Extracted from:

GIS for Web Developers

Adding *Where* to Your Web Applications

Chapter 2

Vectors

In this chapter we talk about getting your hands on vector basemap data. Prepare yourself for a bit of a scavenger hunt—there isn't a single place where you can download everything you need. Once you have it, you'll probably want to see it as well. We download a free viewer so that you can gaze lovingly at the hard-earned results of your work.

2.1 Raw Materials

Most traditional software development projects start from bare dirt—clean, pristine, empty database tables. . . sketches of screens and workflow diagrams on notebook paper and cocktail napkins. . . nothing but hope and potential.

Data is rarely a consideration during the early stages of development. Sure, one of the first steps you generally take is to plan your data structures. You might even create a sample or two of how the data will look for prototyping and testing purposes. But the bulk of the production data is usually generated by the software once it goes live.

GIS projects are unique in that they depend on having some existing data in place. Thankfully you are not expected to draw the outline of the United States or sketch in the highways and cities to the best of your recollection. This preexisting data, called *basemap data*, is generally created and maintained by someone else. Your job as a GIS developer is to find it and incorporate it into the finished product.

For example, let's say you are creating a new system to keep track of your customers. If your goal is to eventually display your customers' locations on a map, you'll need to create a *spatial* field to store their

geographic locations in addition to the usual assortment of string and integer fields. The term *spatial* means "the space around you." (I would have voted for calling it a "location" field, but no one had the foresight to ask me.)

But the spatial field alone is not enough. If the only layer in the finished application is the customer spatial data, all you'll see is a bunch of black dots floating in space over a white background. Although there is *some* information you could glean from this—seeing how your customers are clustered together might be vaguely interesting—seeing your customers in relation to known landmarks such as state boundaries, roads, and airports is probably more valuable. Layering your data over the basemap data puts it in context and gives it meaning. Are you looking at a city block? A county? A state? A country? Even if you really *did* just want to see how tightly clustered your customers are, adding this additional reference information will help.

If you've ever watched the weather report on the evening news, you should be familiar with the idea of map layers. (See Figure 2.1, on the following page.) The newscaster stands in front of a whirling storm system (the data layer) superimposed over a map of the United States (the basemap layer). When the newscaster zooms in for your local forecast, the basemap layers change to counties, cities, and roads.

To put it in programming terms, GIS applications are a series of loosely coupled, highly cohesive map layers. You might say that the rest of this book, and for that matter a large part of the GIS industry, is about combining map layers in new and interesting ways. (Granted, the most interesting data layers will probably end up being the ones you create yourself through data collection or analysis.)

2.2 Raster Data

When it comes to map layers, you need to consider two primary types of data: raster data and vector data.

Raster data is nothing more than a top-down photograph of the earth. It can be an image from a satellite or an aerial photo. Cartographers call it raster data strictly for the intimidation factor—it keeps us from clapping our hands in the middle of a business meeting and saying giddily, "Ohhhh, let's add a pretty picture to the map."

