

Using Open Source GIS Toolkits



Web Mapping

Illustrated

O'REILLY®

Tyler Mitchell

Converting and Viewing Maps

While presenting maps on the Web is fantastic, the data for those maps has to come from somewhere. You'll also want to have a toolkit for creating or modifying maps to fit your needs, especially if you're developing in an environment that isn't already GIS-oriented. This chapter introduces end-user applications for viewing and sharing data as well as low-level tools for data access and conversion.

While many other open source GIS and mapping tools exist, this chapter covers the small selection used throughout the remainder of this book. While many other excellent options exist, the sample of tools described here are robust enough for professional use. These free and open source GIS/mapping products are successfully used throughout industry, government, and academia.

For more general information and links to other tools, see the following reference web sites:

<http://freegis.org>

<http://opensourcegis.org>

<http://maptools.org>

If you have the funds or already have the tools, you can, of course, use proprietary GIS software to create the data you'll be presenting with open source software. One of the best features of the open source tools is their ability to work with data created by proprietary applications and stored in proprietary formats.

Raster and Vector

The terms raster and vector are used throughout this chapter. They both refer to specific types of data. *Raster data* is organized as a matrix or grid that has rows and columns; each row/column intersection is a cell or pixel. Each cell has a value, for example, an elevation. Images and digital elevation models are rasters. They are a specific number of pixels high and wide, with each pixel representing a certain size

on the ground; for example, Landsat satellite images are 185×185 km in size. Each pixel is 30×30 m in size.

Vector data is represented as coordinates that define points or points that are strung together to make lines and polygons. This data often has an associated table of information, one for every feature (point, line, or polygon) in the dataset. Keeping these distinctions in mind will help you better understand the remaining parts of this chapter.

OpenEV

OpenEV is a powerful open source desktop viewer. It allows users to explore data in any of the image and vector data formats supported by the Geospatial Data Abstraction Library (GDAL/OGR), which will be introduced later in the chapter. OpenEV can be used for custom image analysis as well as drawing simple maps and creating new data. OpenEV comes as part of the FWTools package, available at <http://fwtools.maptools.org>.

Many users overlook OpenEV's ability to view and navigate in real time through images draped over 3D landscapes. Figure 3-1 shows an example of an image being viewed with OpenEV, and Figure 3-2 shows the same image but modified with one of the OpenEV image enhancement tools.

OpenEV is a good example of how GDAL/OGR capabilities can be accessed through other programming languages. OpenEV is built using Python, which makes OpenEV extensible and powerful because it has the flexibility of that language running behind it. The most powerful feature of OpenEV may be its ability to use Python capabilities within OpenEV; this allows access to GDAL/OGR functions and to any other Python module you need.

MapServer

MapServer is the primary open source web mapping tool used in this book. The main MapServer site is at <http://mapserver.gis.umn.edu>.

There are numerous reasons people decide to use MapServer. One is the ability to make their mapping information broadly accessible to others, particularly over the Internet. Many GIS and mapping analysts need to create custom mapping products for those they support or work for; MapServer makes it possible for users to create maps without needing particular tools installed or assistance from mapping analysts. This in turn reduces the pressure on specialized staff.

Others come to MapServer because it is one of few solutions available for those with diverse data formats. MapServer, through the use of libraries such as GDAL/OGR, can access various data formats without data conversion.



Figure 3-1. A raw Landsat satellite image being viewed with OpenEV

Consider that you could have a collection of 10 different sets of mapping data, all of which need to appear on the same map simultaneously without any of the data being converted from its native format. The native formats can include those used by different commercial vendors. ESRI shapefiles, Intergraph Microstation design files (DGN), MapInfo TAB files, and Oracle spatial databases can all be mapped together without conversion. Other nonproprietary formats can be used as well, including the OGC standards for Geography Markup Language (GML), Web Map Server (WMS), Web Feature Server (WFS), and PostGIS and other databases. The ability to have simultaneous access to diverse data formats on the fly without conversion makes MapServer one of the only options for those who can't (or won't) do a wholesale conversion to a specific format.



