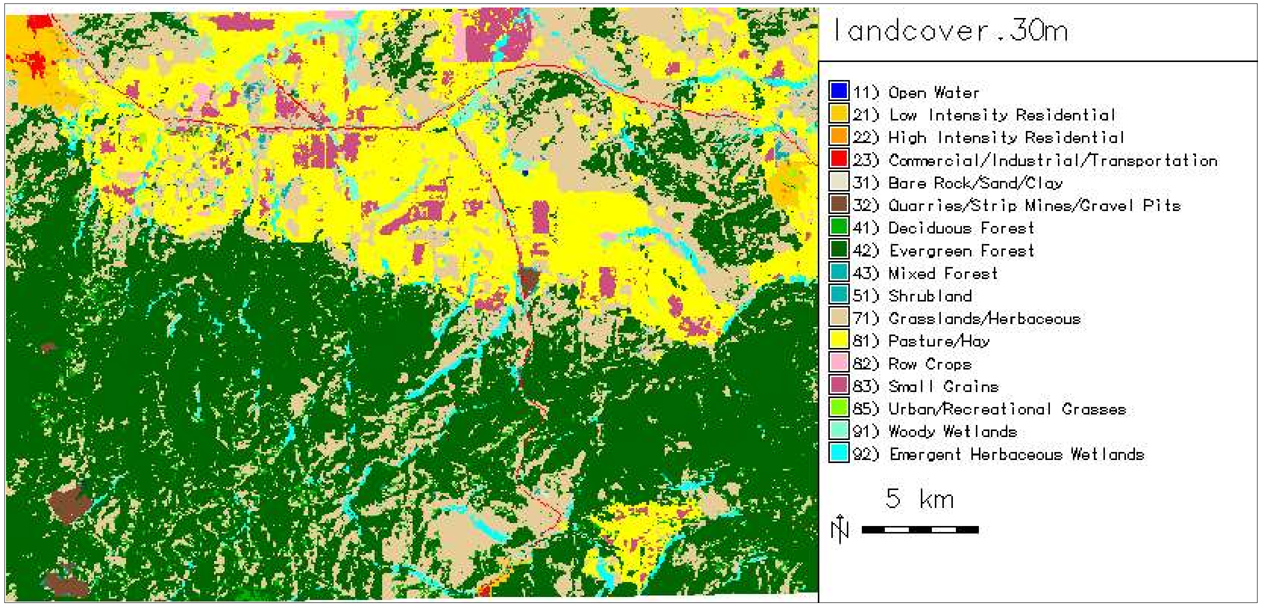
Faces



Raster-Vector conversion – extraction 1/2

Extraction of residential areas from raster landuse map

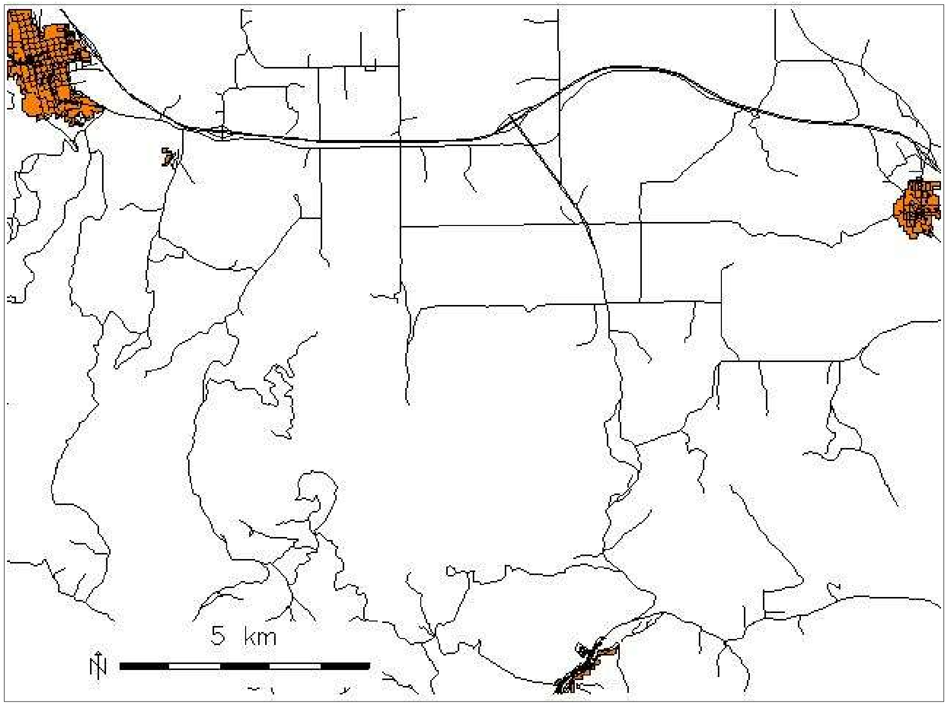
```
# look at the landuse/landcover map with legend:
 g.region rast=landcover.30m -p
 d.erase
 d.rast.leg -n landcover.30m

# Automated vectorization of the landuse/landcover map:
 r.to.vect -s landcover.30m out=landcover30m feature=area

# see attribute table:
 v.db.connect -p landcover30m
 v.db.select landcover30m
```



