New GUI for GRASS GIS based on wxPython

Martin Landa

Department of Geodesy and Cartography
Faculty of Civil Engineering, Czech Technical University in Prague
and
FBK-irst, Trento, Italy
martin.landa@fsv.cvut.cz

Abstract. This article discusses GUI development for GRASS GIS. Sophisticated native GUI is one of the key points (besides the new 2D/3D raster library, vector architecture improvements, etc.) for the future development of GRASS.
The current GUI is written in Tcl programming language using Tk graphical toolkit. The limitations of Tcl/Tk toolkit appeared to be fundamental for the future development. This issue has been several times discussed in the GRASS developer mailing list. At the end has been decided to leave Tcl/Tk and to design new native GUI from the scratch using another graphical toolkit. In particular, wxPython – a blending of the wxWidgets library with the Python programming language. The project started in the beginning of 2007. This article attempts to summarize the results of the development.
The new wxPython-based GUI will be available in GRASS 6.4. Native GUI is crucial for GRASS user politics especially connected to the newcomers or the users who essentially request GUI. The GUI need to reflect the specific needs of GRASS users, must be intuitive, simple and in the certain point ”minimal”.

Keywords: GIS, GRASS, GUI, development, wxPython

Abstrakt. Tento článek popisuje vývoj GUI pro GRASS GIS. Tvorba sofistikovaného nativního GUI je jedním ze zásadních bodů (kromě nové 2D/3D rastrové knihovny, vylepšení vektorové architektury a pod.) v aktuálním vývoji GRASSu.
Dosavadní GUI je napsáno v programovacím jazyce Tcl s využitím grafického toolkitu Tk. Omezení Tcl/Tk se ukázala jako zásadní pro další vývoj. Tato tématika byla několikrát diskutována ve vývojářském mailing listu GRASSu. Výsledkem bylo rozhodnutí opustit Tcl/Tk a navrhnout od základů nové nativní GUI s využitím jiného grafického toolkitu. Ve výsledku byl pro budoucí vývoj zvolen wxPython (knihovna wxWidgets pro programovací jazyk Python). Na projektu se pracuje od počátku roku 2007. Tento článek si klade za cíl shrnout dosavadní výsledky vývoje.
GUI založené na wxPython bude dostupné v GRASS 6.4. Nativní GUI je důležité pro uživatelskou politiku GRASSu zejména v souvislosti se začátečníky nebo uživateli, kteří v zasadě pro svoji práci GUI vyžadují. GUI musí reflektovat specifické potřeby uživatelů GRASSu, musí být intuitivní, jednoduché a v určitém ohledu ”minimalistické”.

Klíčová slova: GIS, GRASS, GUI, vývoj, wxPython

1 The GRASS History

GRASS (*Geographic Resource Analysis Support System*) GIS [1] as a raster/vector GIS (Geographical Information System) contains over 350 programs (so-called modules) and tools that can create, manipulate, and store spatial data. It is widely recognized in the open source GIS community that GRASS is the largest, most powerful and reliable Free Software GIS project.

One of the strengths of GRASS is its modular nature, which allows users to add their own commands to the base system. GRASS provides the opportunity for a person with basic C language programming skills to write and link his/her own modules to the package's internal "front end". The ability to study the code organization of existing modules is very helpful and facilitates the creation of new modules.

1.1 Historical notes

Beginning in the early 1960s, GIS have evolved from mainframe computer programs written in FORTRAN into highly complex desktop PC software that we have today. In the early 1980s, GIS development went into *two directions*. Commercial GIS companies such as ESRI (Environmental Systems Research Institute) [7] began producing commercial GIS packages such as Arc/Info while the United States Army Construction Engineering Research Laboratory (USA-CERL) began developing a no-fee GIS package called Geographic Resource Analysis Support System (GRASS). Costs involved in implementing GIS (acquiring data and hardware, learning GIS skills and computer maintenance skills) became so high, USA-CERL decided that they could reduce cost of extremely expensive systems by developing their own software in a Unix environment. This made GRASS the first GIS package to be available on a PC. It wasn't until 1986 that some of the larger commercial GIS packages such as Arc/Info had made the transition from large computers to the PC (PC Arc/Info was released on a Unix platform). By the mid-1990s many of the original government GRASS GIS users were switching to proprietary software such ArcView. ESRI's ArcView would ultimately evolve into present day ArcGIS 9.0, while the USA-CERL version of GRASS would eventually become today's open source and freely available GRASS 6.2 [21].

GRASS has been under continuous development since 1982 and has involved a large number of federal US agencies, universities, and private companies. The core components of GRASS and the management of the integration efforts into GRASS releases were accomplished by the USA-CERL in Champaign, Illinois. USA-CERL completed its last release of GRASS as version 4.1 in 1992 and also wrote the core components of the GRASS 5.0 floating point version [18].

GRASS as the oldest Free GIS software is still active. It has played an important role in the progress made in the geospatial model, both in education and in the scientific community. Moreover, it has played an important role in the field of business, for creating solutions to solve spatial problems. USA-CERL discontinued development of GRASS in 1995 and turned over development to the first GRASS research group at Baylor University in 1997. Currently, GRASS is revised and updated by a worldwide GRASS development team. The first GPL'ed GRASS GIS has been released in 1999 as version 5.0.

In the result, GRASS GIS is 25 years of age and still attractive for users and developers [16].

1.2 The GUI for GRASS GIS

While the early decision to first develop GRASS as a command-line system remained good, the need to provide users with a GUI (Graphical User Interface) was increasingly difficult to ignore [18]. A standard way to process command-line arguments was developed and added to every program. This also provided automatic user prompting and help options to users of nearly all commands. In the early 1990s software efforts were underway to develop an X-windows user interface and to add floating point support to GRASS. Although GRASS design and development continued, GRASS releases became less frequent which helped accelerate the movement of the GRASS user community towards commercial software.

GRASS as a piece of software with a long tradition is closely linked to Unix environment. The GRASS modules are originally CLI-oriented (Command Line Interface). This can be advantage for developers or power users. The real capabilities of GRASS are basically connected to the power of CLI. On the other hand, absence of GUI makes learning GRASS extremely hard for newcomers – time consuming task. The position of GRASS on the market does not correspond to the capabilities which the software can offer to the end-user. In GRASS world, GUI can be seen as an alternative way how to use the system, the main advantage for the users is powerful CLI. Anyway GUI is very important point for the most of GRASS users, requested especially by newcomers. Besides that, there are tools like Georectifier or Digitizer which are basically mouse-driven.

In this section will be shortly described history of GUI for GRASS GIS. All native GUIs were written in Tcl scripting language using Tk graphical toolkit.

The first native GUI for GRASS was originally developed by Jacques Bouchard in 1999 and became part of GRASS 5.x, so-called *TCLTKGRASS* (fig. 1). The core component was the menu which allowed users to run GRASS modules via simple graphical dialogs.