Figure 3-2. A Landsat satellite image being viewed with OpenEV and an equalization enhancement

Data Access and Performance

MapServer supports a variety of formats. Some are native to the MapServer executable, while others are accessed through the GDAL/OGR libraries. The latter approach is necessary for formats not programmed directly into MapServer. Access through the libraries adds an extra level of communication between MapServer and the data source itself (which can cause poor performance in some cases).

In general, formats supported natively by MapServer should run faster than those using GDAL/OGR. For example, the most basic format MapServer uses is the ESRI shapefile or GeoTiff image. OGR supports the U.S. Census TIGER file format. The performance difference between loading TIGER or shapefiles can be considerable.

However, using GDAL/OGR may not be the problem. Further investigation shows that the data formats are often the bottleneck. If the data in a file is structured in a way that makes it difficult to access or requires numerous levels of interpretation, it affects map drawing speed.

The general rule of thumb for best performance is to use ESRI shapefile format or Geo-Tiff image format. Because `gdal_translate` and `ogr2ogr` can write into these formats, most source data can be translated using these tools. If you access data across a network, storing data in the PostGIS database may be the best option. Because PostGIS processes your queries for data directly on the server, only the desired results are sent back over the network. With file-based data, more data has to be passed around, even before MapServer decides which pieces it needs. Server-side processing in a PostGIS database can significantly improve the performance of MapServer applications.

Wholesale conversions aren't always possible, but when tweaking performance, these general rules may be helpful.

Portability

MapServer and its supporting tools are available for many hardware and operating systems. Furthermore, MapServer functionality can be accessed through a variety of programming language interfaces, making it possible to integrate MapServer functionality into custom programs. MapServer can be used in custom environments where other web mapping servers may not run.

Because MapServer is open source, developers can improve, fix, and customize the actual code behind MapServer and port it to new operating systems or platforms. In fact, if you require a new feature, a developer can be hired to add it, and everyone in the community can benefit from the work.

MapServer is primarily a viewing and mapping application; users access maps through a web browser or other Internet data sharing protocols. This allows for visual sharing of mapping information and real-time data sharing with other applications using the OGC specifications. MapServer can perform pseudo data conversion by reading in various formats and providing access to another server or application using common protocols. MapServer isn't an analysis tool, but it can present mapping information using different cartographic techniques to visualize the results.

Geospatial Data Abstraction Library (GDAL)

GDAL is part of the FWTools package available at <http://fwtools.maptools.org>. GDAL's home page (<http://www.gdal.org>) describes the project as:

...a translator library for raster geospatial data formats... As a library, it presents a single abstract data model to the calling application for all supported formats.

GDAL (often pronounced *goodle*) has three important features. First, it supports over 40 different raster formats. Second, it is available for other applications to use. Any application using the GDAL libraries can access all its supported formats, making custom programming for every desired format unnecessary. Third, prebuilt utilities help you use the functionality of the GDAL programming libraries without having to write your own program.

These three features offer a powerhouse of capability: imagine not worrying about what format an image is in. With GDAL supporting dozens of formats, the odds are that the formats you use are covered. Whether you need to do data conversion, display images in your custom program, or write a new driver for a custom image format, GDAL has programming interfaces or utilities available to help.

Raster Formats Supported by GDAL

GDAL supports dozens of raster formats. This list is taken from the GDAL web site formats list page found at http://www.gdal.org/formats_list.html.

- Arc/Info Binary Grid (*.adf*)
- Microsoft Windows Device Independent Bitmap (*.bmp*)
- BSB Nautical Chart Format (*.kap*)
- VTP Binary Terrain Format (*.bt*)
- CEOS (Spot, for instance)
- First Generation USGS DOQ (*.doq*)
- New Labelled USGS DOQ (*.doq*)
- Military Elevation Data (*.dt0*, *.dt1*)
- ERMMapper Compressed Wavelets (*.ecw*)
- ESRI *.hdr* labeled
- ENVI *.hdr* labeled Raster
- Envisat Image Product (*.n1*)
- EOSAT FAST Format
- FITS (*.fits*)
- Graphics Interchange Format (*.gif*)
- Arc/Info Binary Grid (*.adf*)
- GRASS Rasters
- TIFF/GeoTIFF (*.tif*)
- Hierarchical Data Format Release 4 (HDF4)
- Erdas Imagine (*.img*)
- Atlantis MFF2e
- Japanese DEM (*.mem*)
- JPEG, JFIF (*.jpg*)
- JPEG2000 (*.jp2*, *.j2k*)
- NOAA Polar Orbiter Level 1b Data Set (AVHRR)
- Erdas 7.x *.LAN* and *.GIS*