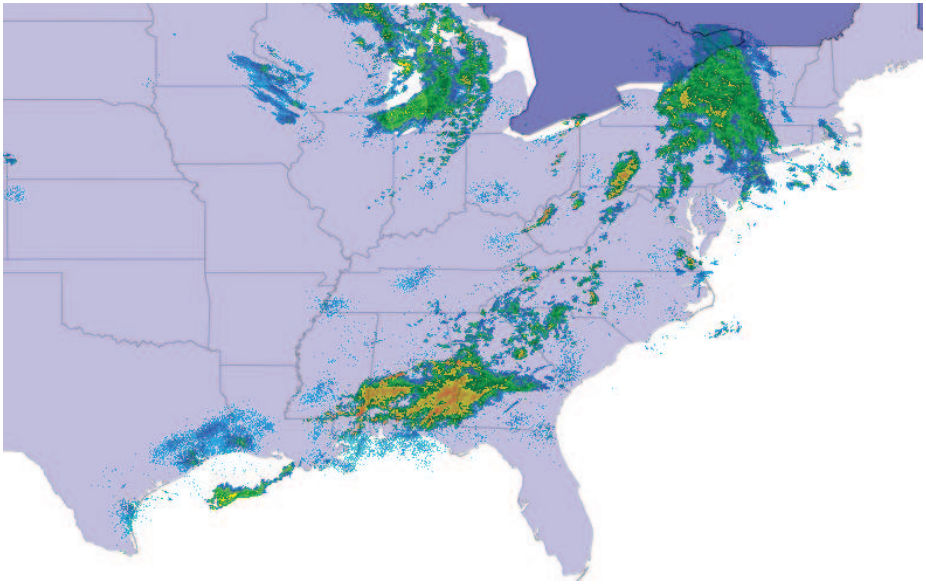

Figure 2.1: A weather map with multiple map layers

What, you want a more precise description than that? OK—the technical definition of a raster is a file that stores its data in discrete cells organized into rows and columns. Think of it as a spreadsheet; however, in this case, the individual cells are the pixels of the photo.

The information stored in the cells could simply be the portrayal information—the red, green, and blue values for each pixel that tells the rendering software how to display it. But it could also be data such as the historical yield of a corn field in bushels per acre. Instead of color information, each pixel contains a value that corresponds to the yield of a specific area on the ground. In that case, the file isn't a photograph at all, even though it's stored in TIFF, which you normally associate with viewable images. You wouldn't ever try to view it directly.

Instead, you'd hand it off to a piece of GIS software for further analysis. Or maybe you'd upload it to your tractor so that it could lay down additional fertilizer in precisely the areas where your field underperformed in the past. (Don't laugh! Do a web search on *precision agriculture* to read case studies about this sort of thing.) Regardless, we're simply using a well-known image file format as a convenient series of buckets to transport our data. So, to be annoyingly precise, all photos are rasters, but not all rasters are photos.

Are you sorry you asked? Don't worry if all of this raster/photo nonsense is confusing right now. It should become clearer when we get to Chapter 4, *Rasters*, on page 73. Why not talk more about it now? Because I said so.

OK, the real reason I'm putting off rasters until later is that oftentimes photographic data is simply not needed. Consider the weather map mentioned earlier. The newscaster probably started with a satellite image of a big cloud, but few people would understand what they were looking at without additional hints. It's only when the newscaster draws big arrows on the screen showing the direction of the storm that we can clearly see what the newscaster is trying to convey.

Similarly, roads are pretty tough to tell apart from the air. And even if you can distinguish one from the other, they might be obscured by clouds or hidden under a canopy of trees. So, the newscaster superimposes the name of the road over the raster layer and outlines it in a bright color to help you get oriented. At this point, the line drawings almost become more important than the photograph itself.

The meteorologist frequently draws in data that doesn't show up at all in photographs, such as wind direction and temperature. Meteorologists even draw in data that doesn't exist for *temporal* (time-related) reasons, such as expected high temperatures and predicted snowfall.

As you can see, the raster data layer plays a minor role in modern weather reporting. It is the raw source of much of the data, but the important stuff (in terms of the finished report) happens in the non-raster layers.

For all of these reasons, we can safely ignore raster data until later chapters. There is no raster data on the road maps in your glove compartment. There is no raster data on the home page of today's most popular mapping websites. (Don't believe me? Go to any of the websites I mentioned at the beginning of Chapter 1, *Introduction*, on page 15.) I'm not saying that raster data is unimportant; I'm saying that we can convey a whole bunch of information without showing actual photographs.

Now, am I saying that satellite imagery isn't an unbearably cool aspect of those websites? Of course not. But after you get over the initial "gee whiz" factor, tell me honestly which view you use more often to get your driving directions. Which view do you print and take with you in the car: the vector or raster view? (It's OK—I knew the answer before I even asked it.)

Getting Oriented

Have you ever stopped to think about what the phrase "getting oriented" really means? When you pull a road map out of your glove compartment, you first generally orient it so that it is "right side up." But the choice of north as up is fairly arbitrary. When you live on a round planet, any side of your map could be considered "right side up."

Early Roman maps used east as their up or orientation direction. Since the sun always rises in the east, it was a natural choice for getting your paper map lined up with the real world. (The English word *orient* comes from the Latin verb *oriens*—to rise.)