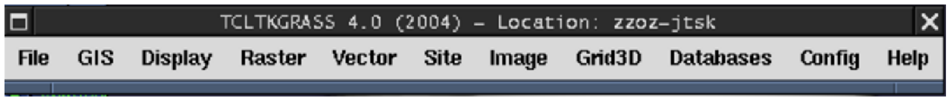
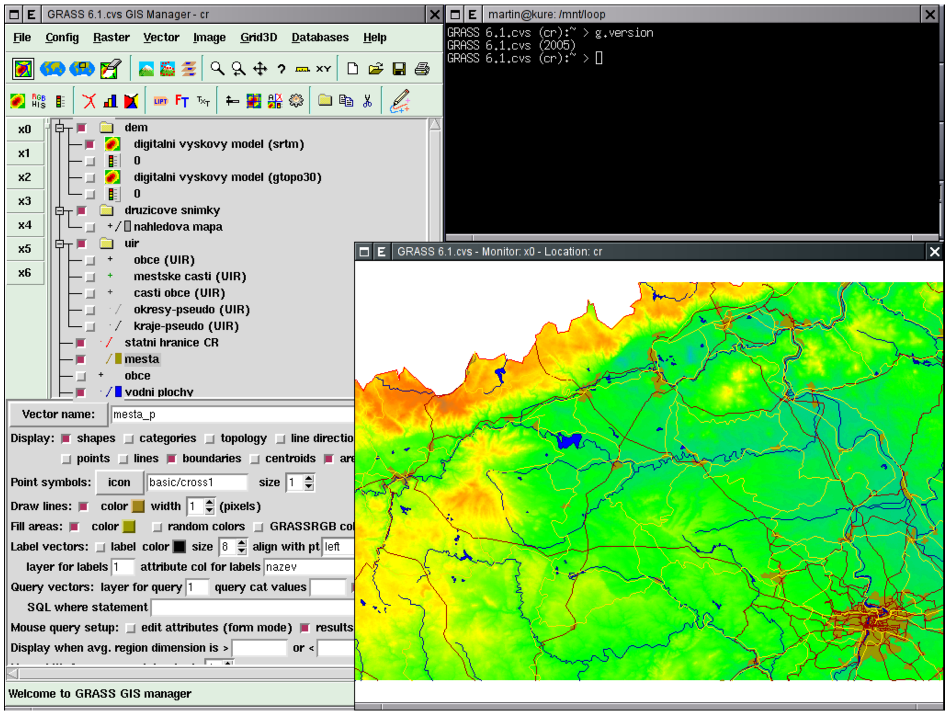


Fig. 1. First native GUI for GRASS GIS – TCLTKGRASS available in GRASS 5.0.

The replacement of TCLTKGRASS has been developed by Michael Barton, Radim Blažek and others. The *Display Manager* (`d.m` command) allowed users to run GRASS modules from menu and moreover graphically manage the GRASS monitors (original X11 driver-based graphics displays), see fig. 2.

The next step in GUI evolution was *GIS Manager* (`gis.m` command) developed mainly by Michael Barton in 2006 (fig. 3). Display architecture, module menu, graphical module dialogs (which are generated on-the-fly by the GRASS parser) have been completely rewritten, new output window and map layer management introduced. Moreover new external tools have been integrated, e.g. Georectifier (replacement of X11 driver-based `i.points` module). Also map display window with integrated basic tools like zooming, panning, data querying has been implemented (as a replacement of X11 driver-based GRASS monitors).

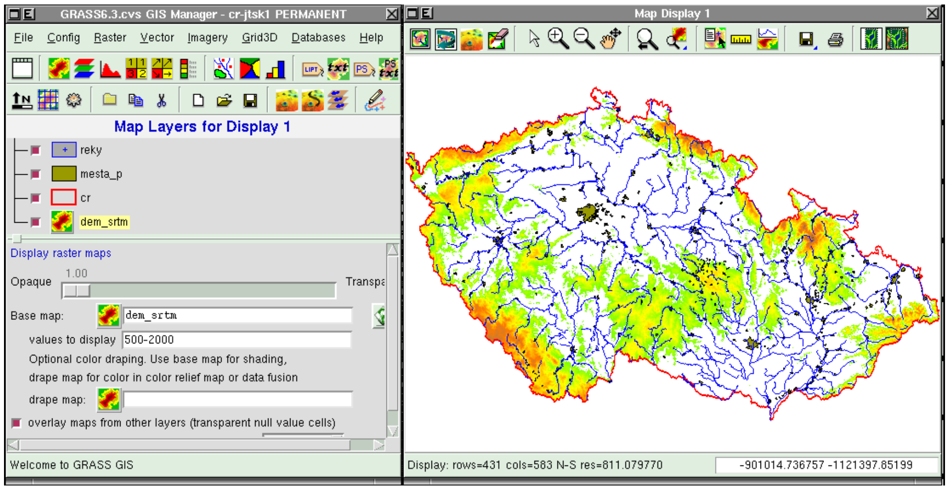


```
GRASS 6.1.cvs (cr):~ > g.version
GRASS 6.1.cvs (2005)
GRASS 6.1.cvs (cr):~ >
```

Vector name: mesta_p
Point symbols: icon basic/cross1 size 1
Draw lines: width 1 (pixels)
Label vectors: size 8 align with pt left
layer for labels 1 attribute col for labels nazev
Query vectors: layer for query 1

GRASS 6.1.cvs GIS Manager - cr
File Config Raster Vector Image Grid3D Databases Help
x0 x1 x2 x3 x4 x5 x6
dem
digitalni vyskovy model (srtm)
0
digitalni vyskovy model (gtopo30)
0
druzicove snimky
nahledova mapa
uir
obce (UIR)
mestske casti (UIR)
casti obce (UIR)
okresy-pseudo (UIR)
kraje-pseudo (UIR)
statni hranice CR
mesta
obce
vodni plochy
Display: shapes categories topology line directio
points lines boundaries centroids ar
Fill areas: color random colors GRASSRGB co
SQL where statement
Mouse query setup: edit attributes (form mode) results
Display when avg. region dimension is > or <
Welcome to GRASS GIS manager
martin@kure: /mnt/loop
GRASS 6.1.cvs - Monitor: x0 - Location: cr

Fig. 2. TCLTKGRASS successor – Display Manager in GRASS 6.1.



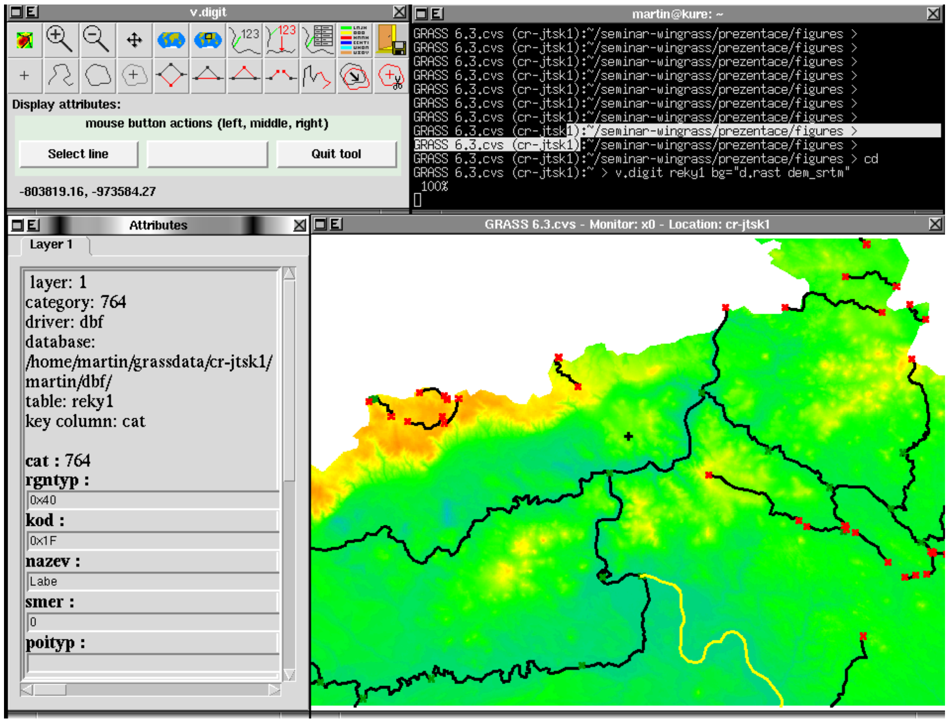
Base map: dem_srtm
values to display 500-2000
Display: rows=431 cols=583 N-S res=811.079770
-901014.736757 -1121397.85199

GRASS6.3.cvs GIS Manager - cr-jtsk1 PERMANENT
File Config Raster Vector Imagery Grid3D Databases Help
Map Layers for Display 1
reky
mesta_p
cr
dem_srtm
Display raster maps
1.00
Opaque Transpa
Optional color draping. Use base map for shading, drape map for color in color relief map or data fusion
drape map:
overlay maps from other layers (transparent null value cells)
Welcome to GRASS GIS
Map Display 1

Fig. 3. GIS Manager in GRASS 6.3.