In Memory Raster
Atlantis MFF
Multi-resolution Seamless Image database
NITF
NetCDF
OGDI Bridge
PCI *.aux* labeled
PCI Geomatics database file
Portable Network Graphics (*.png*)
Netpbm (*.ppm*, *.pgm*)
USGS SDTS DEM (**CATD.DDF*)
SAR CEOS
USGS ASCII DEM (*.dem*)
X11 Pixmap (*.xpm*)

This list is comprehensive but certainly not static. If a format you need isn't listed, you are encouraged to contact the developers. Sometimes only a small change is required to meet your needs. Other times it may mean your request is on a future enhancement waiting list. If you have a paying project or client with a particular need, hiring the developer can make your request a higher priority. Either way, this is one of the great features of open source software development—direct communication with the people in charge of development.

All these formats can be read, but GDAL can't write to or create new files in all these formats. The web page shown earlier lists which ones GDAL can create.

Programming Libraries

As mentioned earlier, an important feature of GDAL is its availability as a set of programming libraries. Developers using various languages can take advantage of GDAL's capabilities, giving them more time to focus on other tasks. Custom programming to support formats already available through GDAL isn't necessary: reusability is a key strength of GDAL.

GDAL's application programming interface (API) tutorial shows parallel examples of how to access raster data using C, C++, and Python. You can also use the Simplified Wrapper and Interface Generator (SWIG) to create interfaces for other programming languages such as Perl, Java, C#, and more. See <http://www.swig.org/> for more information on SWIG.

The ability to directly link to GDAL libraries has helped add features to an array of GIS and visualization programs both commercial and open source. The GDAL website lists several projects that use GDAL, including FME, MapServer, GRASS, Quantum GIS, Cadcorp SIS, and Virtual Terrain Project.

GDAL Utilities

GDAL also has some powerful utilities. Several command-line data access/manipulation utilities are available. All use the GDAL libraries for tasks such as the following:

gdalinfo

Interrogates a raster/image file and gives information about the file. This command, when given a raster file/data source name, provides a listing of various statistics about the data, as shown in the following code:

```
# gdalinfo vancouver.tif
Driver: GTiff/GeoTIFF
Size is 1236, 1028
Coordinate System is:
PROJCS["NAD83 / UTM zone 10N",
  GEOGCS["NAD83",
    DATUM["North_American_Datum_1983",
      SPHEROID["GRS 1980",6378137,298.2572221010042,
        AUTHORITY["EPSG","7019"]],
      AUTHORITY["EPSG","6269"]],
    PRIMEM["Greenwich",0],
    UNIT["degree (supplier to define representation)",0.01745329251994328],
    AUTHORITY["EPSG","4269"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",-123],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",0],
  UNIT["metre",1,
    AUTHORITY["EPSG","9001"]],
  AUTHORITY["EPSG","26910"]]
Origin = (480223.000000,5462627.000000)
Pixel Size = (15.00000000,-15.00000000)
Corner Coordinates:
Upper Left ( 480223.000, 5462627.000) (123d16'19.62"W, 49d18'57.81"N)
Lower Left ( 480223.000, 5447207.000) (123d16'16.88"W, 49d10'38.47"N)
Upper Right ( 498763.000, 5462627.000) (123d 1'1.27"W, 49d18'58.96"N)
Lower Right ( 498763.000, 5447207.000) (123d 1'1.10"W, 49d10'39.61"N)
Center ( 489493.000, 5454917.000) (123d 8'39.72"W, 49d14'48.97"N)
Band 1 Block=256x256 Type=Byte, ColorInterp=Red
Band 2 Block=256x256 Type=Byte, ColorInterp=Green
Band 3 Block=256x256 Type=Byte, ColorInterp=Blue
```

Various pieces of important information are shown here: image format, size, map projection used (if any), geographic extent, number of colors, and pixel size. All these pieces of information can be very useful when working with data, particularly data that is from an external source about which little is known.

gdal_translate

Translates a raster/image between formats. It has numerous powerful functions such as image resizing, adding ground control points for geo-referencing, and

taking subsets of data. This tool can also manipulate any supported format for other purposes, such as web or graphic design. This is particularly useful when images are very large and not easily handled by other software.

