Table of Contents

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Chapter 3. Mashing Up Google Maps

3.1. Hacks 17–28: Introduction

In music, when you create a new song by taking the melody from one song and the lyrics from another, it is called a *mashup*. A lot of times things go poorly, but now and then the results are stunning. What happens when you take pieces from two web sites and mix them together? You get a Web 2.0 mashup.

The Web is moving from a collection of disconnected web sites to a ubiquitous computing platform. This new reality is often referred to as *Web 2.0*. In the beginning, we had static web sites with a few links between them. This evolved into dynamic content and data-driven sites. The next step has been using the web as a platform.

eBay is a useful site for buying and selling trinkets, trash, or treasure. In that role, it is what might be called "Web 1.5." But eBay is also a platform. There is a whole ecology that has built up around eBay that uses the platform in ways that were not initially intended by the programmers.

Amazon and Google Search have also become platforms. Amazon, eBay, and Google (not to mention Flickr, del.icio.us, and many more) have created public Application Programming Interfaces (APIs) that allow anyone to mix and match information from one site with information from another.

The missing piece in the ecology of open web APIs has been location. Nearly everything we do, on the Web or off, has a location. Everything we touch, write about, read, think, or work on has to happen somewhere. Everything has a geospatial component. Perhaps the geospatial component of some things is irrelevant. Do you really care where you were when you remembered to add dish soap to your shopping list?

Yes! We are the species that looks for patterns, and where we are, and where we have been, is one of the strongest sources of pattern in our lives! We are able to learn huge amounts from rooting through other people's trash, er, treasure—and we can learn similar amounts by analyzing the debris of our passing as recorded in position.

At the Where 2.0 conference in San Francisco in June of 2005, Tim O'Reilly explained his fascination with Paul Rademacher's Housing Maps site (http://www.housingmaps.com), described in "Find a Place to Live" **[Hack #23]**: "Google Maps with Craigslist is the first true Web 2.0 application, as neither of the sites was involved…. A developer put it together. Hackers are teaching the industry what to do."

Google Maps brought location into the world of open APIs, and the results have been stunning! The result brilliantly demonstrates the elegance of the Web 2.0 concept—a brave new world in which hackers can combine open standards and open APIs in novel ways to create new sites and services that fill a need or are sometimes just plain cool.

Mixing it up with data or code from multiple sites is the heart of the Web 2.0 experience. These mashups are leading the way to a Web that allows each of us to author our unique experiences of the Web, and to share those experiences with others.

In this chapter we explore just a few of the nearly countless Google Maps mashups that have come into existence in just a few months.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Hack 17. Map the News

See where it happened with BBC News and Google Maps.

Human beings have spent most of their time in small groups of 100-odd individuals, and our information-processing abilities came from those experiences, not from our current world. If we want to keep track of disturbances in Denmark, fog in Finland, elections in Istanbul, and war all over, we need tools to help us: assisted cognition.