Digitization tool (`v.digit` module) is originally CLI-based (GRASS 5). GRASS 6 brought completely rewritten GUI-based module (fig. 4). This module lacks some useful features of old-fashioned original CLI module, e.g. bulk-labeling (automated labeling of contours). Because of X11 driver dependency of GUI-driven `v.digit`, the module has been never integrated into GIS Manager.

The last major part represents NVIZ (fig. 5), a tool for visualization in 2.5/3D supporting 3D raster (voxel) and vector data.



Display attributes:
mouse button actions (left, middle, right)
Select line | | Quit tool

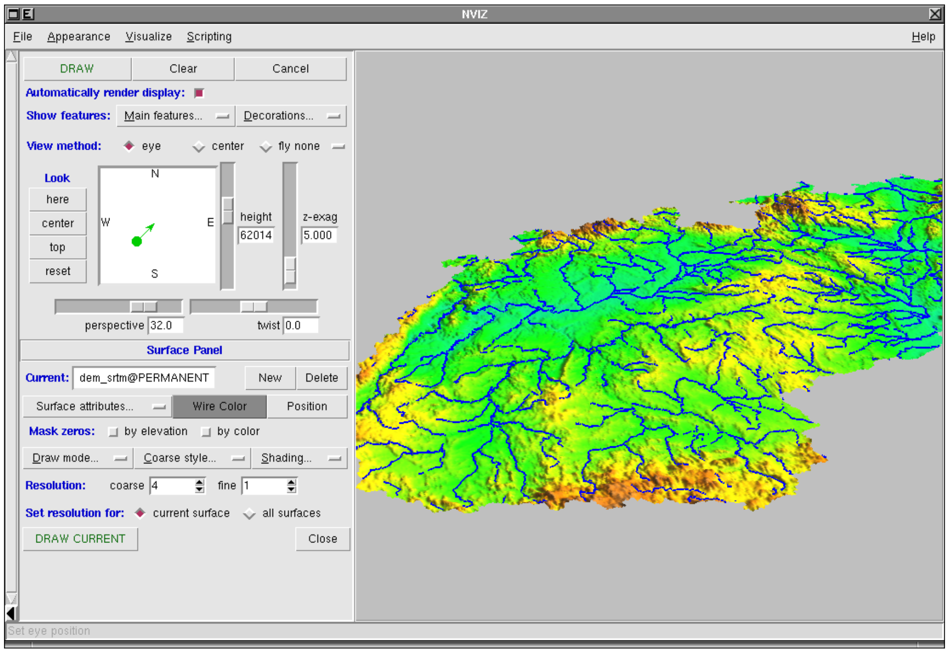
-803819.16, -973584.27

```
GRASS 6.3.cvs (cr-jtsk1):~/seminar-wingrass/prezentace/figures >
GRASS 6.3.cvs (cr-jtsk1):~/seminar-wingrass/prezentace/figures >
GRASS 6.3.cvs (cr-jtsk1):~/seminar-wingrass/prezentace/figures >
GRASS 6.3.cvs (cr-jtsk1):~/seminar-wingrass/prezentace/figures >
GRASS 6.3.cvs (cr-jtsk1):~/seminar-wingrass/prezentace/figures >
GRASS 6.3.cvs (cr-jtsk1):~/seminar-wingrass/prezentace/figures >
GRASS 6.3.cvs (cr-jtsk1):~/seminar-wingrass/prezentace/figures >
GRASS 6.3.cvs (cr-jtsk1):~/seminar-wingrass/prezentace/figures >
GRASS 6.3.cvs (cr-jtsk1):~/seminar-wingrass/prezentace/figures >
GRASS 6.3.cvs (cr-jtsk1):~/seminar-wingrass/prezentace/figures > cd
GRASS 6.3.cvs (cr-jtsk1):~ > v.digit reky1 bg="d.rast dem_srtm"
 100%
```

layer: 1
category: 764
driver: dbf
database:
/home/martin/grassdata/cr-jtsk1/
martin/dbf/
table: reky1
key column: cat

cat : 764
rgntyp : 0x40
kod : 0x1F
nazev : Labe
smer : 0
poityp :

Fig. 4. Digitization tool in GRASS 6.3 – module `v.digit`.



Automatically render display: ■
Show features: Main features... Decorations...
View method: ◆ eye ◇ center ◇ fly none
height 62014
z-exag 5.000
perspective 32.0
twist 0.0
Current: dem_srtm@PERMANENT
Surface attributes...
Mask zeros: ☐ by elevation ☐ by color
Draw mode... Coarse style... Shading...
Resolution: coarse 4 fine 1
Set resolution for: ◆ current surface ◇ all surfaces
Set eye position

Fig. 5. NVIZ, a module for visualization in 3D.

2 Motivation for GUI development

As with many ongoing projects, the number of options, flags, and parameters has grown quickly, and it has become inconvenient to remain in the command line environment (which was the only GRASS option 10 years ago). The development of an interactive front end was therefore a very useful option. The current GUI for GRASS is based on Tcl/Tk scripting language libraries [22].

GRASS shows high capability to analyze and manipulate data, and covers many application areas in GIS projects. However, it shows an incomplete and scarcely intuitive graphical user interface, which causes slow performance of many options such as layout map [14].

One of the main obstacles to its whole sale adoption by GIS professionals, are the issues of *ease of use* and *portability*. A large number of GIS professionals use MS Windows and Macintosh operating systems. GRASS on these platforms is not without problems. Additionally, the user interface of GRASS has lagged behind modern GUI, making it much harder to operate.

Besides of the native GUI also a few alternatives are available. One of them is JGRASS [4; 15] which has been developed to bring GRASS up-to-date as it were, and facilitate its use among a larger number of GIS professionals using a wider range of operating systems (especially MS Windows). Another is QTGRASS, a simple prototype of GUI for GRASS based on QT graphical library, this project seems to be currently not active. There is also project based on wxPython called wxGRASS, detailed information in Spanish are available on-line [9].

2.1 The Goals

All native GUIs for GRASS GIS have been written in Tcl scripting language using Tk graphical toolkit (including TCLTKGRASS, the latest GIS Manager, NVIZ or `v.digit` module user interface).

During the time, limitations of Tcl/Tk toolkit appeared to be fundamental for the future development. This issue has been several times discussed in the GRASS developer mailing list [10]. At the end has been decided to design new native GUI from the scratch using another graphical toolkit.

The main goals of GUI development for GRASS GIS are:

- *Portability*, enabling GRASS to be fully functional on the GNU/Linux, Unix, Mac-OS and MS Windows operating systems.
- *Ease of Use*, providing all the features common to state of the art GIS applications.
- *Extensibility*, all tools integrated into GUI, including digitization, georectification, image classification, etc.