Raster-Vector conversion – extraction 2/2

Extraction of residential areas from raster landuse map

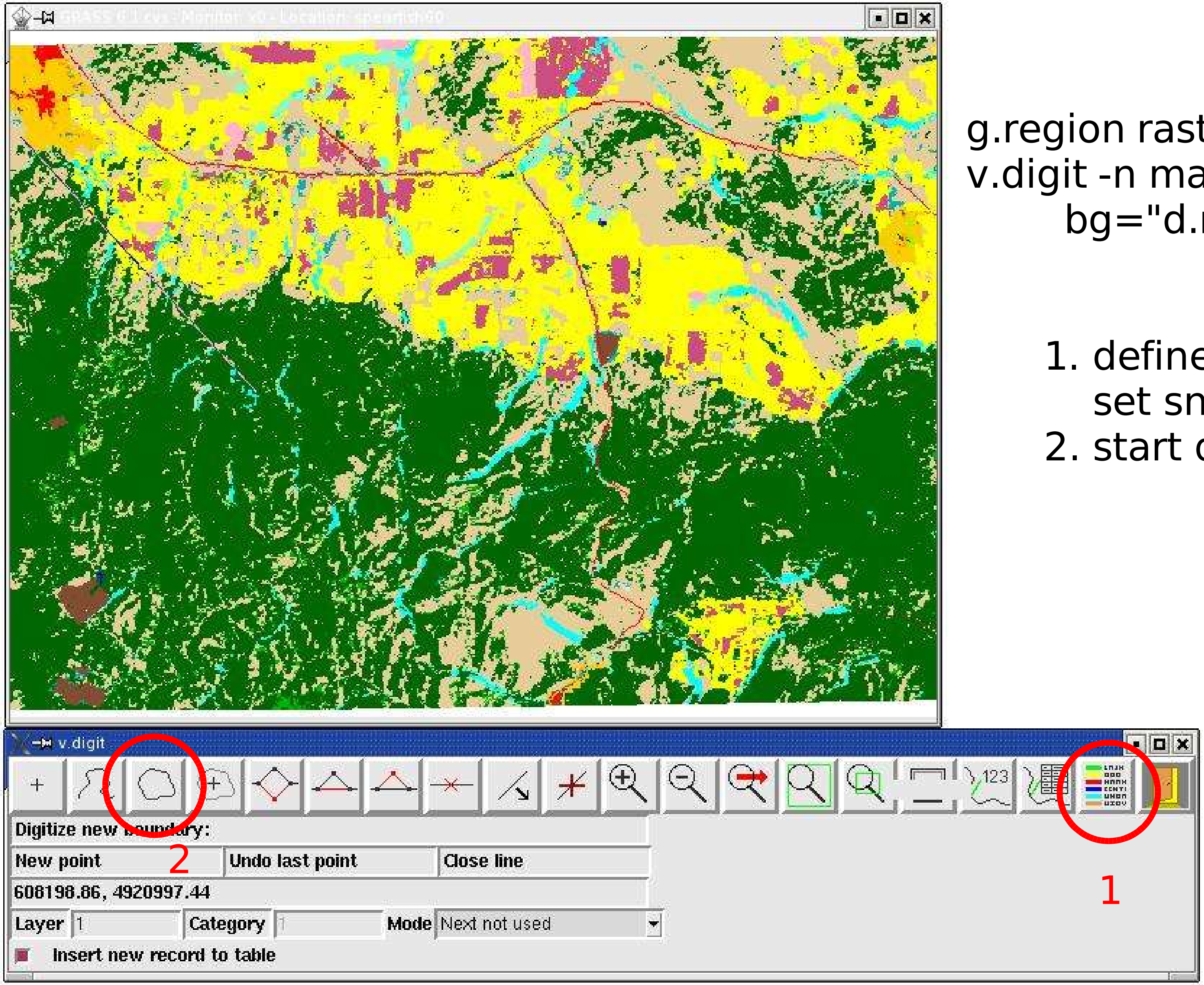
```
# generate list of unique landuse/landcover types:
 v.db.select landcover30m | sort -t '|' -k2 -un

#display selected categories:
 d.erase
 d.vect landcover30m where="value=21 or value=22" fcol=orange

# Extract residential area into a new vector map:
 v.extract landcover30m out=residential where="value=21 or value=22"

d.frame -e
d.vect residential fcol=orange \
           type=area
d.vect roads
d.barscale -mt
```