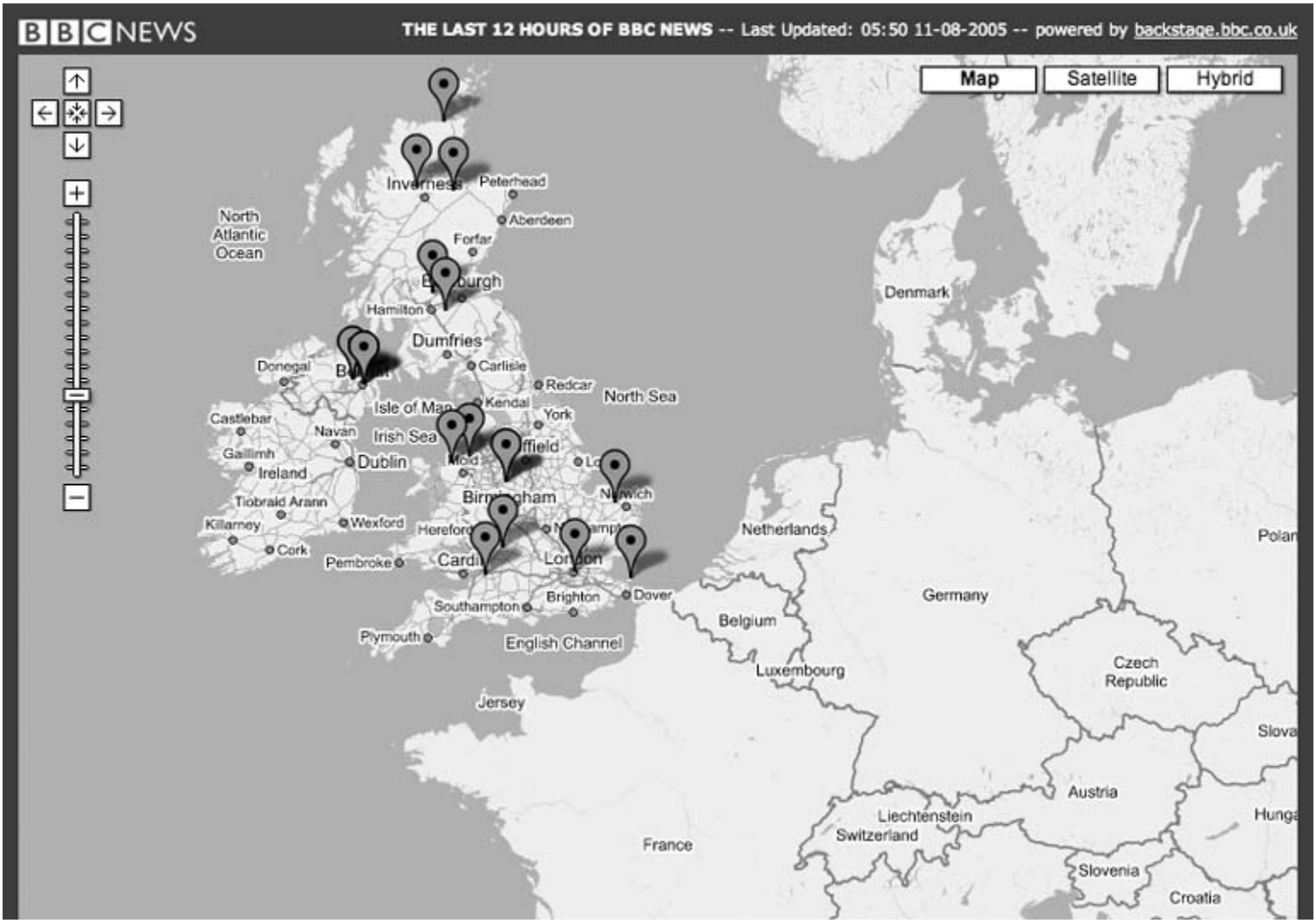
At http://boneill.ninjagrapefruit.com/wp-content/bbc/newmaps/ you can see the locations associated with the last 12 hours of BBC news, as shown in Figure 3-1. As with most cartographic efforts, there is the nearly inevitable, but still regrettable, focus on just one place—so the last 12 hours of BBC news will generally be more interesting if you prefer news of the British Isles.

Clicking on the markers brings up an information window like that shown in Figure 3-2. This includes the lead from the story, as well as date and time information and a link to the full story. As we can see, human interest and soft news can make an appearance!

This hack is possible because, well, the BBC rocks. They have decided that their responsibility to the public trust means they need to open their content to the public. See http://backstage.bbc.co.uk/ for data and ideas so that you can "build what *you* want using BBC content." This is the heart of the Web 2.0 concept—the idea that open APIs and open formats allow us to make more use of and draw richer connections between the vast amounts of information that are already out there on the Internet.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-1. Geolocations for the last 12 hours of BBC News



Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-2. Not all geolocated news is created equal

In 2003, the BBC announced plans to open their archives of radio and television programs for non-commercial use. Its intent is good, but sadly there are issues that must be worked out. Ben Hammersley wrote a stirring call to action for the Guardian at http://www.guardian.co.uk/online/story/0,,1522351,00.html.

The archive, Ben writes, "is a vault of the most important public culture of the past three generations. It is a gift for the future that is so far-sighted, and so much a good thing, that it is the duty of the BBC and, especially, the government to follow through."

