# GRASS GIS wxGUI: current state and outlook

## (from the developers' point of view)

Anna Petrášová
Václav Petráš

OSGeoREL
Faculty of Civil Engineering
Czech Technical University in Prague

April 16, 2013

# Outline

## Additions

- wxIClass (Supervised Classification Tool)
  - potential to become general interface for classification
- Map Swipe tool

## Improvements

- `d.mon` (command line driven map display) has some additional features
- `g.gui.*` modules (command line access to wxGUI tools)

## What is done?

- `wxFrame` (top level window) with one or two maps
- command console (GUI) for writing and running GRASS commands and showing output
- introduction of wxGUI API — GrassInterface (external usage, internally as intermediate layer)

## What is needed?

- separation of map displaying functionality from `wxFrame` classes
- specification of layer manager API
- ...

Continuous refactoring of specific parts of wxGUI simplifies code while keeping the same behavior.

# Specialized wxGUI-based applications

Different tools, components and widgets can be combined to create new applications.

## Examples

- wxIClass (Supervised Classification Tool)
  - simple layer manager
  - map display with two (possibly synchronized) maps
  - digitizer
- Map Swipe tool
  - map display with two maps which behave in special way

Refactoring enables to develop wxGUI-based applications for specific purposes (do not have to be a part of the GRASS source code).

## Tree Model/View

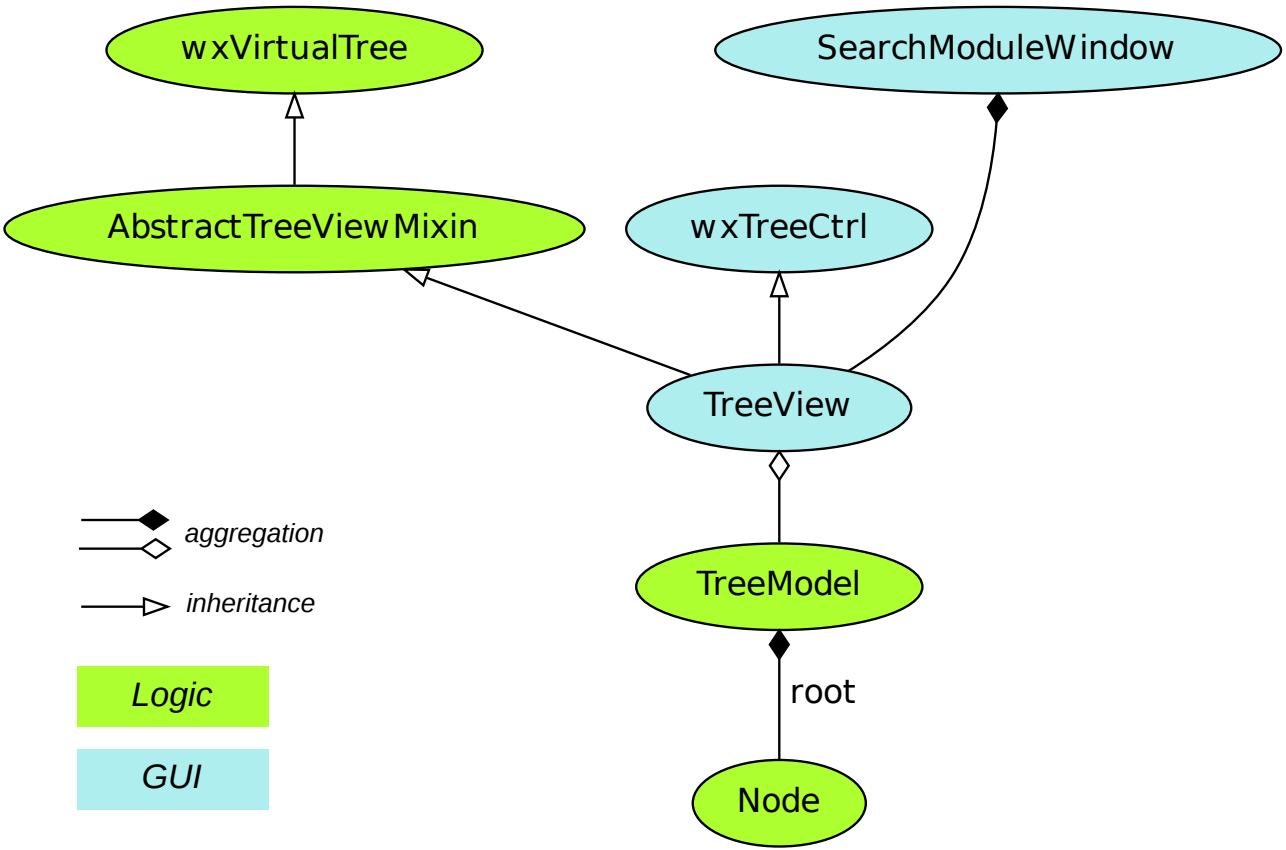
There are many different tree widgets in wxGUI:

- Search module in Layer Manager, Search tree for addons extensions, menus, Query results, Layer tree

Splitting the GUI and logic is possible thanks to wxPython VirtualTree class.

- new simple generic classes written: TreeModel, TreeView
- supports different wx tree widgets
- results in less code duplication, better testing of logic

# Tree Model/View



## The need for signal mechanism

Simple function calls are not sufficient in a large, event driven environment with many persistent objects such as the wxGUI. Using simple function calls leads to tightly coupled code.

## Example code with signals

```
1  # file: gconsole.py
2  from grass.pydispatch.signal import Signal
3  # somewhere in the class
4  self.mapCreated = Signal('GConsole.mapCreated')
5  # in some method which emits signal/event (in some class)
6  self.mapCreated.emit(name=name, ltype=prompt)


50 # file: layermanager.py
51 # connection in some __init__ (in some other class)
52 self._gconsole.mapCreated.connect(self.OnMapCreated)
53 # handler (in the other class)
54 def OnMapCreated(self, name, ltype):
55     doSomethinUseful(name, ltype)
56     # in this case function could be connected directly
```

## Advantages of signals over wxPython events

- No possibility to unbind certain wx event handler.
- Clear syntax:
  - Signal parameters are just function call parameters.
  - It is easy to pass one signal to another signal.
- Two different systems of wx events (`wx.CommandEvent` and `wx.Event`).
- System of wx events propagation is difficult to understand and leads to unreadable code.

The disadvantage of the new system is mixing signals and wx events. However, the usage is generally clear: wx events are used for low level gui-logic communication, signals are for the rest.

## Implementation

- Signals are based on PyDispatcher library written by Patrick K. O'Brien and Mike C. Fletcher.
- Signals API was added to improve and simplify the signal usage. Further it enables a possible switch of underlying library if necessary.

See GRASS 7 Programmer's Manual [1, 2] for detailed information and usage.

[1] `http://grass.osgeo.org/programming7/pydispatch.html`
[2] `http://grass.osgeo.org/programming7/signal_8py.html`

# GUI testing

- doctests

```
>>> signal = Signal('signal')
>>> def echoHandler_(text):
...     print "handler: %s" % text
>>> signal.connect(echoHandler)
>>> signal.emit(text="Hello")
handler: Hello
```

  - can run automatically
  - splitting GUI and logic (MVC architecture)
- `g.gui.*` modules
  - provides command line access to wxGUI modules/components
  - examples: `g.gui.gmodeler`, `g.gui.iclass`
- GUI modules with main

  `python path/to/the/module.py`

  - not available to users (no useful functionality included)
  - available for advanced users for testing purposes

# Single window interface

- current multiple window interface is unusual but has some advantages
- single window interface requested by many
- similarity to GIMP (recently added possibility to choose)

Don't be confused with SDI and MDI.