Later in Europe, churches were built facing east toward the holy city of Jerusalem. Religious reasons notwithstanding, this established a convenient set of landmarks to help line up their maps at night or on a cloudy day.

So, what was the most obvious choice of names for the Asian countries located to the east of Europe? The Orient, of course.

Once magnetic compasses came into common use, north became the natural direction to orient your map. Here is a tiny device that always points in the same direction—rain or shine, day or night, independent of religious affiliation. What better reason to change the way you line up your map, even if you can't be bothered with changing the description of what you're doing?

For an exercise in disorientation, take a look at some south-side-up maps.* They are quite popular with tourists "down under" in Australia and New Zealand.

*. http://www.flourish.org/upsidedownmap

2.3 Vector Data

The arrows, lines, and dots used by the television meteorologist are all examples of *vector data*, which is nonphotographic line-based data. The earliest maps were comprised of nothing but vector data. The caveman who scratched lines in the sand with a stick was using vector data. Much as painted portraits predate photographs by thousands of years, vector map data predates satellite images.

The question of whether to use raster or vector data on a map is not a question of which is qualitatively better than the other—it is a question of which is more appropriate for the story you are trying to tell.

Earlier we said that raster data stores values in discrete cells. Each pixel in a photograph holds a specific value. Vector data differs in that it stores only *vertices*. In other words, it stores each corner point rather than the entire line. This makes for a much more compact data format, but it is appropriate only for data where discrete values are not required. Think of it this way: vector data is generally appropriate for storing outlines of objects, while raster data is more suited for expressing the content of objects.

A vector outline of a farmer's field is appropriate for showing where it is located in the county. Raster data is more appropriate for doing scientific analysis of the crops growing in the field that year. Showing the results of that analysis, such as areas of the field that yielded significantly more or less than the average, might again be a better candidate for a vector data layer. Neither format is intrinsically better or worse than the other, but one is certainly more appropriate than the other depending on the intended use of the application.

Another important consideration in the raster vs. vector discussion is that vector data is an interpretation or generalization of natural phenomena. It is an abstraction of reality. A photograph of a river shows every twist and turn; a vector representation of the river can be generalized to the point where it is represented by a straight line.

2.4 Types of Vector Data

Three basic types of vector data exist: point, line, and polygon.

Points are the simplest form of vector data. They are dots on a map layer. On a two-dimensional map, points are represented by an (X,Y) coordinate pair. 3D points add a Z coordinate.