What's more, "[d]igital technology not only makes the Creative Archive possible, but by doing so makes it a moral imperative." Mapping the news is an example of something we can do now, with just the textual content. Imagine the possibilities of geocoded audio and video!

As Ben's article concludes, "Now that we can, we must."

Hack 18. Examine Patterns of Criminal Activity

Augment your local government's crime reports with Google Maps.

Chicagocrime.org (http://www.chicagocrime.org/), one of the original Google Maps hacks, is a freely browsable database of crimes reported in Chicago. It combines data that was screen scraped from the Chicago Police Department's Web site (http://12.17.79.6/) with Google Maps, enabling many new ways for Chicago residents to keep tabs on their neighborhoods and explore crimes reported throughout their city. The site lets you browse crime reports in many ways: crime type, street name, date, police district/beat, ZIP Code, city ward, and generic "location" (e.g., bowling alley, bar, gas station). Figure 3-3 shows a rash of peeping toms around residential Chicago, while Figure 3-4 shows the locations for bogus check reports. There's also a City map page at http://www.chicagocrime.org/map/ that lets you combine search criteria.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-3. The most recent reports of illegal surveillance activity in Chicago

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-4. The most likely places to find bogus checks in Chicago

I developed Chicagocrime.org over a month's worth of nights and weekends in April 2005. Having gotten excited by the recently launched (at that time) Google Maps site, I spent a few evenings digging around Google's JavaScript and trying to embed custom maps into my own pages. After some hacking, I was able to display a custom map successfully. With my hacked-together map framework in place, it was just a matter of screen scraping the CPD's web site, geocoding each crime, and displaying the data. After Google released its official API at the end of June 2005, I updated Chicagocrime.org to make use of it.

3.3.1. Adding Ward and ZIP Code Boundaries

Aside from displaying a custom Google Map with relevant crime data on almost every page, Chicagocrime.org uses the Google Maps API in a couple of innovative ways. One way of navigating crime data is by police district. Because some residents of Chicago may not know their assigned districts, I created a "Find your beat and district" feature—http://www.chicagocrime.org/districts/—that helps people figure out which police beat and district they live in. It's simple to use: just pan and zoom the map to center it on a location, then click "Guess district."

It's simple under the hood, too. When a user clicks "Guess district," a bit of JavaScript calculates the center longitude and latitude of the current map view and sends that through a JavaScript `XMLHttpRequest` to a server-side script. The server code, written in Python and Django (http://www.djangoproject.com), uses a spatial query against PostGIS, a set of spatial extensions to PostgreSQL, to determine which district contains the given point. Finally, it passes the answer back to the site's JavaScript in your browser.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

You can find out more about PostGIS at http://postgis.refractions.net/. O'Reilly's *Web Mapping Illustrated* offers a good tutorial on PostGIS, as well.

Similarly, Chicagocrime.org lets you navigate crimes by city ward, and there's a "Find your ward" feature on the ward page: http://www.chicagocrime.org/wards/. For ward and ZIP Code pages, chicagocrime.org uses the Google Maps polyline-drawing API to draw the border for the given ward or ZIP Code on the map. I did this by obtaining the ward and ZIP Code boundaries in ESRI Shapefile format from the City of Chicago's GIS department at http://www.cityofchicago.org/gis/. I loaded the shapefiles into a PostgreSQL database and converted the data into longitude-latitude coordinates using the conversion functions in PostGIS. Finally, it was just a matter of feeding the points into the Google Maps polyline-drawing API, and *voila*: we have ward and ZIP Code borders.

3.3.2. See Also

- Rendering arbitrary GIS vector data on Google Maps is an interesting and still evolving subject. "How Big Is That, Exactly?" **[Hack #28]** involves rendering vector data from GIS sources on top of Google Maps.

—*Adrian Holovaty*

Hack 19. Map Local Weather Conditions

Find out whether there's weather where you are.

It's a well-known fact that everyone likes to talk about the weather, yet no one ever seems to do anything about it. Seriously, though, whenever two strangers meet and make small talk, it's inevitable that the recent meteorological conditions will make an appearance in the conversation. The state of the weather outside today, whatever it turns out to be, is something we all have in common—we're all obliged to endure it or enjoy it—at least, those of us that go out of doors.