The improved native GUI should help to bring GRASS to production environments as opposed to the research environment. The GUI and improved scripting facility (focused on Python programming language) will allow an application oriented user environment to be focused on the task which need to be performed. The improved GUI should reduce the time of project implementation.

2.2 The graphical toolkit

The major players are QT [11], GTK [12] and wxWidgets [2; 17], the main requirements for graphical toolkit:

– Portability
– Native look and feel
– OpenGL widget available
– Powerful Python binding
– Performance and flexibility

Based on requirements has been decided to use wxWidgets. The wxWidgets library (formerly known as wxWindows) is well-documented, well-known and widely used graphical toolkit. The crucial characteristics connected to the GRASS GUI development:

– Powerful Python binding (know as wxPython).
– Uses native platform SDK (native look and feel), it means that a program compiled on MS Windows will have the look and feel of a Windows program, and when compiled on a GNU/Linux machine, it will get the look and feel of a Linux program.
– Free OpenGL widget (in contrast to QT) which is requested for future development (NVIZ replacement).

For the real development has been chosen Python cross-platform wrapper for the wxWidgets which is known as wxPython [3; 20]. The choice to use Python as the operating platform for this effort comes rather naturally when thinking about modularity, portability, extensibility and reliability. Python as "easy-to-learn", object oriented, currently "very popular" language should enable more people actively contribute on the development in contrast to wxWidgets which works for C++ (or Tcl/Tk used for the old GUI). Moreover wxPython is currently actively developed and maintained.

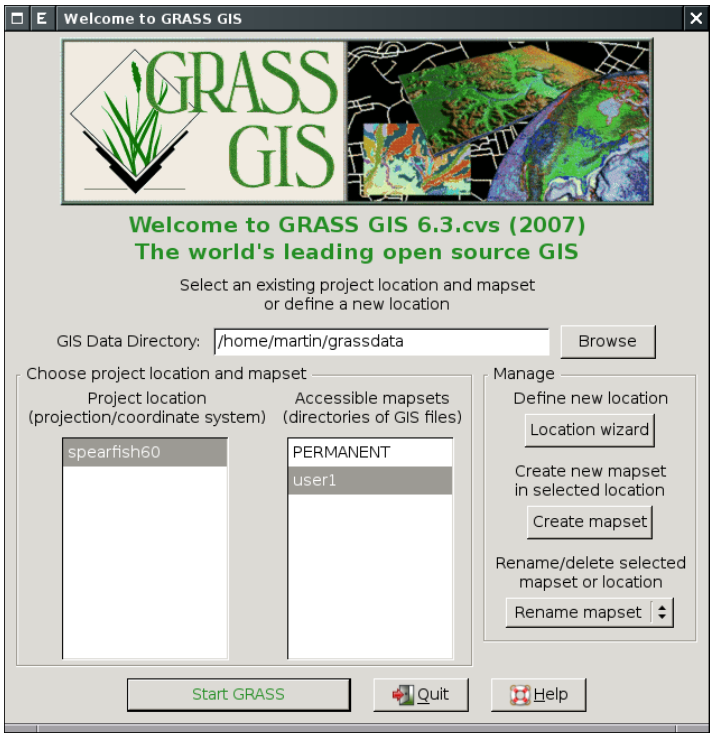
3 The development of wxPython-based GUI

GUI development based on wxPython graphical toolkit started in the end of 2006. The wxPython-based GUI is planned to be default for GRASS 6.4 and GRASS 7. The current version of GRASS is 6.3 [13]. Detailed information are available on GRASS-Wiki page [8].

3.1 The Core GUI Components

The GUI components are implemented as Python modules. They handle all the interactions between users and data. The core system components are:

– `MapFrame` (Python module `mapdisp`) which provides Map Display Window(s) to display a map composition.
– `GMFrame` (Python module `wxgui`) which provides Layer Manager.
– `AttributeManager` (Python module `dbm`) for GRASS Attribute Table Manager, to browse, select, modify attribute data linked to the vector map layers.
– `Digit` (Python module `digit`) which provides digitization (modifying vector map layers) ability integrated into GUI.
– `Georectify` (Python module `georect`) for integrated georectification tool.
– `GrassGUIApp` (Python module `menuform`) for GUI dialogs created on-the-fly based on XML output from the GRASS parser.
– `ProfileFrame` (Python module `profile`) which provides integrated Profiling tool.
– `HistFrame` (Python module `histogram`) which provides histogramming of raster map layer.



Welcome to GRASS GIS
GRASS GIS
Welcome to GRASS GIS 6.3.cvs (2007)
The world's leading open source GIS
Select an existing project location and mapset
or define a new location
GIS Data Directory: /home/martin/grassdata
Browse
Choose project location and mapset
Project location (projection/coordinate system)
Accessible mapsets (directories of GIS files)
spearfish60
PERMANENT
user1
Manage
Define new location
Location wizard
Create new mapset in selected location
Create mapset
Rename/delete selected mapset or location
Rename mapset
Start GRASS
Quit
Help

Fig. 6. Welcome to GRASS GIS – The start-up screen.

– `GWizard` (Python module `location_wizard`) for creating new GRASS locations.

The GUI is composed by two main components:

– *Layer Manager* which allows users to run different GRASS modules from menu, includes map layer management, integrated command-line prompt, and command output window.
– *Map Display Window(s)* which integrates basic tools for zooming, panning, data querying, decorations (north arrows, barscale, etc.). This component replaces the old X11-based GRASS monitors. The user is allowed to start various Map Display Windows during one session. The Layer Manager registers started Map Display Windows using different tabs.

Layer Manager. The window frame can be visually split into four parts. In the top-most part of the window is located menu bar which allows to run GRASS commands via graphical dialogs (e.g. see fig. 11), the bottom-most part is occupied with interactive command line prompt which allows to run GRASS modules by typing command including all parameters and flags. Below the menu bar is placed toolbar. The rest of the window is occupied by notebook widget with two tabs. The first tab "Map layer for each display" contains sub-tabs "Display x" for each started Map Display Window instance. Each tab contains list of map layers which are rendered in the given Map Display Window. Properties of map layers in the layer tree can be changed via contextual (pop-up) menu (see fig. 7). The second tab "Command output" contains log area for output messages of commands launched from the menu or from the command-line prompt (see fig. 8).

Map Display Window. This component contains toolbar area (set of toolbars), map canvas where a map composition is displayed, and statusbar.

