Aalto University
School of Science
and Technology

Geospatial layers and features: from virtual base classes to platforms for planning and modeling

Ari Jolma
Aalto University, Finland

FOSS4G 2010
Barcelona
SEP 6th – 9th

ari.jolma@tkk.fi

OSGeo
Your Open Source Compass

Aalto University
School of Science
and Technology

Requirements of environmental planning and management workflows

- Interaction between baseline data and plan data
  - Baseline data: large quite static datasets, often simple data models
  - Plan data: complex dynamic features, links between features, dynamic meta data
- Support creativity and innovation
- Comparison of plans, impact assessments
- Collecting field data and linking to sensor data sources

ari.jolma@tkk.fi

Aalto University
School of Science
and Technology

Requirements of environmental modeling workflows

- Development of conceptualizations with the help of baseline data
  - From spatial descriptions to process and system descriptions
    - Some things are preserved, some new things are created
- Saving the conceptualizations
- Populating the conceptual models with data and parameter values
  - Analyzing the data
  - Estimating the parameter values
- Linking to scientific data processing and analytical tools

ari.jolma@tkk.fi

Aalto University
School of Science
and Technology

Example

- Consider catchment management (for water quality, erosion, flooding and other problems)
  - Several spatial processes
    - Hydrology
    - Complex land cover and land use changes
  - Numerous management options
    - Effects varies
    - Location and allocation
  - Participatory methods
    - using interactive software systems

ari.jolma@tkk.fi

Aalto University
School of Science
and Technology

Technological conclusions from requirements

- Several technologies have desirable characteristics
  - GIS
  - Spreadsheets (design of computations)
  - CAD (drafting)
  - Analytical tools (statistics for example)
  - Environmental simulation models
- Object-oriented software
  - Inherit those characteristics
- Multi-language software
  - Dynamic creation (planning) of dynamic features etc: Dynamic languages
- Interactive software
  - Planning and design is interactive by nature

ari.jolma@tkk.fi

Aalto University
School of Science
and Technology

Materials and methods

- GDAL
  - Data access
  - Methods (GEOS, GDAL native, ...)
  - Foreign function interface (SWIG API)
- Cairo, GTK+, GNOME
  - Multiple output target 2D graphics
  - GUI toolkit
  - Software desktop
- Perl
  - High-level programming language

ari.jolma@tkk.fi

Aalto University
School of Science
and Technology

GDAL FFI: Case Perl

- GDAL foreign function interface (FFI) has a common OO API, built on top of the C API
- Some adjustments for Perlishnesses
- Objects in guest language link to C++ objects in GDAL
  - GDAL provides stand-alone geometry and feature objects besides objects for features and geometries in a data store
  - Most of part_of links recreated in Perl to prevent core dumps due to auto-destroy of objects
- Can use Perl subs as callbacks

ari.jolma@tkk.fi

Aalto University
School of Science
and Technology

OO in Perl

- An object in Perl is a variable that is “blessed” into a package (namespace)
  - Developer may add compilation units into a package as she wishes
- Perl variables may be dynamic and complex or opaque links to objects in underlying code (C++ for example)
  - Thus it is easy for example to a Perl object contain other objects – also dynamically
- Perl supports multiple inheritance like it supports many other things
- Run-time evaluation of code means for example that we allow methods to be called ad-hoc by end-user

ari.jolma@tkk.fi

Aalto University
School of Science
and Technology

Results

- Generic geospatial layer class (Gtk2::Ex::Geo::Layer)
  - Styling information, Dialogs
  - API for interaction, basic screen behavior, interacting with *features*
- Map canvas widget class (Gtk2::Ex::Geo::Overlay)
  - Subclass of Gtk2::ScrolledWindow
  - + contains image, event handling, rubberbanding, layers
    - Image is a pixmap, which is made from a pixbuf, which is used as a Cairo surface
- Glue class (Gtk2::Ex::Geo::Overlay)

ari.jolma@tkk.fi

Aalto University
School of Science
and Technology

The glue class

- Holds a GUI together
- Layer classes register with it to announce their capabilities
  - Menus, commands, dialog boxes, interface elements, “variable upgrading”
- Text entry to
  - create objects and
  - send methods to objects
- Link views of the objects (map canvas, list of layers)
- Manage menus, buttons etc.