Markus Neteler 2005

Creating/modifying vector maps

Digitizing in GRASS




```
g.region rast=landcover.30m -p
v.digit -n map=cities \
      bg="d.rast landcover.30m"
```

1. define table
   set snapping threshold
2. start digitizing

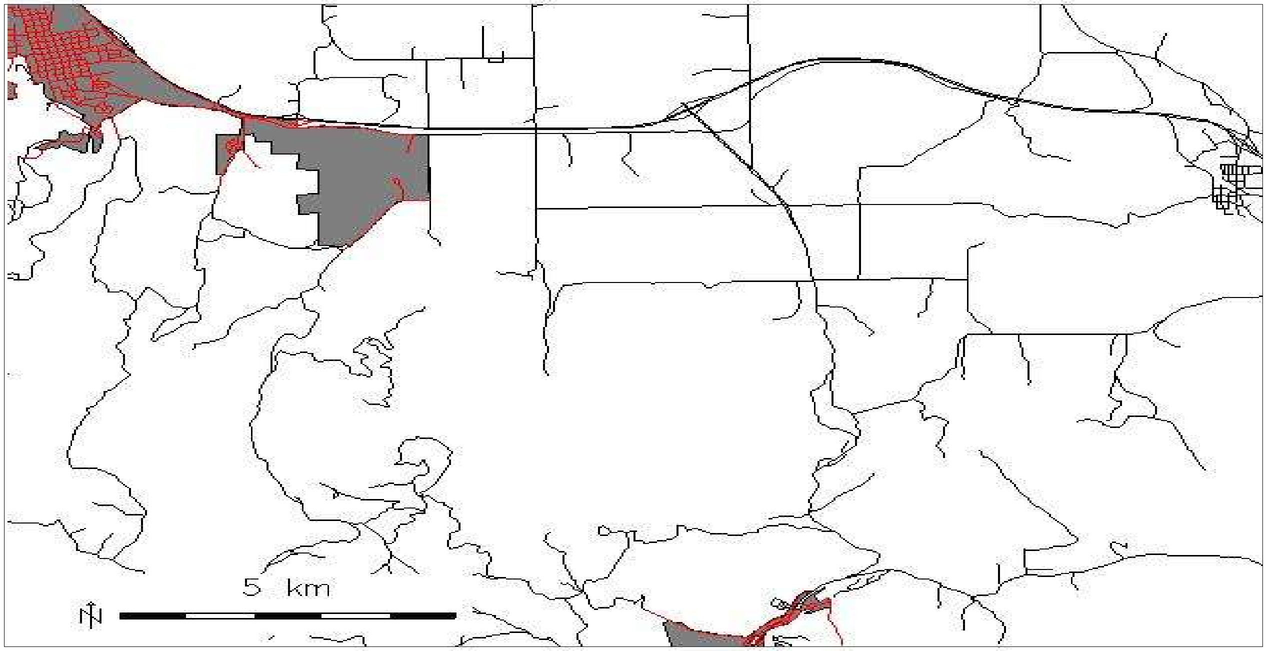
Alternative:
QGIS digitizer

Markus Neteler 2005

Vector map clipping

TIGER data: roads in urban areas

```