`gdaladdo`

Adds overview levels to a file. This feature improves application performance viewing a file. Applications that are able to read these overviews can then request appropriate resolutions for the display and map scale without loading the full resolution of the file into memory. Instead, portions of an image at reduced resolution are quickly provided.

`gdalwarp`

Takes a source image and reprojects it into a new image, warping the image to fit the output coordinate system. This is very useful when source data isn't in the required coordinate spatial reference system. For example, a geo-referenced satellite image may be projected into UTM projection with meter units, but the application requires it to be unprojected in geographic coordinates (latitude/longitude) measured in degrees.

`gdal_merge.py`

A very powerful tool that takes multiple input images and stitches them together into a single output image. It is a Python script that requires the Python interpreter software to be installed on your system and the GDAL Python module to be loaded. This is a good example of how powerful programs can be built on top of GDAL using higher-level languages such as Python. See <http://python.org/> for more information about the programming language. Recent download packages of FWTools include GDAL and Python as well. See <http://fwtools.maptools.org/>.

`gdaltindex`

Creates or appends the bounds of an image into an index shapefile. You run `gdaltindex` with image names as a parameter. GDAL checks the geographic extents of the image and creates a rectangular shape. The shape and the name of the image files are then saved to the output shapefile. This image index can be used by MapServer to define a single layer that is actually made up of more than one image. It works as a virtual image data source and helps MapServer find the right image efficiently.

In general, GDAL aids in accessing, converting, and manipulating rasters/images. When further programming is done using languages such as Python, GDAL can also serve as a powerful analysis tool. In one sense GDAL can also be used as a rough viewing tool because it allows conversion into commonly viewable formats (e.g., JPEG) for which many non-GIS users may have viewing software.



If you use the GDAL utilities that come with FWTools for Windows, you will have a desktop icon called FWTools Shell. This launches a command window for running these utilities.

OGR Simple Features Library

OGR is part of the FWTools package that's available at <http://fwtools.maptools.org>. The OGR project home page (<http://www.gdal.org/ogr>) describes OGR as:

... a C++ open source library (and command line tools) providing read (and sometimes write) access to a variety of vector file formats including ESRI Shapefiles, S-57, SDTS, PostGIS, Oracle Spatial, and Mapinfo mid/mif and TAB formats.



The historical definition of the acronym OGR is irrelevant today, but it's used throughout the code base, making it difficult to change.

OGR supports more than 16 different vector formats and has utilities similar to GDAL's raster utilities.

Vector Formats Supported by OGR

The following list of the vector data formats supported by OGR was taken from the OGR formats web page at http://www.gdal.org/ogr/ogr_formats.html. The web page also shows which formats can be written or only read by OGR.

- Arc/Info Binary Coverage
- ESRI shapefile
- DODS/OPeNDAP
- FMEObjects Gateway
- GML
- IHO S-57 (ENC)
- Mapinfo file
- Microstation DGN
- OGDI vectors
- ODBC
- Oracle Spatial
- PostgreSQL
- SDTS
- UK .NTF
- U.S. Census TIGER/Line
- VRT: Virtual Datasource

OGR is part of the GDAL/OGR project and is packaged with GDAL. GDAL deals with raster or image data, and OGR deals with vector data. GDAL is to painting as OGR is to connect-the-dot drawings. These data access and conversion libraries cover the breadth of mapping data.