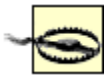
3.4.1. The Situation Outside Is…

Naturally, it didn't take long for someone to set up a Google Maps site devoted to tracking the weather—and we don't just mean any old weather, we mean detailed weather data, including temperature, wind speed and direction, relative humidity, and daily rainfall. Dave Schorr's Google Weather Maps site, at http://www.weatherbonk.com/, collects meteorological data aggregated by Weather Underground and Weatherbug from thousands of personal weather stations across the world, and then plots that information in a rich Google Maps interface on the Web.

Figure 3-5 shows the default view of *weatherbonk.com*, centered on San Francisco, California. If, as chance would most likely have it, you're not in San Francisco, you can start by entering your location in the search box at the top of the page. Valid location styles take the form `city,`

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

state, or *city, country*. U.S. ZIP Codes also work. In addition, you can overlay points from multiple locations by separating each query with a semicolon (;). For example, entering `33010;33446` will give you points along the southeast coast of Florida.



While a good number of international cities come up with results, you may need to be careful about what you type in here; for example, typing in "London, UK" returns nothing, while "London, England" returns what you would expect.

Also, at the time of this writing, Google only has detailed street maps for the U.S., U.K., Canada, and Japan. If you see broken image links in the background of the map, Google has not yet created street maps for your area. In this case, keep zooming out until the map appears, and/or click the Satellite button on the map to switch to the satellite view.

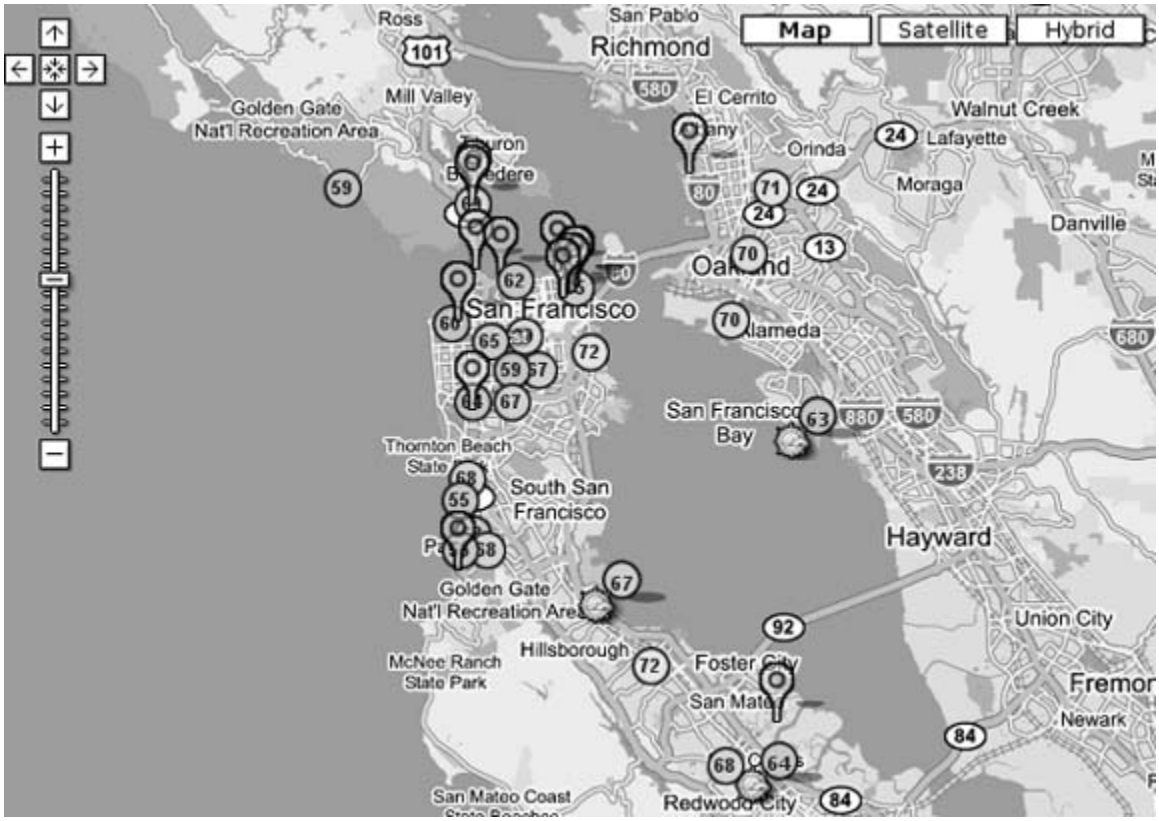
As you can see from Figure 3-6, the Weatherbonk.com site uses dynamically generated marker icons to convey a great deal of information at once. Each one of the weather station markers plotted on the map is color-coded according to the local temperature, ranging from blue (coldest) to red (hottest). Also, the temperature reading is displayed on the markers themselves, in either degrees Fahrenheit or Celsius, at your option.