# Single window interface implementation

- wxGUI code is monolithic
- refactoring will break it into parts
- from parts it will be possible to create both single and multiple window interface
- current state of refactoring is not ready yet for the implementation of single window interface
- we need to decide the layout and behavior of each GUI component which can differ from the current one

## Bug fixing

- most fixed errors come from stricter rules (e.g. using sizers)
- tested on Ubuntu with wxPython 2.9.4 (classic)
- more updates will be probably needed for wxPython Phoenix

## New widgets

- property editor
- info bar (known from web browsers)
- PDF viewer (pyPDF)
- items picker (moves items from one list to another)
- shortcut editor

## Web application

- interface accessible over the Internet or local network
- no local installation on user computer required
- graphical user interface runs in user web browser
- computations run at a server

Do we need Web GRASS?
Yes!
Do we need to implement graphical interface for Web GRASS?
Not really.

## Secure Shell (SSH)

- with forwarding X

  ```
  ssh -X
  ```

- works for many Unix-like OS (including GNU/Linux and Mac OS X)
- usage on some platforms including MS Windows is impossible or limited

## IPython Notebook

- powerful Python command line
- displays images etc.
- not a window-based GUI interface
- works in web browser

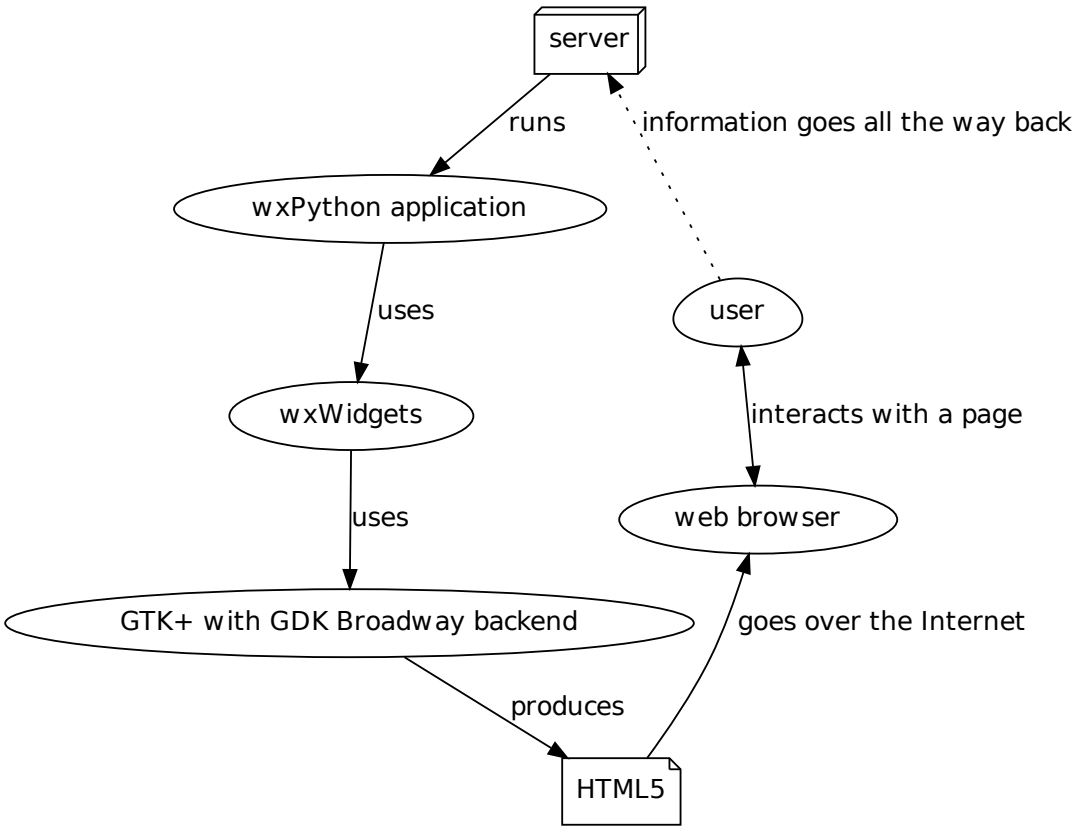
## wxWidgets application in web browser

- wxWidgets 2.9.5 supports GTK+ 3 with Broadway backend
- Broadway backend renders interactive HTML5