OGR Utilities and Examples

Like GDAL, OGR consists of a set of libraries that can be used in applications. It also comes with some powerful utilities:

ogrinfo

Interrogates a vector dataset and gives information about the features. This can be done with any format supported by OGR. The following code shows ogrinfo being used to show information about a shapefile:

```
# ogrinfo -summary placept.shp placept
Had to open data source read-only.
INFO: Open of `placept.shp'
using driver `ESRI Shapefile' successful.

Layer name: placept
Geometry: Point
Feature Count: 497
Extent: (-140.873489, 42.053455) - (-52.808067, 82.431976)
Layer SRS WKT:
(unknown)
AREA: Real (12.3)
PERIMETER: Real (12.3)
PACEL_: Integer (10.0)
PACEL_ID: Integer (10.0)
UNIQUE_KEY: String (5.0)
NAME: String (50.0)
NAME_E: String (50.0)
NAME_F: String (50.0)
UNIQUE_KEY: String (5.0)
UNIQUE_KEY: String (5.0)
REG_CODE: Integer (2.0)
NTS50: String (7.0)
POP91: Integer (7.0)
SGC_CODE: Integer (7.0)
CAPITAL: Integer (3.0)
POP_RANGE: Integer (3.0)
```

This example shows many vital pieces of information including geographic extent of features, a list of the attributes, their types, and how many features are in the file. Additional parameters can be added that help access desired information more specifically.

Running it in different modes (the example shows summary mode) will reveal more or less detail. A complete listing of all the values and geographic locations of the features is possible if you remove the `-summary` option. You can also specify criteria using standard database query statements (SQL), as shown in the following code. This is a very powerful feature that provides access to spatial data using a common database querying language. Even file-based OGR data sources can be queried using SQL statements. This function isn't limited to database

data sources. For more information on OGR's SQL query capabilities, see http://www.gdal.org/ogr/ogr_sql.html.

```
# ogrinfo placept.shp -sql "select NAME, NTS50, LAT, LONG, POP91 from placept
where NAME = 'Williams Lake'"
```

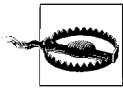
```
OGRFeature(placept):389
  NAME (String) = Williams Lake
  NTS50 (String) = 093B01
  POP91 (Integer) = 10395
  POINT (-122.16555023 52.16541672)
```

ogr2ogr

Takes an input OGR-supported dataset, and converts it to another format. It can also be used to reproject the data while converting into the output format. Additional actions such as filtering remove certain features and retain only desired attributes. The following code shows a simple conversion of a shapefile into GML format.

```
# ogr2ogr -f "GML" places.gml placept.shp
```

This code takes an ESRI shapefile and easily converts it to GML format or from/into any of the other formats that OGR supports writing to. The power of these capabilities is surpassed by few commercially available packages, most notably SAFE Software's Feature Manipulation Engine (FME).



Note that the syntax for `ogr2ogr` puts the destination/output filename first, then the source/input filename. This order can be confusing when first using the tool. Many command-line tools specify input and then output.

OGR, like GDAL, aids in accessing, converting, and manipulating data, specifically vector data. OGR can also be used with scripting languages, allowing programmatic manipulation of data. GDAL/OGR packages typically come with OGR modules for Python. In addition to Python bindings, Java and C# support for GDAL/OGR are in development as of Spring 2005. Perl and PHP support may also be available in the future.

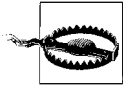


A PHP extension for OGR is available but isn't supported or actively developed. It was developed independent of the main GDAL/OGR project and is available at: http://dl.maptools.org/dl/php_ogr/. If you can't wait for official PHP support through the GDAL/OGR project, give this one a try.

PostGIS

PostgreSQL is a powerful enterprise-level relational database that is free and open source but also has commercial support options. It is the backbone of data

repositories for many applications and web sites. Refrations Research (<http://www.refrations.net>) has created a product called PostGIS that extends PostgreSQL, allowing it to store several types of geographic data. The result is a robust and feature-rich database for storing and managing tabular and geographic data together. Having this ability makes PostgreSQL a *spatial* database, one in which the shapes of features are stored just like other tabular data.



PostgreSQL also has several native geometry data types, but according to Refrations, these aren't advanced enough for the kind of GIS data storage they needed. The PostGIS functions handle the PostGIS geometry types and not the native PostgreSQL geometry types.