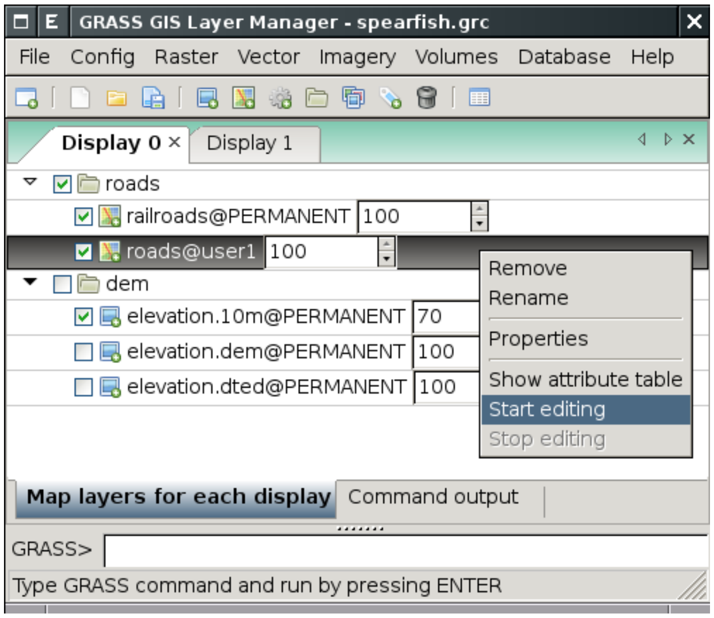
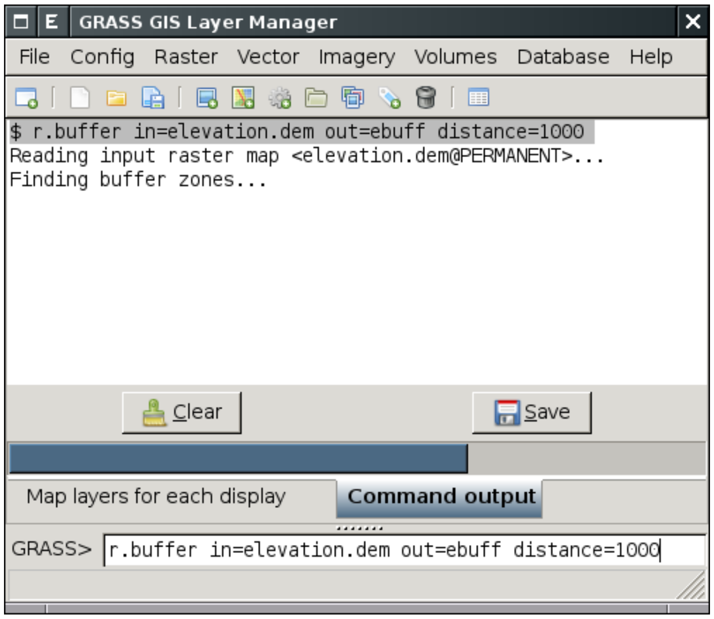

GRASS GIS Layer Manager - spearfish.grc
File Config Raster Vector Imagery Volumes Database Help
Display 0
Display 1
roads
railroads@PERMANENT 100
roads@user1 100
dem
elevation.10m@PERMANENT 70
elevation.dem@PERMANENT 100
elevation.dted@PERMANENT 100
Remove
Rename
Properties
Show attribute table
Start editing
Stop editing
Map layers for each display
Command output
GRASS>
Type GRASS command and run by pressing ENTER

Fig. 7. Layer Manager – Map layers tree tab.


GRASS GIS Layer Manager
File Config Raster Vector Imagery Volumes Database Help
Clear
Save
Map layers for each display
Command output
GRASS>

```
$ r.buffer in=elevation.dem out=ebuff distance=1000
Reading input raster map <elevation.dem@PERMANENT>...
Finding buffer zones...
```

```
GRASS> r.buffer in=elevation.dem out=ebuff distance=1000
```

Fig. 8. Layer Manager – Command output tab.

Except of "main" toolbar with basic tools like zooming, panning, data querying, etc. are available optional toolbars dedicated to digitization (see fig. 9), under active development is also Georectification tool.

Statusbar of Map Display Window has several modes (see fig. 10):

– coordinates: current coordinates

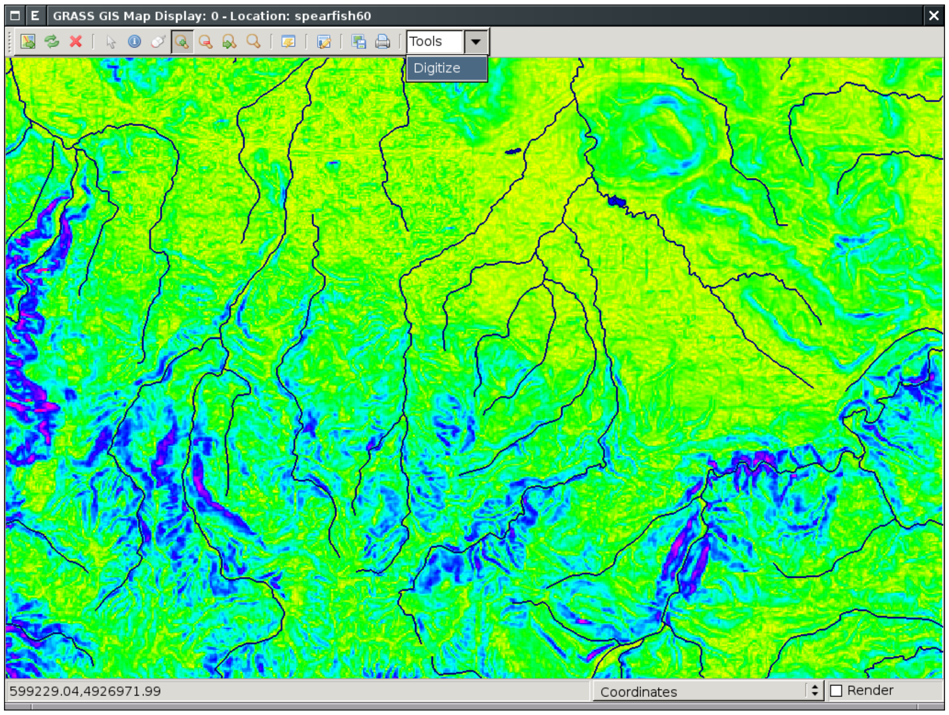


Fig. 9. Map Display Window with additional toolbars.

- extent: region extent W-E-S-N
- geometry: window geometry (number of rows and cols, resolution in north-south and east-west direction)
- map scale: approximate map scale (editable)

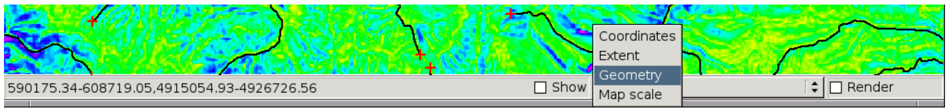


Fig. 10. Changing statusbar mode.

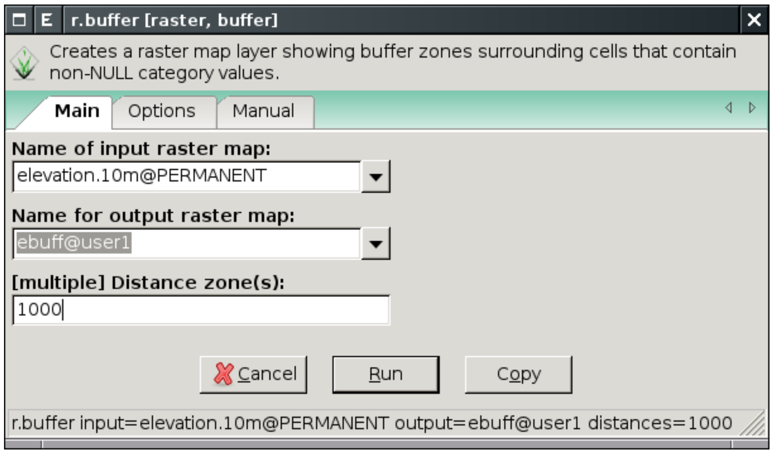
Graphical dialog. Dialogs are generated for all GRASS modules on-the-fly based on XML output from the GRASS parser. Dialog for `r.buffer` module is presented on fig. 11.

3.2 Digitization tool

One of the key GUI components – *Digitization tool* – is currently under active development. The tool is fully integrated into GUI and replaces the current Tcl/Tk-based `v.digit` module.

The functionality of `v.digit` module has been already re-implemented. Moreover new requested[1] features have been implemented:

[1] Development of digitization tool is supported by the GFOSS-TN project (GFOSS Migration Project, Comune di Trento).