# import urban areas (from package tiger2000_UTM13_nad27.tar.gz):
 v.in.ogr dsn=UA_46081_UTM13_nad27.shp out=urban_areas
 d.vect urban_areas type=area

# display roads:
 d.vect roads

# extract all roads within the urban areas:
 v.select ain=roads bin=urban_areas out=urban_roads
 d.vect urban_roads col=red
```

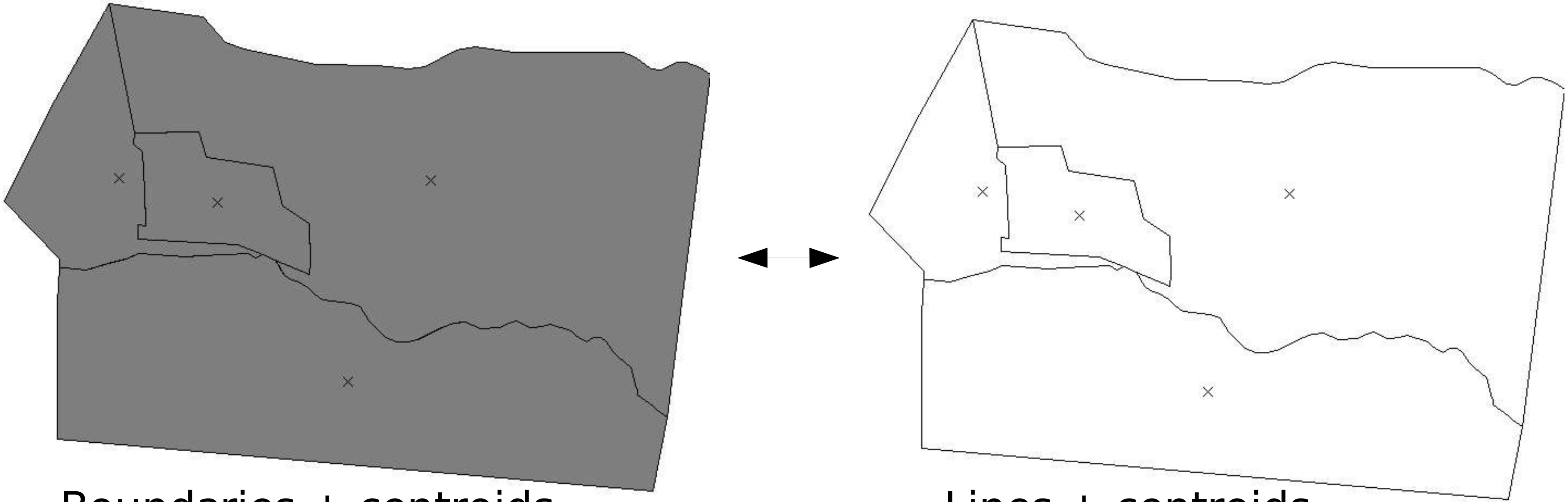


Changing vector types

In GRASS an **area** polygon is defined by a boundary + a centroid.
Lines can be a (poly)line or a boundary.