ari.jolma@tkk.fi

Aalto University
School of Science
and Technology

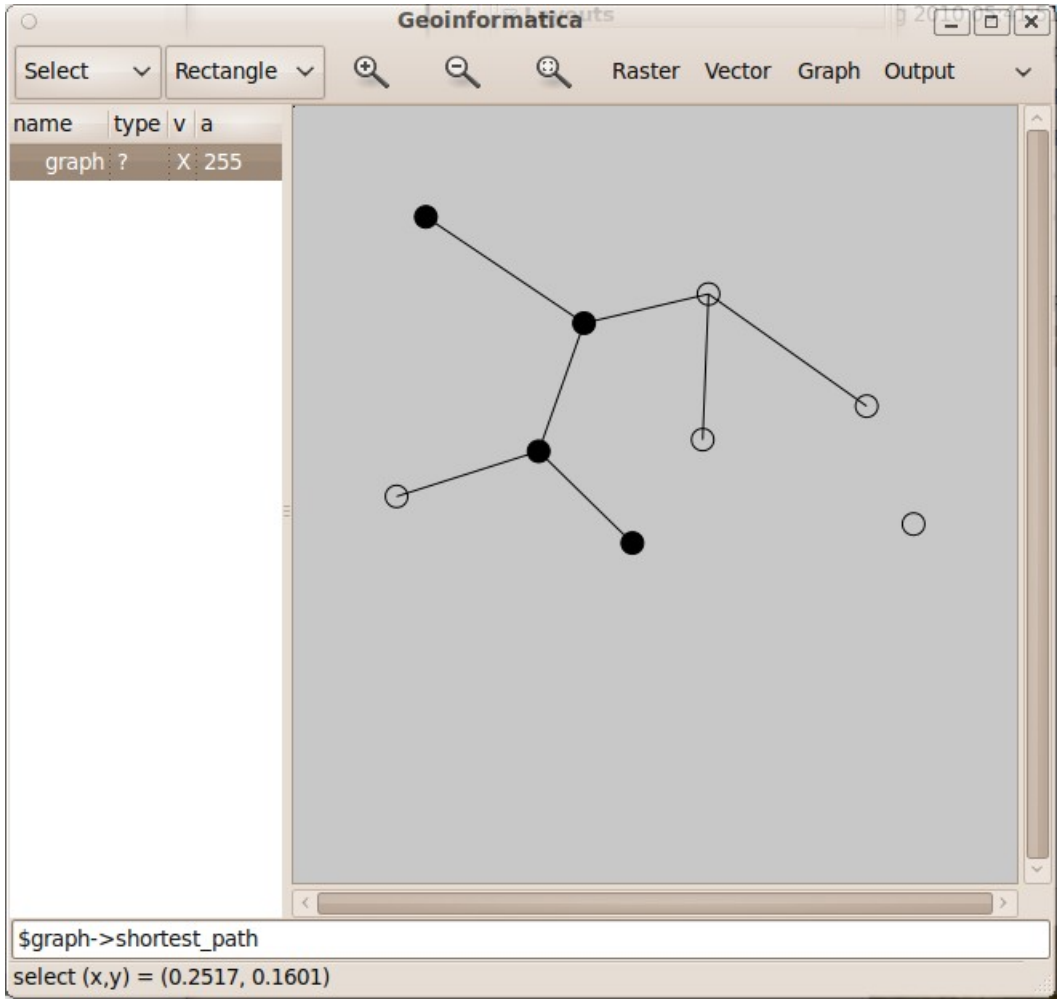
Layer subclasses

- Geo::Raster::Layer and Geo::Vector::Layer
  - Layer classes for wrapped GDAL raster bands and OGR vector layers (and more)
  - Support the traditional desktop GIS paradigm
- Gtk2::Ex::Geo::Graph
  - Wraps Perl module Graph (module for creating abstract data structures called graphs, and for doing various operations on those)
  - Adds required methods, save/open, elementary design

ari.jolma@tkk.fi

Aalto University
School of Science
and Technology

Working with a network




```
sub shortest_path {
    my($self) = @_;
    my($u, $v);
    for my $x (@{$self->selected_features()}) {
        next unless ref $x eq 'HASH';
        $u = $x,next unless $u;
        $v = $x unless $v;
        last;
    }
    $self->select();
    return unless $u and $v;
    my @path =
        $self->{graph}->SP_Dijkstra($u, $v);
    $self->selected_features(\@path);
}
```