Creates a raster map layer showing buffer zones surrounding cells that contain non-NULL category values.

Name of input raster map: elevation.10m@PERMANENT

Name for output raster map: ebuff@user1

[multiple] Distance zone(s): 1000

```
r.buffer input=elevation.10m@PERMANENT output=ebuff@user1 distances=1000
```

Fig. 11. GUI dialog for `r.buffer` module.

- Display vertices, animate movement of vector object by mouse
- Select vector objects by box
- Select vector objects by query (line length, dangles)
- Snap to node or vertex
- Snap to the vector objects (node or vertex) from background vector map layer(s)
- Unsplit (remove pseudo-nodes) – merge selected vector lines
- Connect selected lines (undershooted lines)
- Break selected lines
- Copy vector objects from background vector map layer(s)
- Z bulk-labeling (automated z-coordinate assignment to selected vector lines, e.g. vector contours). This functionality was available in old CLI-driven `v.digit` in GRASS 5.0, never re-implemented in GUI-driven `v.digit` module.
- and others...

On fig. 12 and 13 is presented one example: first, all vector lines shorter than 1000 map units were selected (highlighted in purple color) and then deleted.

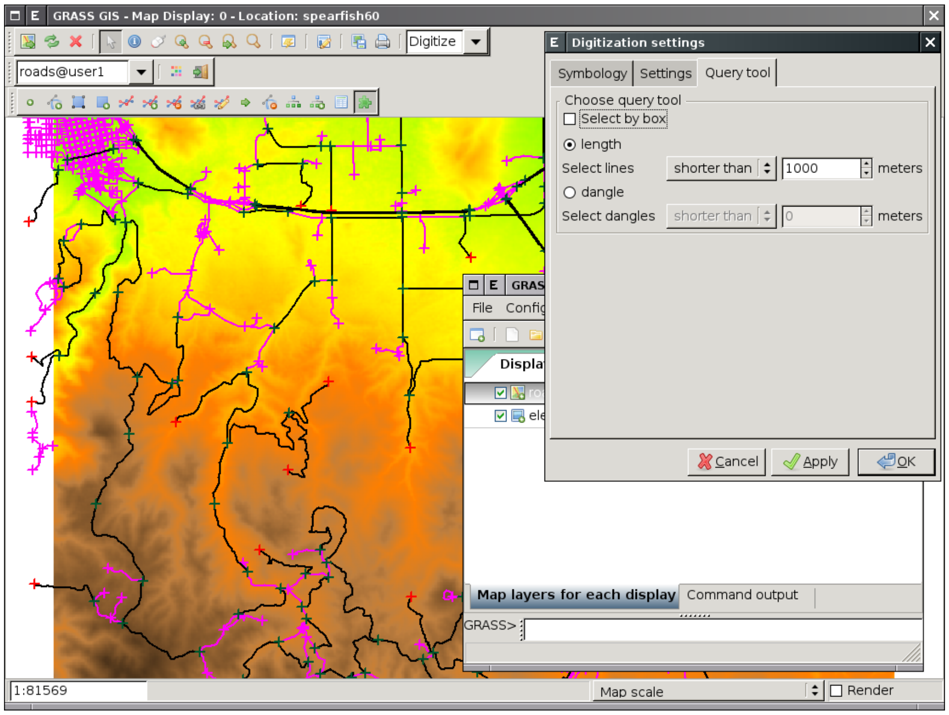
Another example is shown of fig. 14 and 15: copying vector objects from selected background vector map layer.

3.3 The future development

This section describes some of functionality which is planned to be implemented and integrated into the GUI.

Besides Digitization tool, another important GUI component – *Georectification tool* – is currently being developed by Michael Barton.

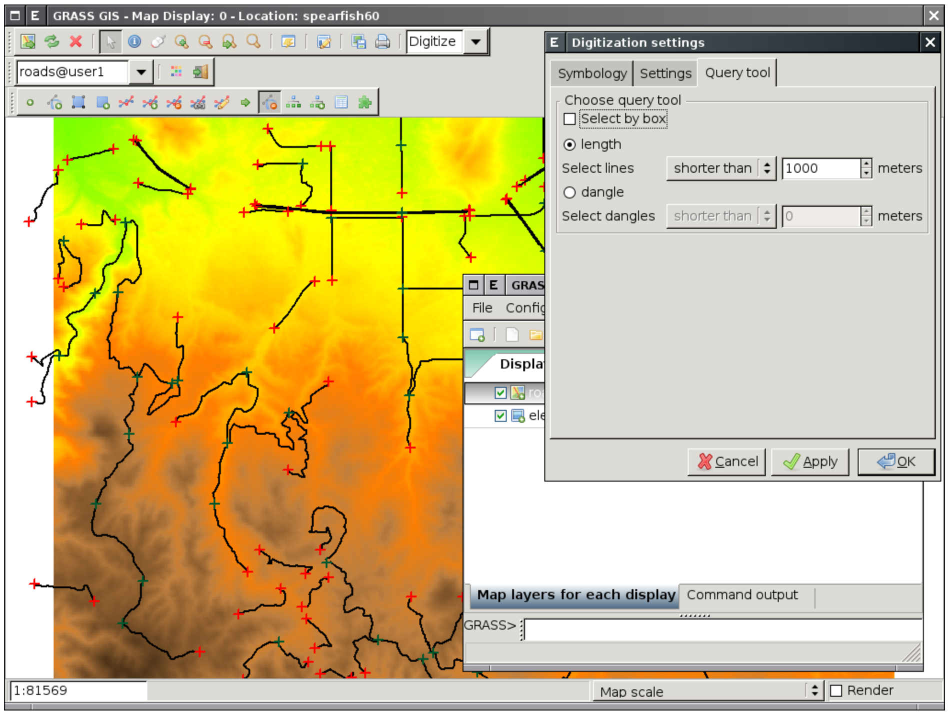
Many other components or sub-components have to implemented or improved including *interactive command-line prompt* – e.g. module search engine (to allow users to find a GRASS module based on the given keywords), location wizard, etc.


GRASS GIS - Map Display: 0 - Location: spearfish60

Digitization settings

Choose query tool
- [ ] Select by box
- (•) length
  Select lines: shorter than 1000 meters
- ( ) dangle
  Select dangles: shorter than 0 meters

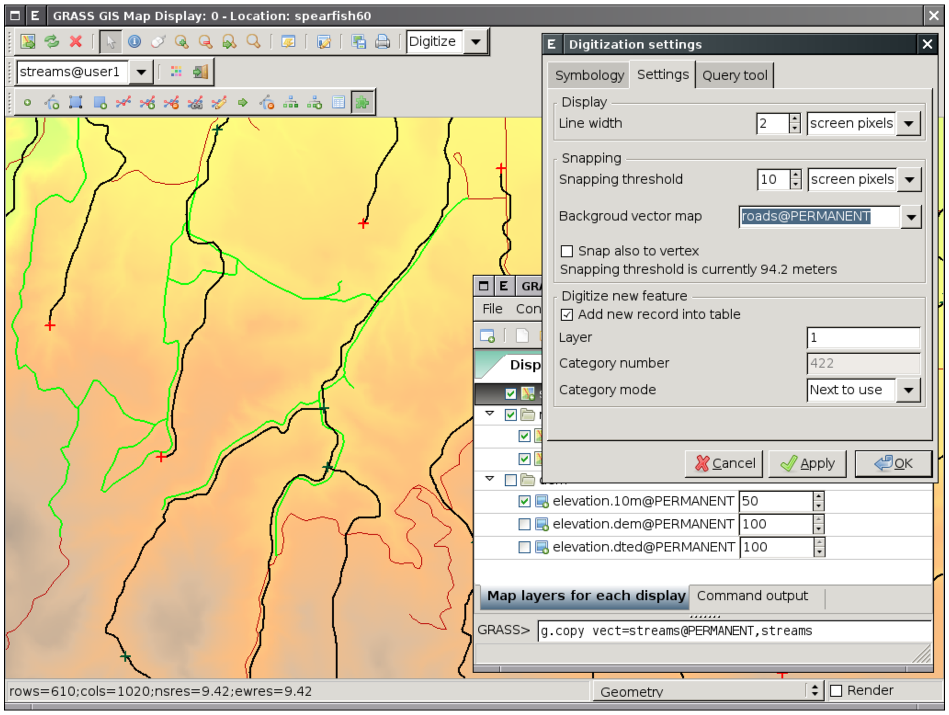
Fig. 12. Digitization tool – Select all vector lines shorter then the given threshold value.


