GEOSPATIAL LAYERS AND FEATURES: FROM VIRTUAL BASE CLASSES TO PLATFORMS FOR PLANNING AND MODELING

Ari Jolma
Aalto University
Niemenkatu 73, 15140 Lahti, Finland

INTRODUCTION

Geographic information system (GIS) is traditionally defined as a set of computer tools capable of storing, manipulating, analysing and retrieving geographic information (Burrough, 1986), while designing objects, including geospatial objects, is specifically the domain of computer-aided design computer technology, and simulating dynamic systems, including geospatial systems, is the domain of simulation model software. A desktop GIS typically is optimized for managing collections of geospatial data in raster and vector formats, accessing such data from servers in the internet, making it available to various geospatial analysis methods, and presenting the data to the user. Presentation of geographic data draws usually on cartographic traditions, whether it is for analytical purposes (MacEachren et al. 2004) or common mapping. User interaction with data in GIS is often limited to simple queries and editing of details. There is an observed need in merging GIS more with design (e.g., the ”GeoDesign” concept recently promoted by Jack Dangermond) and modeling domains (for a review of the issue of linking dynamic models with GIS see for example Brimicombe (2009)). The fundamentals of representation of geographic data in geospatial software remains a research issue. Goodchild et al (2007) present ”geo-atom”, a tuple of point in space-time, property, and property value at the point, as a concept on which all other forms of geographic data can be based on.

In this paper we describe and discuss the virtual base classes for geospatial layers and geospatial features that we have implemented in the Perl modules of the Geoinformatica FOSS4G stack. The motivation for the present work is to build a unified foundation on which to develop geospatial software for spatial planning, design, and modeling.

MATERIALS AND METHODS

The Geoinformatica FOSS4G stack is based on GDAL and associated tools (PROJ4, GEOS, data source drivers), Perl and Perl modules, especially those developed as part of Geoinformatica, and GTK+ user interface toolkit. GDAL provides a unified interface to several data sources (files, databases, servers) (raster and vector sources separately), tools for manipulating the data, and a foreign language interface for using the library from other languages. The Geo::GDAL set of modules is the Perl interface to GDAL. The Perl interface to Gtk2 is developed by the gnome-perl (gtk2-perl) project. The Perl modules Glib, Pango, Cairo, and Gtk2 provide the fundamental Perl interface to GTK+. Cairo is a separately developed vector graphics library.

Perl is a multi-paradigm language, which means that it makes most things possible but enforces few. Object-oriented programming in Perl is based on ”blessed” variables. Variables may be blessed into a module, which is basically a namespace (Geo::GDAL for example). The programmer may send messages to, i.e., call module methods with, blessed variables. Inheritance and multiple inheritance takes place through a module-scoped array, which contains names of the superclasses to this class.

RESULTS AND DISCUSSION

The Perl classes that implement the fundamentals of a Geoinformatica application are

Gtk2::Ex::Geo::Layer, Gtk2::Ex::Geo::Glue, and Gtk2::Ex::Geo::Overlay. The first is a generic class for geospatial layers, the second is a class, which manages a list of layers, and the third is a map canvas widget.

Gtk2::Ex::Geo::Overlay is a subclass of Gtk2::ScrolledWindow, which means that it can be used as a regular GTK+ scrolled window widget. The overlay class contains the image on its window, event handling mechanisms, and the actual layer list. The image is created from a Cairo drawing surface each time the map is drawn. An overlay object handles mouse button and key presses and releases as zoom, pan, select, draw, etc., based on the interaction mode. Based on this interaction the overlay object may also emit signals informing the rest of the application changes in the map canvas. The overlay object also requests from all layers to render themselves on the Cairo drawing surface (using a Cairo context, which can point to, among other things, a pdf document besides an image). The method call contains also the viewport. The overlay class sets the following requirements to layer objects: they must have a name and a world, and they must be able to respond to calls to render themselves. A world of a geospatial layer is the bounding box of its data or events.

Gtk2::Ex::Geo::Glue is a singleton class. A glue object contains an overlay object and other widgets, for example a tree view widget for visual list of the layers. The glue class implements much of the logic that determines what happens when user interacts with the overlay widget, with the layer list, or with separate layers. Most of these actions happen as responses to signals or selections made on a menu. The glue class sets the following requirements to layer objects (not repeating the requirements set by overlay): they must carry visibility and transparency information, and they must respond to got_focus, select, render_selection calls, they also may have dialogs and methods to be exposed to users for example as menu items. The features dialog is the only dialog that is specifically called by the glue object as a response to new_selection signal coming from the overlay. The glue object also asks for information for the layer list widget and tooltips. The dialogs and some of the methods are information that should be available from the layer class as class methods (as opposed to object methods). For this purpose layer classes are registered with the glue object when the application boots.

The requirements set by the Geoinformatica framework for geospatial layers are not very demanding and define the interface only in general terms. This interface is expanded a bit and the basic state variables are implemented by Gtk2::Ex::Geo::Layer, which all layer classes must inherit. The base class for layers contains three sets of information for visual representation of the layer data: symbolization, coloring, and labeling. It also contains dialog boxes for user control for styling. Subclasses may extend styling by simply adding new symbol types or completely overriding some things.

The geospatial feature concept is introduced into the framework already in the map canvas level. This is due to the concept of selecting a feature or features, which is a fundamental human interface task. It is possible to consider a single cell of a geospatial raster a feature, but such code has not (yet) been implemented in Geo::Raster, which is the main raster data class currently.

The subclasses of the base layer class are free to implement their data storage as they wish. Two classes have been implemented on the top of Geo::GDAL classes for regular vector and raster data: Geo::Vector and Geo::Raster.

Interactions between features, complex feature types, and dynamics are important domain characteristics that should be supported by geospatial software tools for these tasks. In this paper we will present virtual base classes for geospatial layers and features. We then present and examine how to inherit new classes for traditional geospatial data and other types of data. The framework for the work is the Geoinformatica geospatial software stack and its graphical user interface core. The

core comprises a set of classes for graphical geospatial software that build on GTK+ and its Perl bindings (gtk2-perl) and Perl itself. The geospatial workhorses of the stack are GEOS and GDAL with its Perl bindings.

Layer functionalities:
- (importing) data for the layer
- rendering the layer
- user interaction
- analytical capabilities

Layer data:
- geographic data (possibly a list of geo features)
- symbolization
- styling
- labeling

geo feature = geometry + a set of (key, value) pairs

Feature functionalities
- edit schema, data, geometry

REFERENCES

Brimicombe, A. (2009), GIS, Environmental Modeling and Engineering, 2nd edition. CRC Press

Burrough, P.A. (1986), Principles of Geographic Information Systems for Land Resources Assessment. Clarendon Press, Oxford.

Goodchild, M.F., Yuan, M. and Cova, T.J. (2007), Towards a general theory of geographic representation in GIS. International Journal of Geographical Information Science. Vol. 21, No. 3, 239–260

MacEachren, A.M., Gahegan, M., Pike, W., Brewer, I., Cai, G., Lengerich, E., Hardisty, F. (2004), Geovisualization for Knowledge Construction and Decision Support. IEEE Computer Graphics and Applications archive. Volume 24 , Issue 1. Pages: 13 – 17.