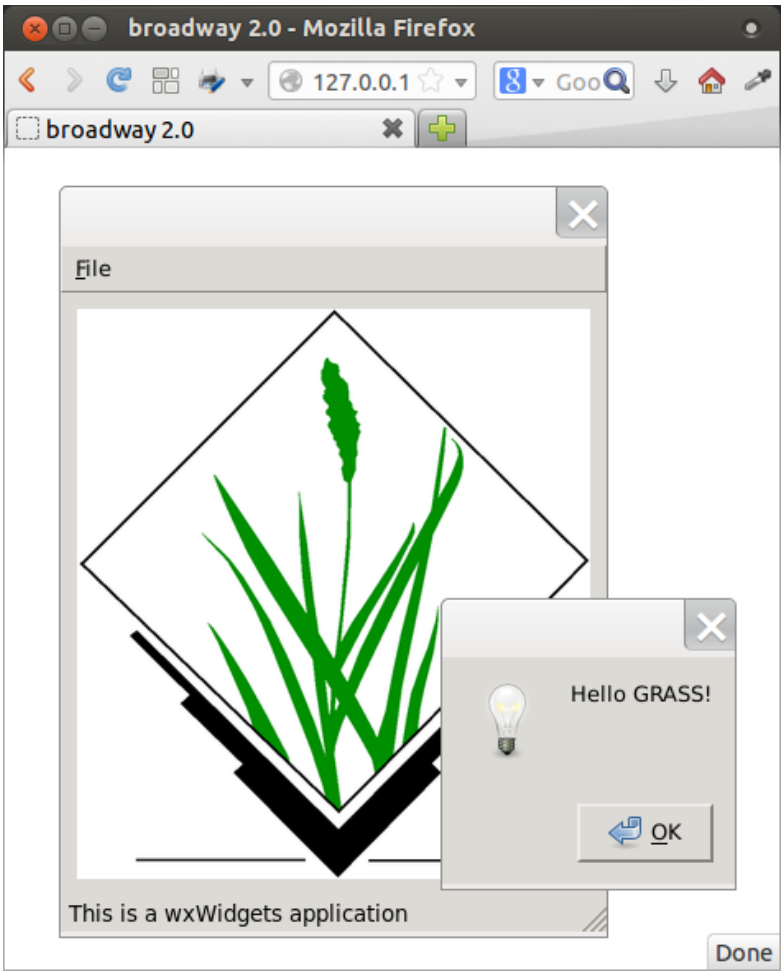
## wxPython

- currently it is not possible to compile wxPython with GTK+ 3
- it is expected to be fixed

# HTML5 interface for web GRASS



# HTML5 interface for web GRASS



## What is needed?

- wxPython support for Broadway
- wxWidgets and GDK Broadway backend improvements
- update GRASS wxGUI for wxPython 2.9
- some smaller changes in wxGUI for Broadway (*webapp mode*)
- standardized HTML5 support in web browsers

## What can we get?

- alternative to live CD/DVDs
- trying GRASS GIS online
- GRASS GIS cloud service
- application for thin client
- wide support on various devices

# Summary

- Refactoring
  - has begun
  - we already have results
  - much more work is needed
- Single window interface depends on refactoring but the work on layout and behavior can begin now.
- Update for wxPython 2.9 is not only necessary but will bring us new possibilities.
- Web GRASS could be available with much smaller effort than expected.