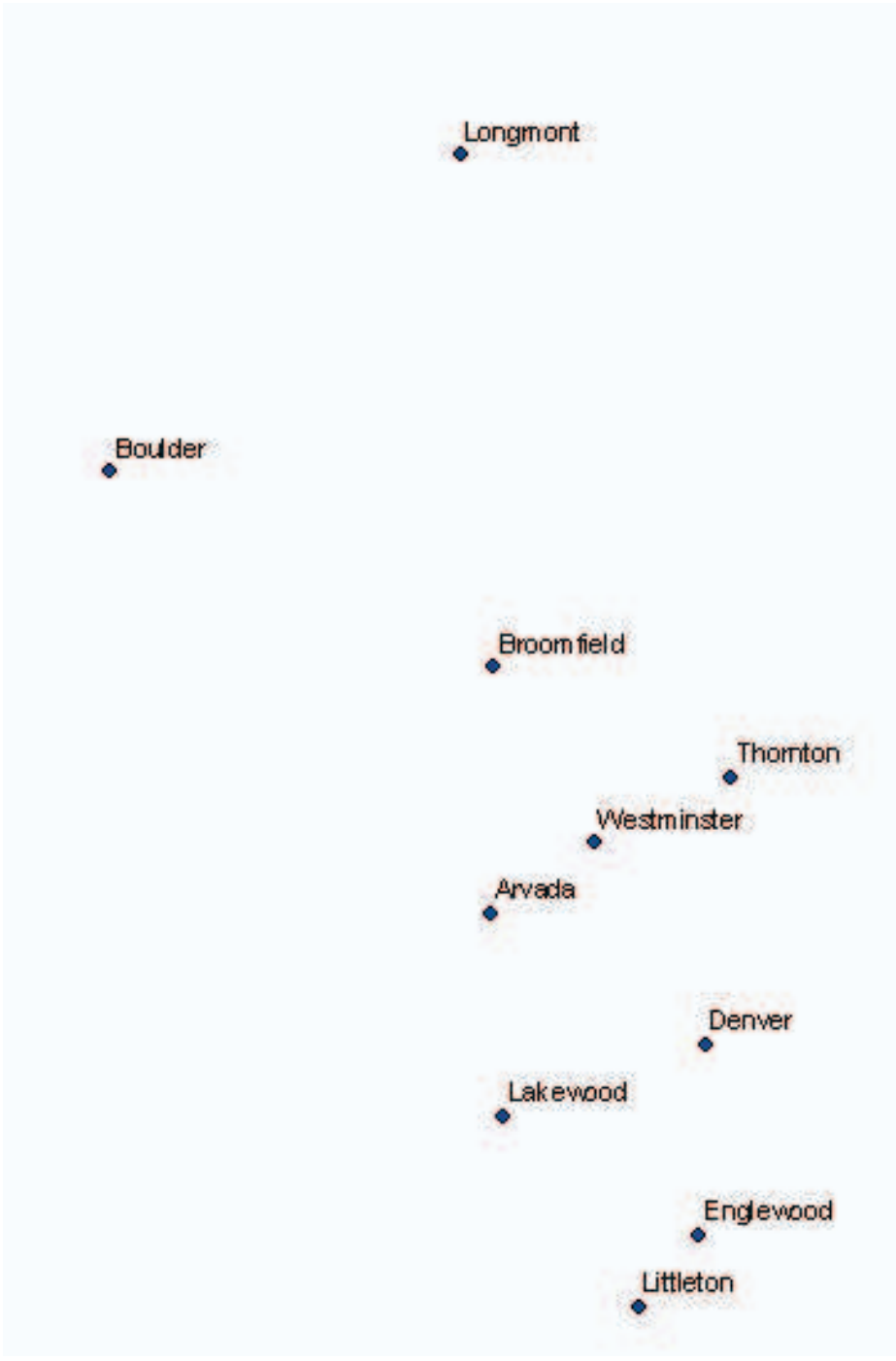

Figure 2.2: Vector points (cities in Colorado)