This description is only part of the story. PostGIS isn't merely a geographic data storage extension. It has capabilities from other projects that allow it to manipulate geographic data directly in the database. The ability to manipulate data using simple SQL sets it ahead of many commercial alternatives that act only as proprietary data stores. Their geographic data is encoded so that only their proprietary tools can access and manipulate the data.



The more advanced PostGIS functions rely on an underlying set of libraries. These come from a Refraction project called Geometry Engine Open Source (GEOS). GEOS is a C++ library that meets the OGC specification for Simple Features for SQL. GEOS libraries can be used in custom applications and were not designed solely for use with PostGIS. For more information on GEOS, see <http://geos.refrations.net/>.

GIS Analysis with SQL

PostGIS allows you to use SQL statements to manipulate and create geographic data—for example, to buffer points and create circles. This is just the tip of the iceberg. PostGIS can be a GIS in and of itself while at the same time, all the power of PostgreSQL as a tabular database is available. GIS overlays, projecting and reprojecting of features into other coordinate systems, and spatial proximity queries are all possible using PostGIS. It is possible to have all the standard GIS overlay and data manipulation processes available in a server-side database solution. Example 3-1 illustrates the marriage of SQL and GIS capabilities by selecting points contained by another shape. More examples are shown in Chapter 13.

Example 3-1. An SQL command that takes a polygon shape from one table and finds all the points that are within a feature in another table

```
> SELECT town_name
   FROM towns, ontario
  WHERE Contains(ontario_polygon,#first feature
                town_points); #containing the others
```

Example 3-1. An SQL command that takes a polygon shape from one table and finds all the points that are within a feature in another table (continued)

```
town_name
-----
Port Hope Simpson
Hopedale
Makkovik
Churchill Falls
North West River
Rigolet
Cartwright
Tignish
Cheticamp
Sheet Harbour
(10 rows)
```

PostGIS is increasingly used as a tool for map data storage and manipulation. It is ideal for situations in which multiple applications access information simultaneously. Regular processing tasks can be set up while also making the data available to a web mapping site.

Other GIS and mapping tools are able to interact with PostGIS data. OGR, for example, can read/write PostGIS data sources. MapServer can also access PostGIS data. Many people rely on this combination to serve up their maps. Some use custom MapServer applications to add or update information stored in PostGIS-enabled databases. A product called GeoServer (see <http://geoserver.sourceforge.net>) uses the OGC Transactional Web Feature Server (WFS-T) standard for read/write access to PostGIS and other formats. A PostGIS function exists for returning Scalable Vector Graphics (SVG) representations of geometries. The Quantum GIS (QGIS) desktop program (see <http://www.qgis.org>) can also read from PostGIS, and current development will extend that to also allow editing.

The internal programming capabilities of PostgreSQL (using related procedural languages such as PL/Pgsql and PL/Python) allow programmatic access to various PostGIS functions and PostGIS data. Many PostGIS functions allow for a range of manipulation as well as conversion into different data types (e.g., as simple text strings, binary, or even as SVG).

A PostGIS spatial has many of the functions of a normal database—one being that they both use SQL commands to access data. With PostGIS, the information can also include the PostGIS geometry data. PostGIS functions can then manipulate, summarize, or create new spatial data.



You may wonder what the differences are between the spatial component of MySQL databases and PostGIS. The main difference is that MySQL lacks many of the spatial functions found in PostGIS. Also, MySQL doesn't support transactional integrity for spatial features.

Summary of Applications

Table 3-1 summarizes the functions of each application discussed in this chapter: one checkmark shows peripheral use; two checks denotes common use.

Table 3-1. Summary of the types of functions each application generally plays

	GDAL	OGR	PostGIS	OpenEV	MapServer
Viewing and mapping	✓			✓✓	✓✓
Analysis	✓		✓✓	✓	✓
Manipulation	✓	✓	✓✓	✓	
Conversion	✓✓	✓✓			
Sharing			✓		✓✓

Though the applications may fit more than one category, this table is intended to show the most common ways they are used. For some tasks, there are several applications that can meet your needs. Ultimately, your end goal will help determine which is best to use.