GRASS GIS - Map Display: 0 - Location: spearfish60

Digitization settings

Choose query tool
- [ ] Select by box
- (•) length
  Select lines: shorter than 1000 meters
- ( ) dangle
  Select dangles: shorter than 0 meters

Fig. 13. Digitization tool – Delete all selected vector objects.


GRASS GIS Map Display: 0 - Location: spearfish60

Digitization settings

Display
Line width: 2 screen pixels

Snapping
Snapping threshold: 10 screen pixels
Backgroud vector map: roads@PERMANENT
☐ Snap also to vertex
Snapping threshold is currently 94.2 meters

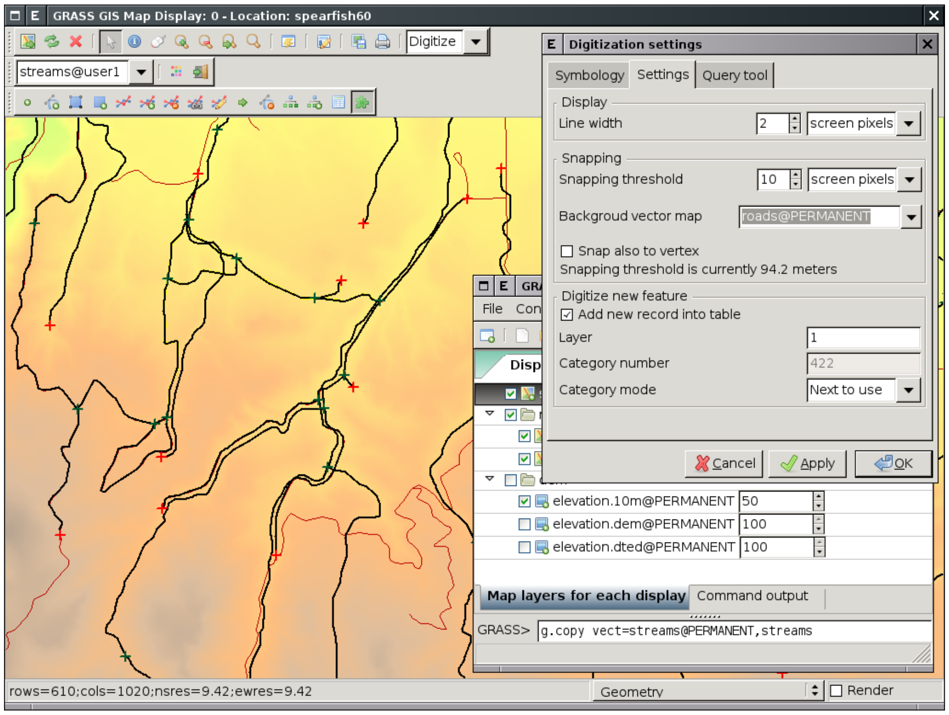
Digitize new feature
☑ Add new record into table
Layer: 1
Category number: 422
Category mode: Next to use

☑	elevation.10m@PERMANENT	50
☐	elevation.dem@PERMANENT	100
☐	elevation.dted@PERMANENT	100

```
GRASS> g.copy vect=streams@PERMANENT,streams
```

rows=610;cols=1020;nsres=9.42;ewres=9.42

Fig. 14. Digitization tool – Select vector objects from background vector map layer.


GRASS GIS Map Display: 0 - Location: spearfish60

Digitization settings

Display
Line width: 2 screen pixels

Snapping
Snapping threshold: 10 screen pixels
Backgroud vector map: roads@PERMANENT
☐ Snap also to vertex
Snapping threshold is currently 94.2 meters

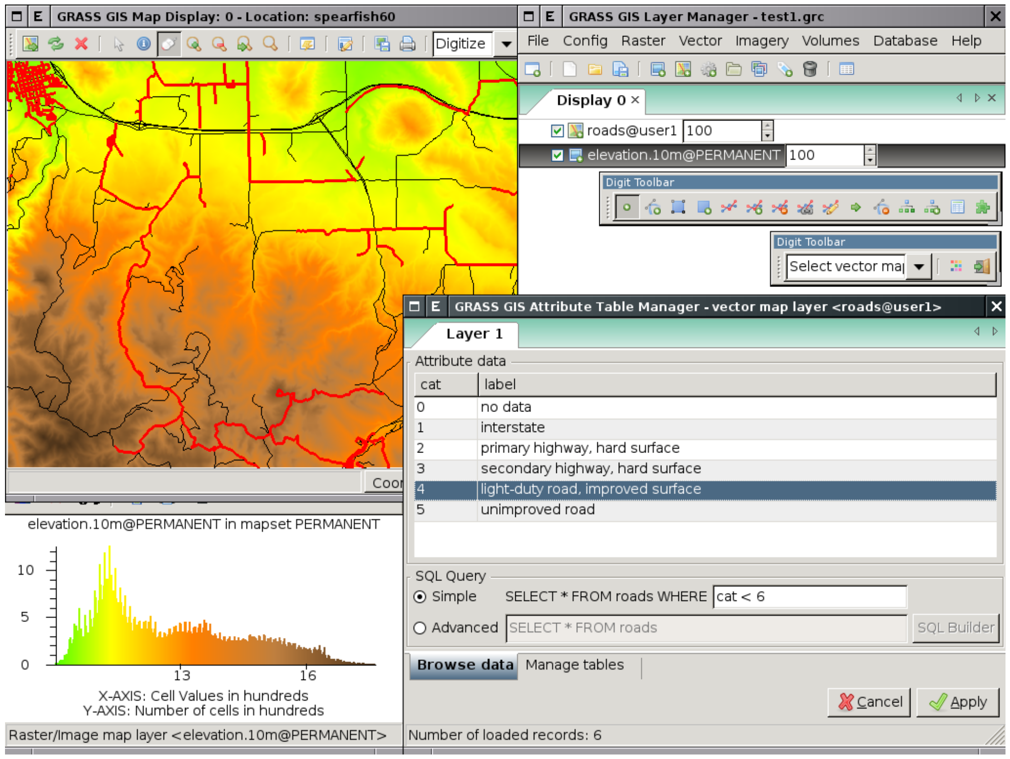
Digitize new feature
☑ Add new record into table
Layer: 1
Category number: 422
Category mode: Next to use

☑	elevation.10m@PERMANENT	50
☐	elevation.dem@PERMANENT	100
☐	elevation.dted@PERMANENT	100

```
GRASS> g.copy vect=streams@PERMANENT,streams
```

rows=610;cols=1020;nsres=9.42;ewres=9.42

Fig. 15. Digitization tool – Copy selected features from background vector map layer.