Vector types can be changed by **v.type/v.build.polyline** such as

point	◄──►	centroid
3D point	◄──►	kernel (3D centroid)
line	◄──►	polyline
line	◄──►	boundary
3D area	◄──►	face



Boundaries + centroids   Lines + centroids

Outline of part II f

7 Vector networking

- Overview
- Shortest path analysis

Vector network analysis methods

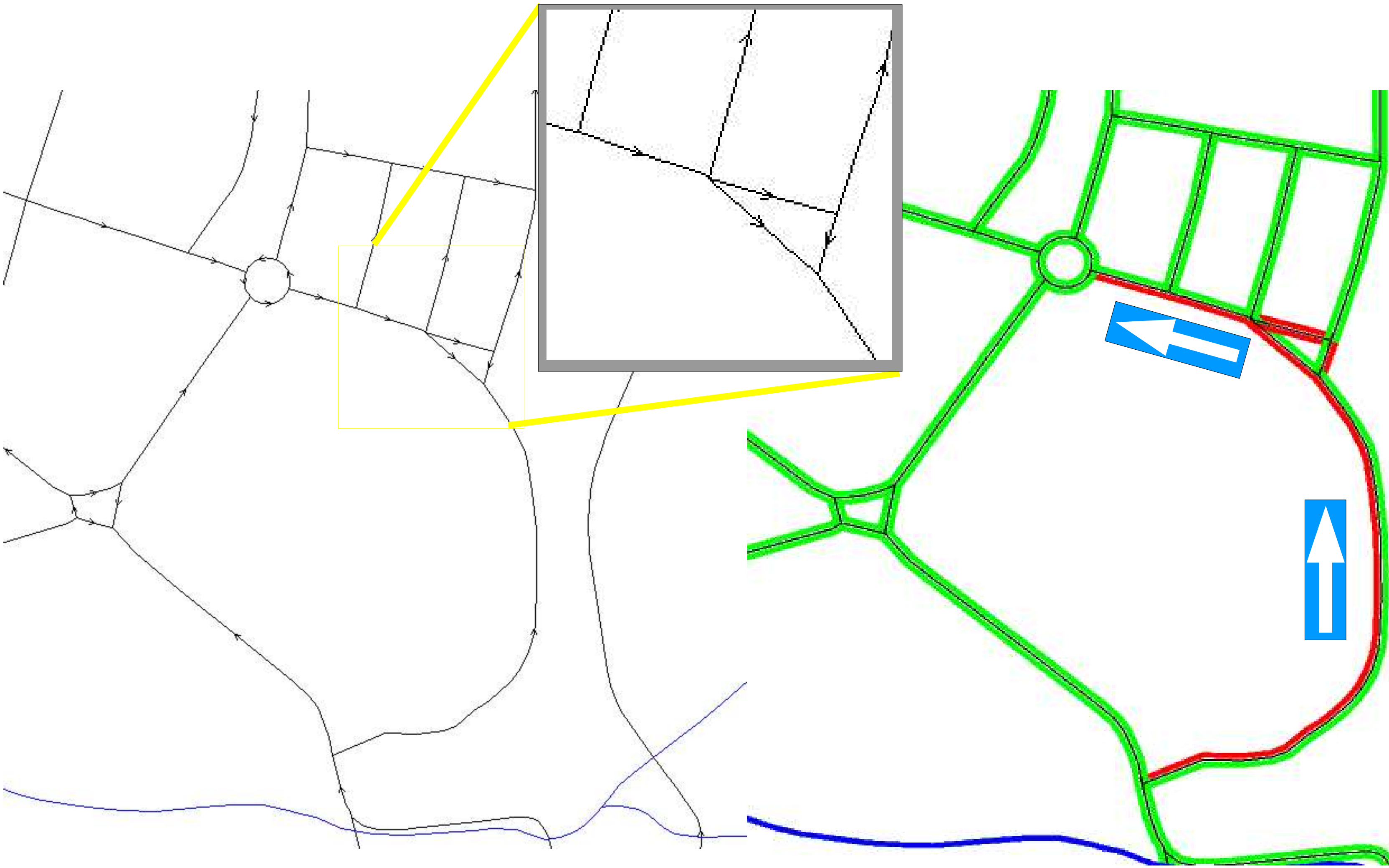
Available methods:

- find shortest path along vector network - *road navigation*
- find optimal round trip visiting selected nodes
  (Traveling salesman problem) - *delivering of goods*
- find optimal connection between nodes
  (Minimum Steiner tree) - *ADSL network*
- subdivide a network in subnetworks
  (iso distances) - *how far can I go from a node in all directions*
- find subnetworks for set of nodes
  (subnet allocation) - "catchment area" for fire brigade etc

Vector network analysis methods

Vector network with one way roads

drawn in ps.map



Street direction
open
closed

Generic vector directions

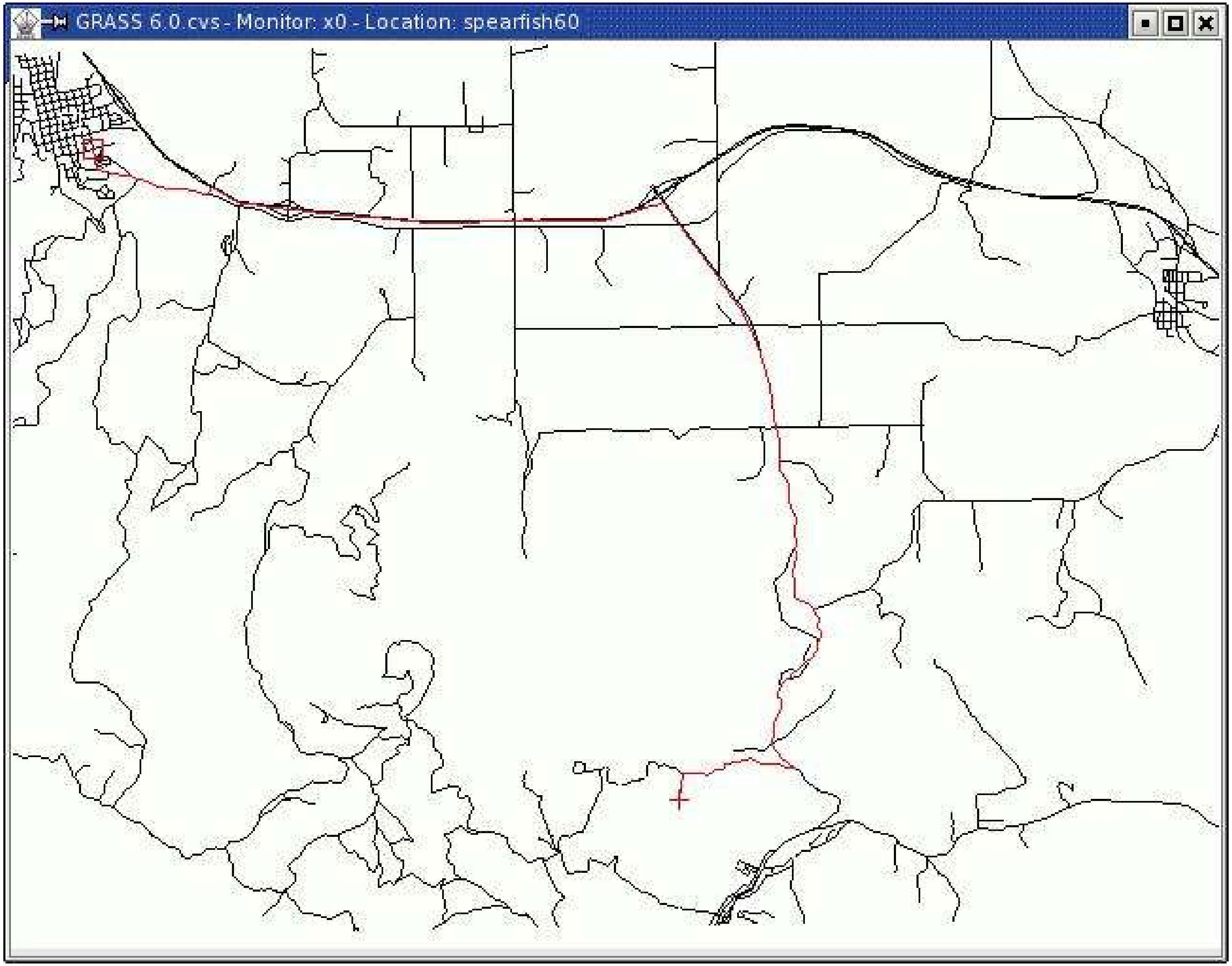
One attribute column for each direction
Value -1 closes direction (for one way streets)