ari.jolma@tkk.fi

FOSS4G 2010 Barcelona SEP 6th – 9th

OSGeo Your Open Source Compass

Aalto University
School of Science
and Technology

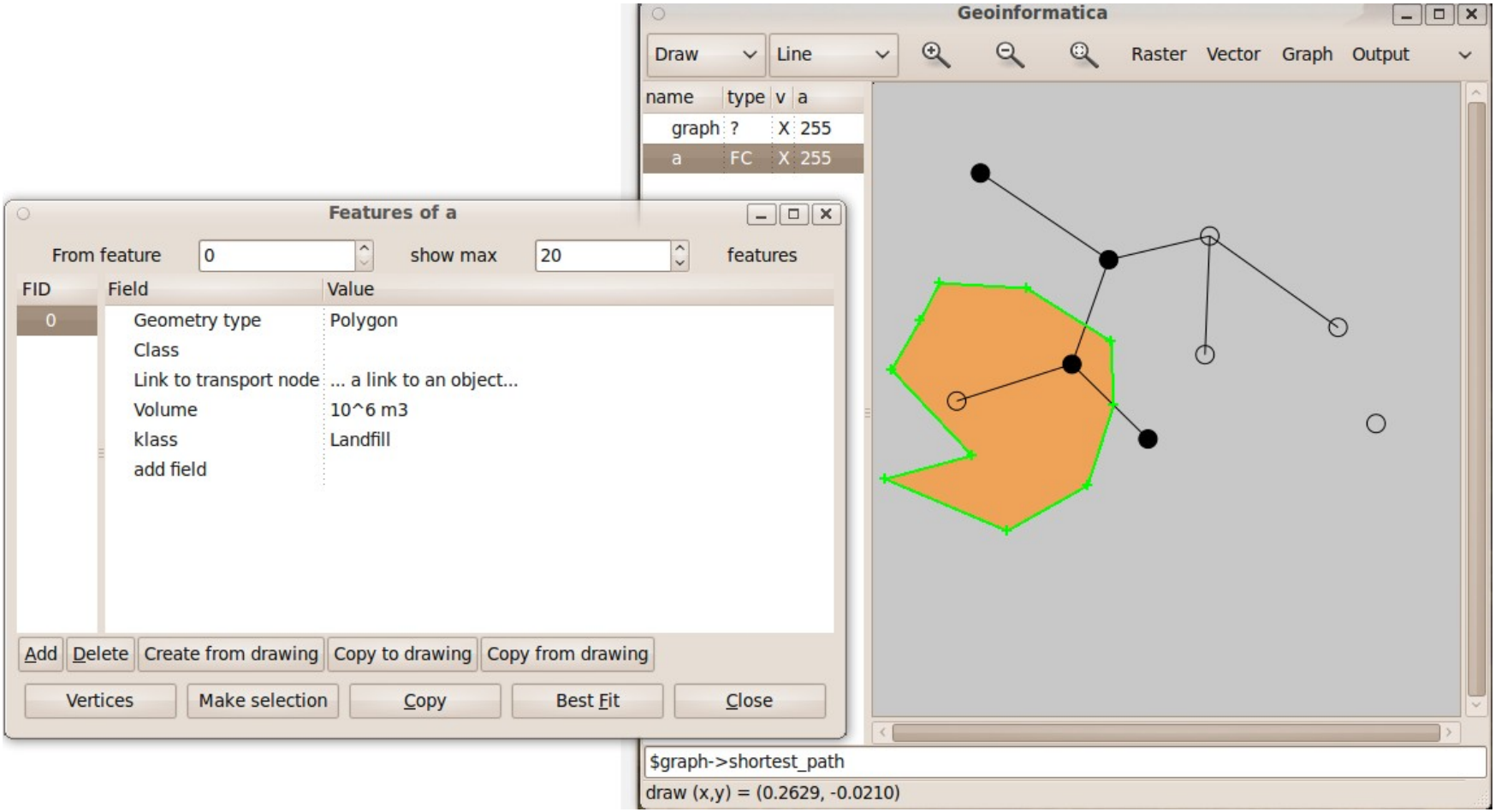
A base class for complex features

- Geo::Vector::Feature
  - A complex alternative to Geo::OGR::Feature
  - Currently a Geo::Vector object may contain
    - an OGR layer = OGR features in an OGR data source, or
    - a Perl array of these features
  - The geometry is an OGR geometry (stand alone)
  - Has methods for save as / create from GeoJSON
    - Uses Perl JSON::XS module
- Still a stub
  - for example linkages between features not yet considered

ari.jolma@tkk.fi

Aalto University
School of Science
and Technology

Designing a feature



FOSS4G 2010 Barcelona SEP 6th – 9th

ari.jolma@tkk.fi

OSGeo Your Open Source Compass

Aalto University
School of Science
and Technology

Conclusions

- Can base classes for features and layers lead to better interoperability between many simple and few complex (as required by this use case)?
  - This work seems to support that idea
- Dynamic languages support design that involves creating new items to a plan / analysis
- FOSS is useful in developing and testing ideas
  - Implementations of tools are easy to integrate
  - Vertical interoperability between high level languages is a problem
  - New OO systems can be built as FFI

ari.jolma@tkk.fi

Aalto University
School of Science
and Technology

Further work

- Temporal dimension
  - Layers that represent processes
- Tools for the planning interaction
  - For example drafting tools could be taken from some FOSS drawing program
- Can the text entry be extended into a spreadsheet
  - Currently new non-spatial variables “disappear”
  - (GNOME) Spreadsheet widget could be easy to integrate
- Case studies
- Visualization library (a proof-of-concept exists, written in C)
  - Cairo graphics
  - GDAL

FOSS4G 2010
Barcelona
SEP 6th – 9th

ari.jolma@tkk.fi

OSGeo
Your Open Source Compass

Aalto University
School of Science
and Technology

Thank you for your attention!
ari.jolma@tkk.fi

OSGeo
Your Open Source Compass