GRASS GIS Map Display: 0 - Location: spearfish60
Digitize
GRASS GIS Layer Manager - test1.grc
File Config Raster Vector Imagery Volumes Database Help
Display 0
roads@user1 100
elevation.10m@PERMANENT 100
Digit Toolbar
Select vector map
elevation.10m@PERMANENT in mapset PERMANENT
10
5
0
13
16
X-AXIS: Cell Values in hundreds
Y-AXIS: Number of cells in hundreds
Raster/Image map layer <elevation.10m@PERMANENT>

GRASS GIS Attribute Table Manager - vector map layer <roads@user1>

Layer 1

Attribute data

cat	label
0	no data
1	interstate
2	primary highway, hard surface
3	secondary highway, hard surface
4	light-duty road, improved surface
5	unimproved road

SQL Query

[✓ Simple] SELECT * FROM roads WHERE cat < 6

Advanced SELECT * FROM roads

SQL Builder
Browse data
Manage tables
Cancel
Apply

Number of loaded records: 6

Fig. 16. Attribute Table Manager in action.

Future development of Map Display Window component will be focused on the integration of *OpenGL* [5]. OpenGL is a vendor neutral window system independent 3D rendering application programming interface (API) that provides onscreen and offscreen access to the graphics hardware. All OpenGL applications produce consistent visual display results on any OpenGL API-complaint hardware, regardless of operating system or windowing system. For this purpose PyOpenGL [6] will be used. In the result, Map Display Window will support 3D rendering including 3D raster (voxels) and vector data. Functionality will be based on the current NVIZ module (fig. 5).

One of the features which would be good to implemented is *on-the-fly projection*, which automatically converts and overlays spatial data in different projections. Currently the user is forced to re-project datasets in order to overlay and view multiple layers manually.

Another issue is to rewrite module `i.class` dedicated to supervised image classification. This module is X11-driver based and cannot be integrated into the GUI. Moreover it lacks some useful features, e.g. report or save statistics for previously defined training areas (regions of interest). The lack of ability to view previously defined training areas during interactive classification is a major limitation of the module.

3.4 Map Layout

The tools for creating hardcopy maps (map layout) are limited in GRASS because its focus on modeling and spatial analysis [19]. GRASS gives the user ability to add only a very simple and standardized legend, north arrow and scale (modules like `d.barscale`, `d.legend`) to display to the graphics monitor and then export the display to an external image file such as a png. These map features are very basic and cannot be customized.

GRASS also contains specialized module for hardcopy map outputs `ps.map` which works non-interactively. The user is forced to prepare the configuration file for `ps.map` manually. This can be very time consuming task especially for non-experienced user. The module has no graphical front end which would enable users to compose map layout interactively without need to create the configuration file. Experimental prototype of GUI for `ps.map` has been created by Jáchym Čepický, available at http://193.84.38.2/~jachym/index.py?cat=gpsmap.

"Map composer", a tool for hardcopy map outputs is one of the fundamental features which need to be implemented in the wxPython-based GUI.

There are basically two ways of implementation:

- To design graphical front-end for `ps.map` module. The `ps.map` is a cartographic mapping program for producing high quality hardcopy maps in PostScript format. Mapping instructions that describes the various spatial and textual information to be printed must be store in a special configuration file. The GUI for `ps.map` would allow the user to prepare map layout (i.e. configuration file) interactively.
  It would require first of all to design special graphical front-end for `ps.map` module and also various improvements of `ps.map`.
- To improve or basically rewrite cartographic GRASS modules such `d.barscale`, `d.legend` or `d.vect`, etc. In that case Map Display Window would be possible to use for map layout preparation. Then content of map canvas would be simply exported to Postscript format using GRASS PS driver or wxWidgets wxPostScriptDC functionality. Or instead of Postscript to other formats such png, pdf or svg.

It is seems to be more effective to improved or rewrite basic cartographic modules available in GRASS that to design special GUI for `ps.map` including requested improvements of this module. Also `d.vect` module will be rewritten to support line styles, area fill patterns, etc. These changes are connected to the planned improvements in display architecture for GRASS 7 (transparency, floating-point coordinates, etc.).

Another component which is planned to be included is *Map Symbol Editor*. The goal is to allow users to prepare simple cartographic outputs comparable e.g. with ArcGIS map layout functionality.

4 Conclusion

The decision to replace currently used Tcl/Tk with wxPython graphical toolkit is crucial for the future GRASS GUI development. WxPython is one of the most active and maintained wrappers for the wxWidgets library. Moreover it supports more or less all features which are requested for the current GUI development (including fundamental OpenGL widget).

WxPython-based GUI is being actively developed by several people. It seems that Python as a "easy-to-learn" programming language enables more people to actively contribute on the development process (in comparison with Tcl programming language).

The improvements (described in the paper or planned) in graphical user interface could help to bring more GIS users and professionals into the GRASS community, in turn leading to further advances in GRASS and GIS in general and ultimately *benefiting the public*.

Reference

[1] GRASS GIS – Geographic Resources Analysis Support System.
http://www.grass-gis.org.

[2] WxWidgets – Cross-platform GUI library.
http://www.wxwidgets.org.

[3] WxPython – Blending of the wxWidgets C++ class library with the Python programming language.
http://www.wxpython.org.

[4] JGRASS.
http://www.jgrass.org.

[5] OpenGL – Open Graphics Library.
http://www.opengl.org.

[6] PyOpenGL – The Python OpenGL Binding.
http://pyopengl.sourceforge.net.

[7] ESRI – Environmental Systems Research Institute.
http://www.esri.org.

[8] GRASS-Wiki page – WxPython-based GUI for GRASS.
http://grass.gdf-hannover.de/wiki/WxPython-based_GUI_for_GRASS.

[9] wxGRASS (in Spanish).
http://www.um.es/geograf/sigmur/wxgrass/wxGRASS.html.

[10] GRASS developer mailing list.
http://lists.osgeo.org/mailman/listinfo/grass-dev.

[11] QT – GUI software toolkit.
http://trolltech.com/products/qt.

[12] GTK – The GIMP Toolkit.
http://www.gtk.org.

[13] GRASS GIS 6.3 Release.
http://trac.osgeo.org/grass/wiki/Release/6.3.0RC3-News.

[14] Seco G.L., Souto C.M., and Maseda C.R. Barros M.D. Evaluation of GRASS (5.0.3) using the common GIS functionality. *GeoFocus*, 5, 2005. ISSN 1578-5157.

[15] Preston J., Antonello A., Neteler M., and Rigon R. JGRASS, a Java based framework for the GRASS GIS. 2003.

[16] Pytel J. NOP. *Geoinformatics FCE CTU*, Volume 2, 2007. ISSN 1802-2669.
http://geoinformatics.fsv.cvut.cz/wiki/index.php/NOP.

[17] Smart J., Hock K., and Csomor S. *Cross-Platform GUI Programming with wxWidgets*. Prentice Hall PTR, 2005. ISBN 0131473816.

[18] Westervelt J. GRASS Roots. In *FOSS/GRASS Users Conference – Bangkok, Thailand*, September 12-14 2004.

[19] Neteler M. and Mitášová H. *Open Source GIS: A GRASS GIS Approach*. Kluwer Academic Publisher, 2002. ISBN 1-4020-7088-8.

[20] Rappin N. and Dunn R. *wxPython in Action*. Manning Publications, 2006. ISBN 1932394621.

[21] Buchanan R.T. Comparision of GIS software (ArcGIS 9.0 and GRASS 6.0): Implementation and case study. 2005.

[22] Smotritsky Y. Running GIS on Open Source. *Connect: Information Technology at NYU*, Fall/Winter 2004, 2004.