If present, a spike extending from a marker points into the prevailing wind direction—i.e., toward the wind, not away from it—while the number of ticks shown on the spike indicates the observed wind speed, ranging from no ticks, representing a wind speed of under 4 mph, up to four ticks, indicating winds of 16 mph or more. Additionally, a marker may contain an icon illustrating other current conditions, such as sunshine, overcast, or rain. A key to these markers is shown at the bottom right corner of the page.

Clicking on any of the markers opens an info window with the details for that weather station, as shown in Figure 3-6. If the station is affiliated with Weather Underground, temperature, wind speed and direction, and relative humidity are shown. Below these readings, you'll see a graph illustrating the 24-hour history for both temperature and dew point. The title at the top of the info window is linked to the homepage of the maintainer of that particular station.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-5. The default view, showing the current conditions in San Francisco



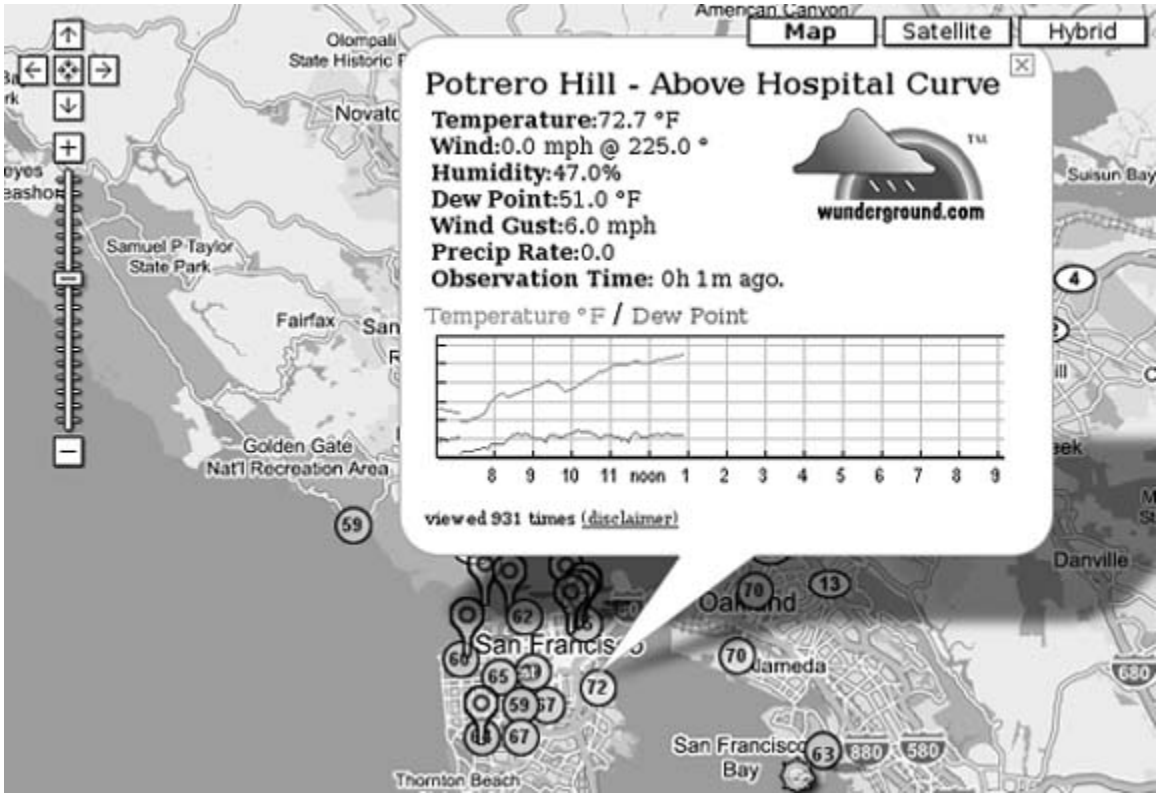
The *dew point*, in case you're wondering, is the temperature at which the water vapor in the air begins to condense into liquid water. Relative humidity, which is what's shown in the info windows, is calculated from a combination of ambient temperature and dew point. Frosty drinks on a hot day often lower the temperature of the air immediately around them to a level below the dew point, which is why a layer of condensation forms on the outside of the glass.

