Building Custom GIS Applications using Open-Source Toolkits – A Case Study

Daniel B. Koch, Ph.D.
Senior R&D Staff
Oak Ridge National Lab
Oak Ridge, TN, USA

kochdb@ornl.gov
http://www.ornl.gov/~ko5
Introduction

- Talk deals with how to get started with a custom development effort
- Motivation for creating your own GIS application
- Decisions to make along the way
- Examples and lessons learned from developing the ORNL Geospatial Viewer (OGV)
Motivation

Several projects needed a simple GIS application for use by non-GIS professionals

Commercial offerings deemed too complicated and/or costly by sponsor

Browser-based offerings had restrictive terms of use and/or copyrighted data

Needed the ability to customize the code for each project
Requirements

- Simple to learn (avoid jargon, feature bloat)
- Allow custom map making and data capture
- Support hardware devices (GPS receiver)
- Run on a laptop for mobile operations
- Liberal license terms
- No per-copy cost
Challenges

- Network connection not always available in the field
- Laptop must carry the data it needs
- User may need to store unstructured or unanticipated data (photos, reports, scanned maps, floor plans, etc.)
- Operation on Windows, Mac, and Linux
Getting Started

Available applications and toolkits
- “Desktop GIS” by Gary Sherman
  - http://desktopgisbook.com

Ideas for use cases
- “Mapping Hacks” by Erle, Gibson, & Walsh
  - http://www.mappinghacks.com

Data sources (region dependent)

Development help (tool dependent)
Application Elements

- Graphical user interface (GUI)
- Geospatial database for spatial queries
- Access to web repositories prior to use in the field
- Hardware drivers
Development Language

Geographic Information Science and Technology

- **Python**
  - Can be used for OS scripting, procedural programming, or object-oriented programming
  - Cross-platform, already installed in many cases
  - Minimalist approach to language elements

- **Resources**
  - [http://python.org](http://python.org)
  - “Learning Python” by Mark Lutz
  - “Python in a Nutshell” by Alex Martelli
  - [http://oreilly.com/python](http://oreilly.com/python)
• **wxPython**
  – wxWidgets C++ library with Python bindings
  – Cross-platform, preserves native OS look and feel
  – Active development and user community
  – Liberal license for personal and commercial use

• **Resources**
  – [http://www.wxpython.org](http://www.wxpython.org)
  – “wxPython in Action” by Rappin & Dunn
Geospatial Database

- **PostgreSQL**
  - Cross-platform, supports spatial queries natively
  - Many extensions (PostGIS)
- **Resources**
  - [http://www.postgresql.org](http://www.postgresql.org)
  - “PostgreSQL” by Douglas & Douglas
- **PsycoPG2**
  - Provides Python DBAPI interface
  - [http://www.initd.org/pub/software/psycopg](http://www.initd.org/pub/software/psycopg)
Data Sources

Geographic Information Science and Technology

- On-board laptop
  - GNIS (http://geonames.usgs.gov)
  - National Atlas (http://www.nationalatlas.gov)
  - LandScan (http://www.ornl.gov/sci/landscan)
  - TIGER (http://www.census.gov/geo/www/tiger)

- Internet access
  - WMS
  - GeoRSS
Hardware Support

- GPS receiver
  - Real-time tracking
  - Downloading waypoints
  - Source of satellite information

- PySerial
  - Cross-platform* Python access to serial port
  - http://sourceforge.net/projects/pyserial

* Windows requires Mark Hammond’s Python extensions (http://www.python.net/crew/mhammond)
Elements of model-view-controller (MVC) used throughout
Graphical User Interface (Linux)

Button bar

Mouse controls pan (drag) and zoom (wheel)

Cursor lat/lon/elev

Database layers pane can be shown or hidden

Icon set by Mark James – http://www.famfamfam.com/lab/icons/silk
Mac OS GUI
Geographic Information Science and Technology

[Map of North America with satellite view and geographic coordinates: Lat: 61.787109°, Lon: -123.574219°, Elev: -]
Windows Vista GUI
### Menu Operations

#### Geographic Information Science and Technology

![Menu Operations Diagram](FOSS4G 2009)

**File**
- New Project... Ctrl+N
- Open Project... Ctrl+O
- Save Project Ctrl+S
- Save Project As...
- Import Feature...
- Print... Ctrl+P
- Quit Ctrl+Q

**Edit**
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Paste Special Shift+Ctrl+V
- Delete
- Select All Ctrl+A
- Group Ctrl+G
- Ungroup Shift+Ctrl+G
- Database Login...
- Project Options...

**View**
- Restore View Ctrl+R
- Zoom Full Extent Ctrl+E
- Export View...
  - Compass
  - Grid
  - Legend
  - Scale
  - Labels
    - Show Hidden Features
    - Reset Layers
  - View Animation
  - Loop
  - GPS Tracking

**Map**
- Get Base Map... Shift+Ctrl+B
- Add Overlay Map... Shift+Ctrl+O
- Add Image... Shift+Ctrl+I
- Add Drawing... Shift+Ctrl+D
- Define Zone Shift+Ctrl+Z
- Define Region Shift+Ctrl+A
- Define Path Shift+Ctrl+P
- Add Placemark Shift+Ctrl+M
- Add Caption Shift+Ctrl+C
- Add Web Link Shift+Ctrl+W
- Clear Cache

**Tools**
- Display Clock
- Convert Geo-coordinates...

**Help**
- Help F1
- Show Tips
- Launch Wizard...
- Web Home
- Check for Updates
- Show Versions
- About
Feature Operations

Geographic Information Science and Technology

Properties and folder
Clipboard ops
Actions
Attributes
Properties

Geographic Information Science and Technology

Dynamic tabs based on input coordinate system

User notes and URL
Make a weather map

Step 1 of 1
Choose a regional or national map below...

Regional weather map
Start regional weather loop

National weather map
Start national weather loop

Instruction pages implemented in HTML
Examples

Geographic Information Science and Technology

- Emergency response planning
- Satellite tracking
- GeoRSS
Emergency Response Planning

Geographic Information Science and Technology

Population updated as circle expands

Vehicle moves along path

Animation showing a hazardous release
Satellite Tracking

Real-time tracking animation using satellite ephemeris

One period of orbit shown

Current position of satellite
Webcams pulled as GeoRSS

Double-clicking webcam icon launches browser camera view

False-color DTED generated map with user-selected features
Lessons Learned

- Make sure you really need a custom solution
- Clearly define a minimum set of functions and data types to support at first
- Pick a language/toolkit you enjoy using
- Allow your code to be organic but constantly refactor into stable bits of functionality
- Avoid feature-creep and excessive options
- Great vehicle for learning GIS concepts
Future Efforts

- GUI improvements
- More simulation capabilities
- 3D viewing via PyOpenGL
- Hardware drivers for other devices
- Animation scripting by end-user