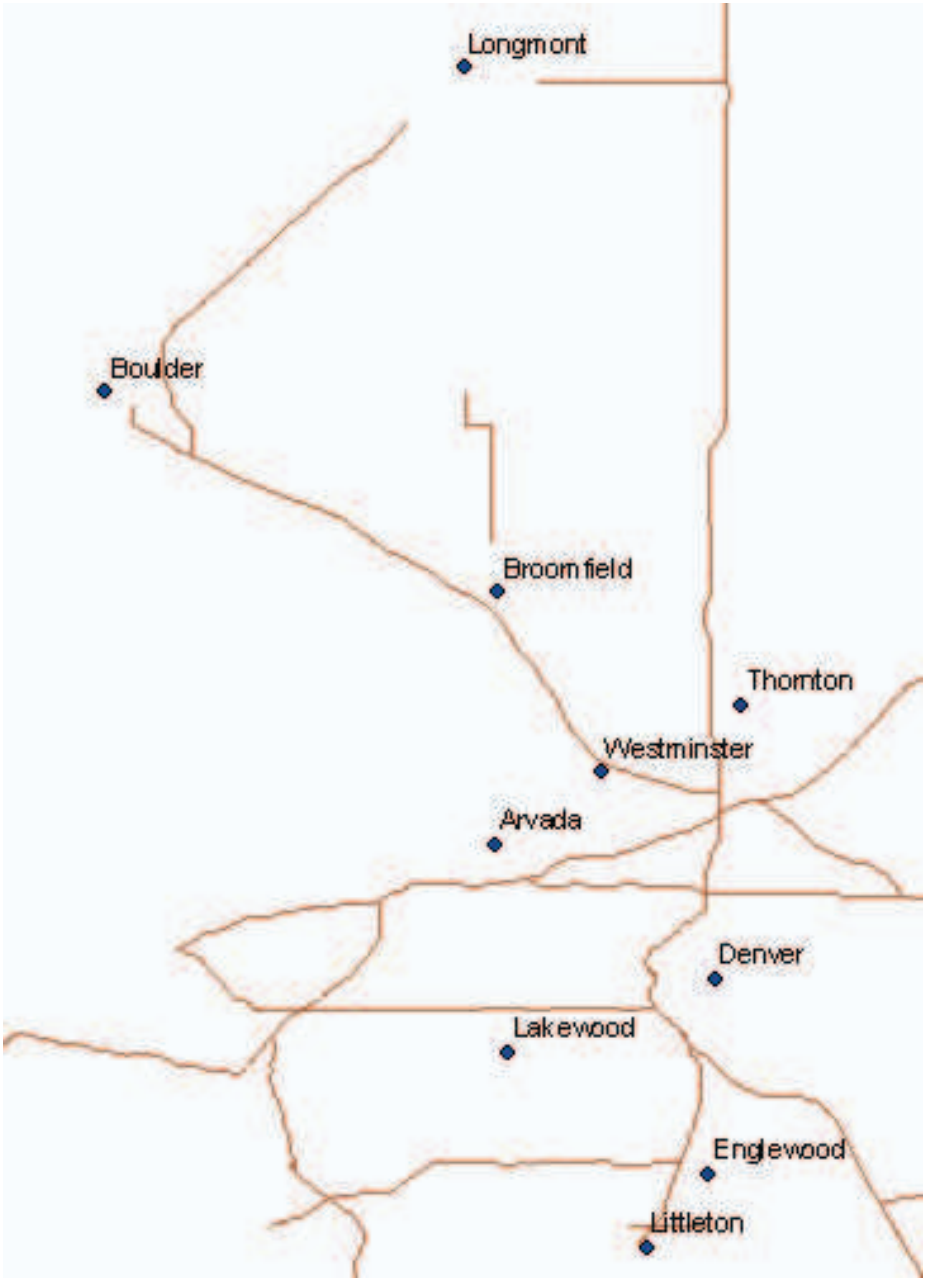

Figure 2.3: Vector lines (highways in Colorado)

You can use point data to visualize cities, restaurants, airports, and so on. In reality these entities are more accurately squares, rectangles, or oddly shaped polygons, but oftentimes the data you are trying to portray on the map is a simplifying assumption.

In some applications an accurate outline of a city is required. Other times a simple "X marks the spot" does the trick. Of course, both might be important depending on the zoom level of your map. Looking at a country- or state-level map, cities are probably best represented as dots. As you zoom in to the street level, the outline of the city becomes a better representation of the feature. (See Figure 2.2, on page 27.)

Lines are the next step up the vector food chain. At least two points are required to define a line. Each point is now called an *endpoint* or vertex. Lines can have as many vertices as necessary. The number of points can be *densified* or *generalized* (increased or decreased) depending on the level of detail required.

Line data is often used to represent static phenomena such as roads and rivers, but it can also be used as a data layer to help visualize dynamic data: driving routes of buses or delivery vehicles, driving directions between two addresses, flight paths, and so on. Notice how adding a basemap layer of roads helps ground the city points? (See Figure 2.3, on the preceding page.) It gives the cities context and a sense of place.

Our final stop in the grand tour of vector data types is the *polygon*, which is Greek for "many gons"—OK, OK: "many angles." To me, the defining characteristic of a polygon is the many *lines*, not the many *angles*. Then again, I'm not Greek, and I didn't invent geometry. (Geography and geometry—so close and yet so far apart….) Just as a line is made up of many points, a polygon is made up of many lines. Another way to differentiate between lines and polygons is that lines are open ended and polygons form closed shapes. Many GIS applications require the first point and the last point of a polygon to be identical, emphasizing that they must be closed shapes in order to be considered well-formed.

Polygons are most commonly used to represent boundaries: continents, countries, states, and the like. Adding county boundaries to our Colorado map completes the picture for now. (See Figure 2.4, on the next page.)