By contrast, a Weatherbug-affiliated station shows an info window with a daily rainfall figure, in place of the historical graph. Some Weatherbug stations also have webcams, which are shown in the info window, if present. Weatherbug stations aren't shown on the map by default—you need to select the Weatherbug checkbox at the top of the page and then click the Update Map button at the top right of the map.

Other, not necessarily weather related, webcams can be shown on the map as well. Select the Webcams checkbox at the top of the page, if it isn't already selected, and then click Update Map. These locations, which are often educational institutions, are identified on the map by transparent markers. As you'd expect, clicking on a webcam icon opens an info window showing the current image from that location.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-6. Weather station details are shown in an info window



The info windows shown on the map can often be quite large, what with all the weather information and webcam images they include. One result of this is that the X button used to close the info window can occasionally wind up obscured by other things on the map. Fear not—you can always close an open info window by clicking on the associated marker a second time.

3.4.2. More Than Just the Weather

The Weatherbonk.com site supports some other interesting features. On the right side of the map, you can select various sources of radar data, to depict cloud cover on the map. Additionally, under the Google Earth section of the page, you can access different sources of cloud radar imagery in KML format, for use with Google Earth.

From the standpoint of Google Maps, however, the most interesting additional features are the three map control buttons at the top right corner of the map, immediately below the map type control. These buttons are labeled Zoom Box, Clear Points, and Update Map. The Clear Points button wipes all the markers off the map, while the Update Map button loads new data from the server, which can be handy if you zoom or pan the map to view a different area.

The Zoom Box feature is particularly nifty and bears a bit of explanation. If you hold down Shift-Z, and then left-click and drag your mouse across the map, a red box appears and follows your mouse drag. Releasing the left mouse button causes the map to zoom into the area within the red box, which can be quite handy for drilling down to a particular local region. Clicking Update Map after you've used the zoom box feature can sometimes reveal weather stations that weren't shown on the larger area map, as the site tries to avoid crowding the map with too

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

many points at any given zoom level. Also, you don't actually need to click the Zoom Box button before using this feature—the button itself doesn't do anything useful, beyond displaying helpful instructions in an alert window.

3.4.3. Microclimates and Distributed Weather Reporting

Like so many other things, the Internet has made it possible for weather reporting to be distributed among many people on a ground-up basis, rather than centralized in a top-down fashion, as it traditionally has been. One thing that this decentralization makes possible is witnessing for yourself the striking variety of weather conditions in an area with lots of microclimates, like the San Francisco Bay Area. Try it mid-to late-afternoon Pacific Time (GMT -7 during the Daylight Savings Time, GMT -8 otherwise), when the fog usually starts to creep in off the ocean, thus cooling some areas, while other places are still clear and warm. To augment this view, try adding the Bay Area fog overlay from the Overlays drop-down on the right side of the page. The differences across an area even as small as San Francisco can sometimes really be quite striking.

3.4.4. See Also

- The Weather Underground site at http://wunderground.com/ has RSS feeds for weather reports in various metropolitan areas.
- http://api.weatherbug.com/ is the home for Weatherbug's data access API.
- METAR is a very terse text format used around the world for reporting weather data. You can find out more about METAR, and get live METAR feeds from the U.S. National Weather Service at http://weather.noaa.gov/weather/metar.shtml.
- http://anti-mega.com/weather/ offers a series of (non-georeferenced) RSSbased weather feeds, using data scraped from *worldweather.org*.