Vector networking

Shortest path with **d.path**

d.vect roads
d.path roads

or:
v.net.path



Outline of part II g

8 GRASS and geostatistics

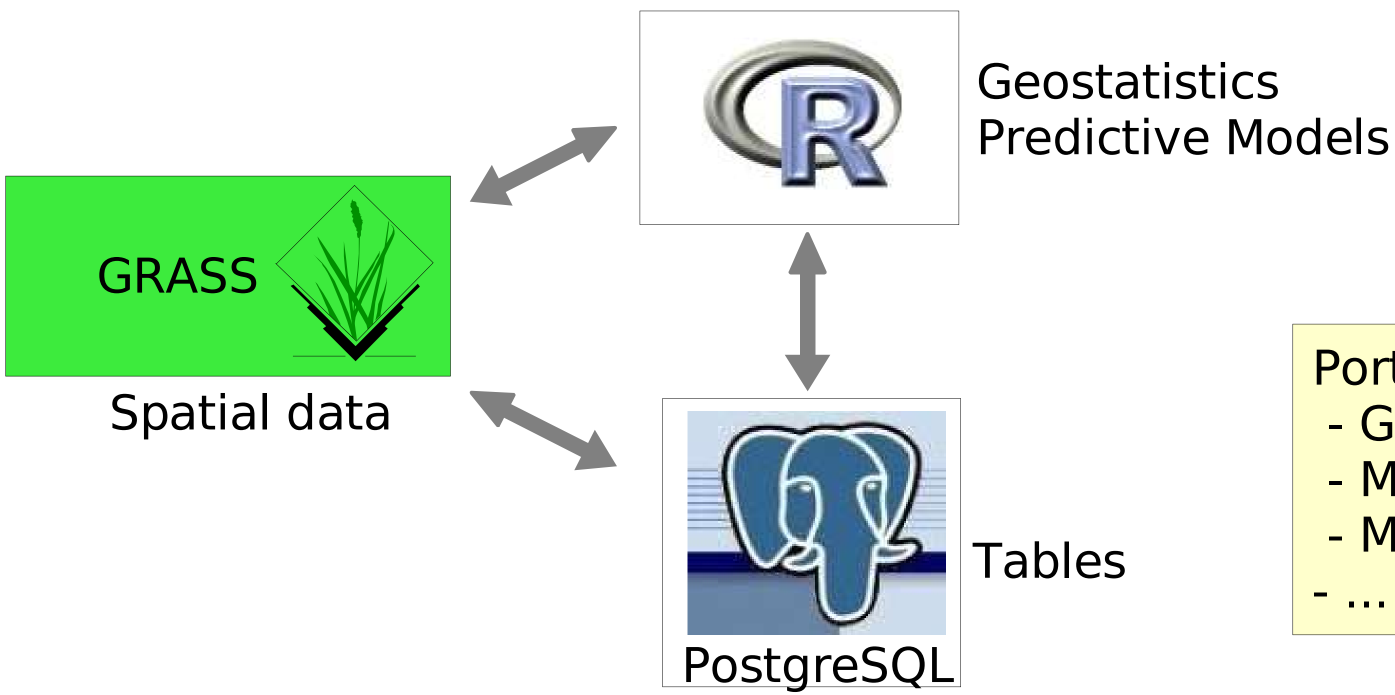
- R-stats/GRASS interface

Closure of the Workshop

GRASS/R-stats interface - R-stats/PostgreSQL interface

http://www.r-project.org
http://grass.itc.it/statsgrass/

- R-stats is a powerful statistical language
- Spatial extentions available for all kinds of **geostatistics, spatial pattern analysis, time series** etc
- Interface to exchange raster and point data between GRASS and R-stats
- Rdbi: connects R-stats to PostgreSQL



Portable Applications:
- GNU/Linux
- MS-Windows
- MacOSX
- ...