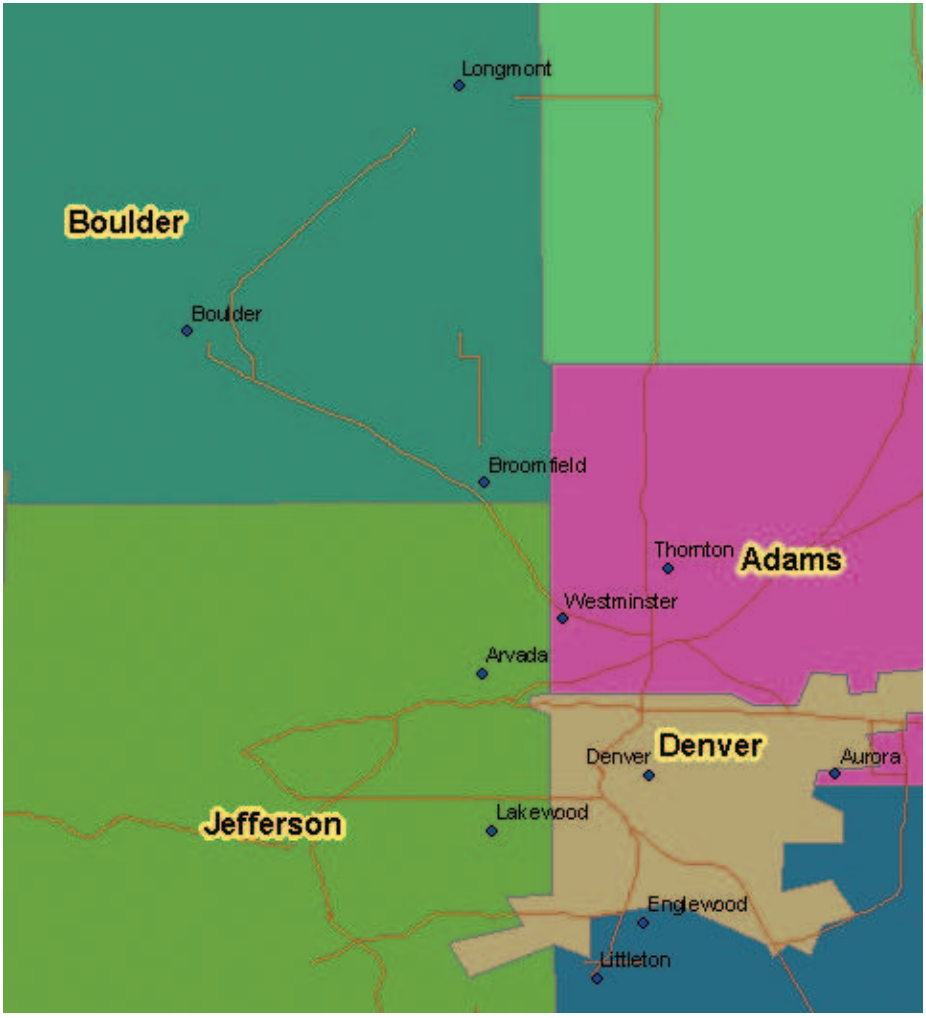

Figure 2.4: Vector polygons (counties in Colorado)

The Pragmatic Bookshelf

The Pragmatic Bookshelf features books written by developers for developers. The titles continue the well-known Pragmatic Programmer style, and continue to garner awards and rave reviews. As development gets more and more difficult, the Pragmatic Programmers will be there with more titles and products to help you stay on top of your game.

Visit Us Online

GIS for Web Developers Home Page
http://pragmaticprogrammer.com/titles/sdgis
Source code from this book, errata, and other resources. Come give us feedback, too!

Register for Updates
http://pragmaticprogrammer.com/updates
Be notified when updates and new books become available.

Join the Community
http://pragmaticprogrammer.com/community
Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

New and Noteworthy
http://pragmaticprogrammer.com/news
Check out the latest pragmatic developments in the news.

Buy the Book

If you liked this PDF, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: pragmaticprogrammer.com/titles/sdgis.

Contact Us

Phone Orders:	1-800-699-PROG (+1 919 847 3884)
Online Orders:	www.pragmaticprogrammer.com/catalog
Customer Service:	orders@pragmaticprogrammer.com
Non-English Versions:	translations@pragmaticprogrammer.com
Pragmatic Teaching:	academic@pragmaticprogrammer.com
Author Proposals:	proposals@pragmaticprogrammer.com