—written with assistance from Dave Schorr

Hack 20. Track Official Storm Reporting

Follow the path of the latest hurricane on a Google Map.

Google Map hacking started for me in early March 2005, when I became aware of some of the great hacks that were already being created. My interest was particularly piqued after seeing the beta release of Adrian Holovaty's Chicagocrime.org web site, as described in "Examine Patterns of Criminal Activity" **[Hack #18]**. I immediately realized the myriad other applications for this new mapping technology. To be more precise, I determined that it would be beneficial to develop a storm-reporting mapping site, which you can visit today at http://www.stormreportmap.com/, as shown in Figure 3-7.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-7. The weather on Google Maps

You can click on a marker to see more details. For example, Figure 3-8 shows a report of hail in Sioux, Nebraska.

Another interesting feature is the hurricane tracking maps. As shown in Figure 3-9 you can see the tracks of tropical storms.

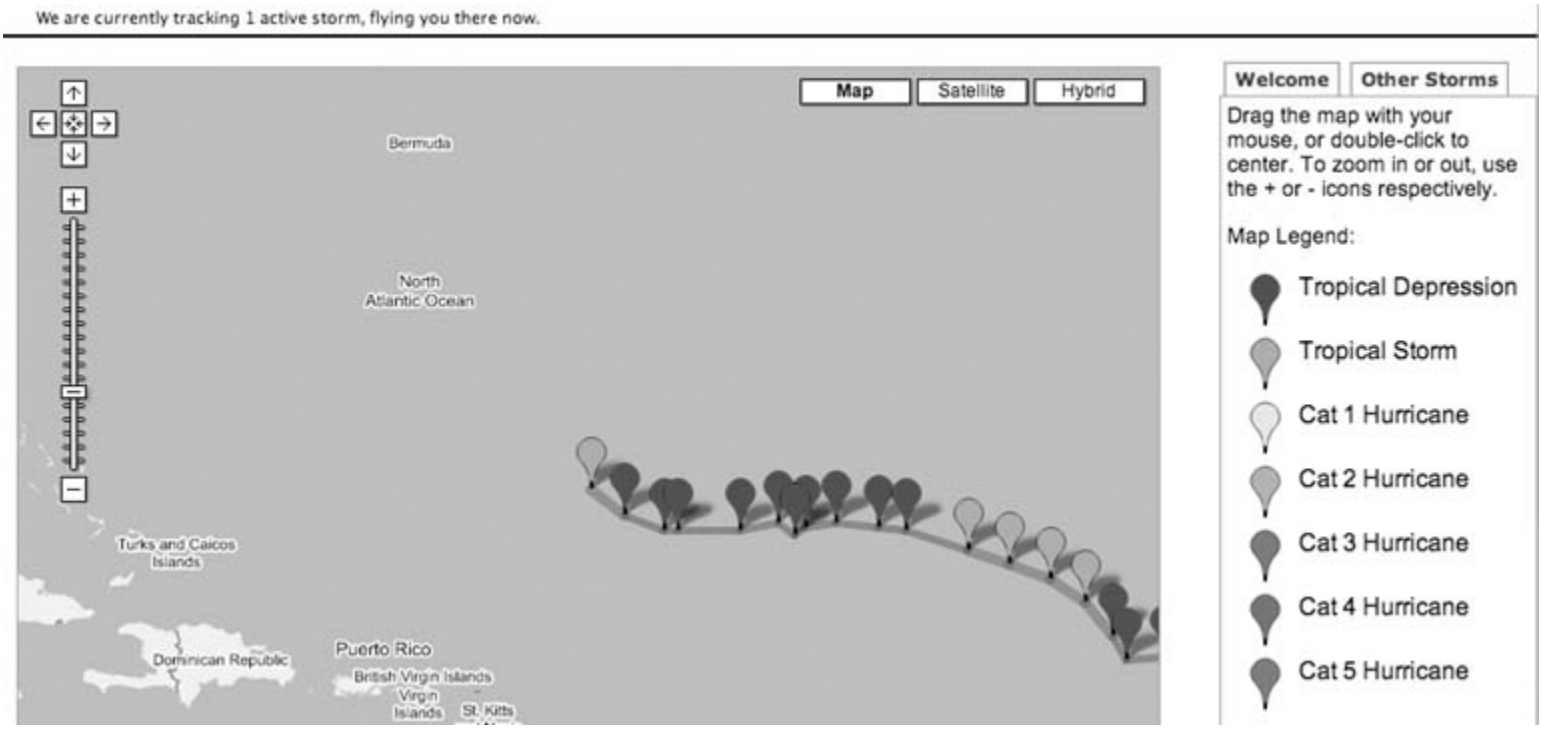
Clicking on the marker icons gives you more information about the storm at that point. In Figure 3-10, we see that Irene has been bouncing between a tropical storm and a tropical depression.