GRASS/R-stats interface

R statistical language

Web site and CRAN:

http://www.r-project.org

http://cran.r-project.org

Object oriented statistical language, a “S” dialect. Examples:

```
 R
> 1
[1]

> 1+1
[1] 2

# assignment into object:
> x <- 1+1
> x
[1] 2

> q()
Save workspace image [y/n/c] n
```

GRASS/R-stats interface

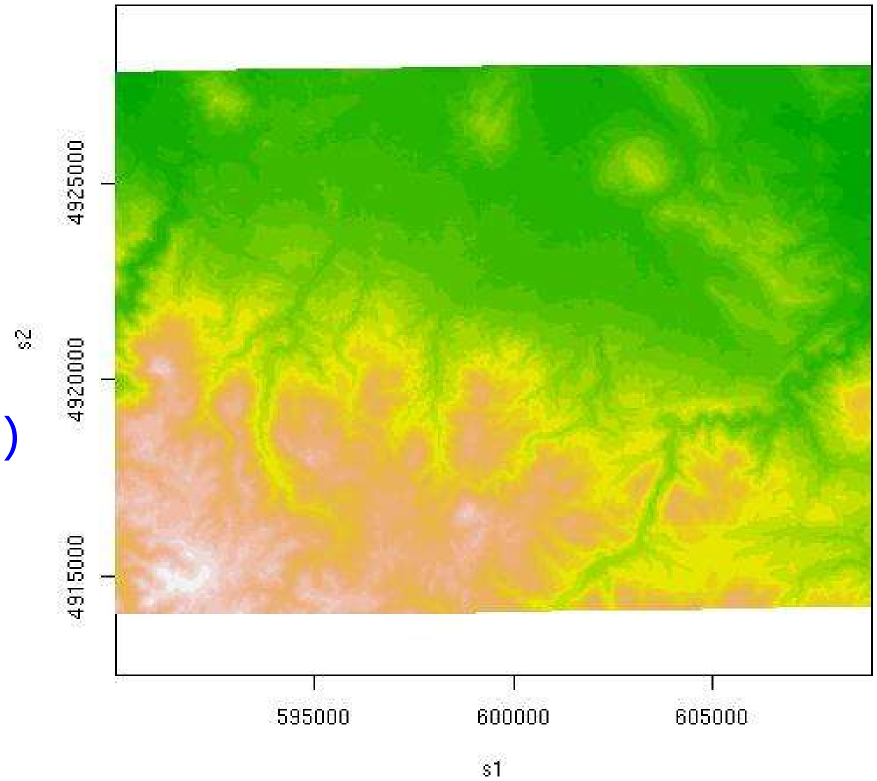
GRASS 6 and R statistical language

```
grass60

#reset region:
g.region rast=elevation.dem -p

#in GRASS start R:
R
library(spgrass6)

#load GRASS environment into R:
G <- gmeta6()

#see environment metadata:
str(G)


# Now R is ready for GRASS data analysis.
```

GRASS/R-stats interface

GRASS 6 and R statistical language (cont'd)

```
# get online help:
?readCELL6sp

# get full help:
help.start()

# load GRASS DEM into R:
elev <- readCELL6sp("elevation.dem")

# show map metadata:
summary(elev)

# show map:
image(elev, col=terrain.colors(20))

# leave R:
q()

# you have the option to save the current R state to local
# file when leaving the program.
```



Closure of the Workshop

Thanks for your interest and your attention!

M. Neteler
neteler at itc it
http://mpa.itc.it
http://www.gdf-hannover.de

ITC-irst, Povo (Trento), Italy
GDF Hannover, Germany

K. Perry
kperry at spatialfocus com
http://www.spatialfocus.com

USA