The data source for the web site is the National Oceanic & Atmospheric Administration's (NOAA) National Weather Service (NWS) Storm Prediction Center (SPC), which can be found online at http://spc.noaa.gov/. One of the top products that the SPC maintains is their Storm Report tracker at http://spc.noaa.gov/climo/. This product takes reports from trained weather spotters, emergency and first responders, and local residents, and then maps them to an unattractive, static page. Moreover, the reports also don't give much in the way of comprehensible locations, aside from latitude/longitude coordinates, and the county and state where the storm was reported. Additionally, since each report is submitted into the database as a separate event, there are times when several tornado reports received are actually all from the same tornado. Due to these shortcomings in the original product, and because of its general popularity, I felt that my project might enhance the product's basic functionality.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

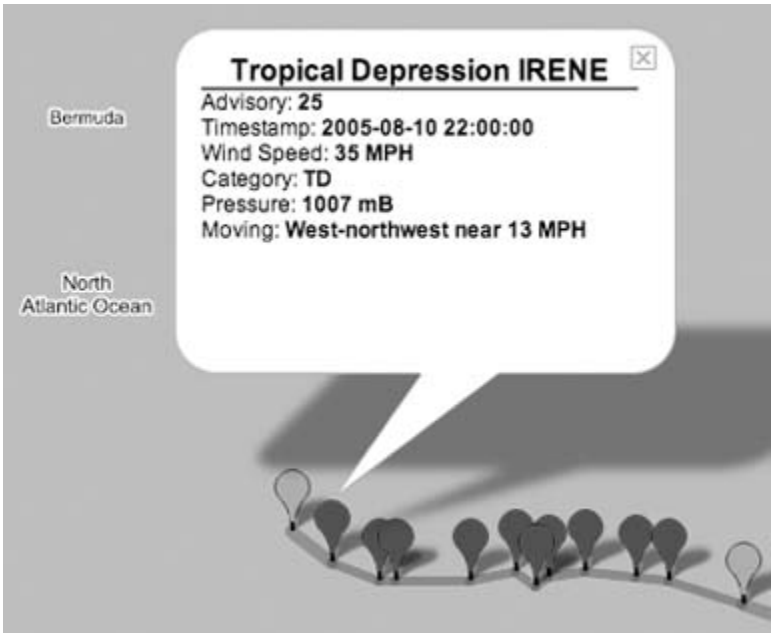
Figure 3-8. Hail in Nebraska

Figure 3-9. Hurricane-tracking maps



Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-10. Good night, Irene!



3.5.1. Getting the Data

The SPC began collecting storm report data in mid-1999. Unfortunately, they did not begin putting the data into a web-friendly format until early 2004. Therefore, without parsing through three different versions of web sites, the project will only show storm report data from early 2004 on.

The data is obtained through a *Comma Separated Values* (CSV) file that is posted along with each update. This very friendly format allows me to parse through the file, convert it into an array, format it to my liking, and push the contents out as an XML file. If you are getting your data from a third-party site through an XML or RSS feed, it's important to realize that bandwidth often costs money.

Before beginning this project, I realized that obtaining the data I want and in the timeframe I wanted it might present a problem for the remote SPC web site. Our assumptions using information obtained from the site were the following:

- The current day's map is updated every 15 minutes.
- Yesterday's map is updated in 3-hour increments.
- Once a map is more than 2 days old, it is no longer updated.

I couldn't develop a project that would simply leech off the SPC web site's data for every visit to my site—it's disrespectful, lazy, and, most of all *slow*. I determined that the best model would be to use a back-end database to store all the requested data and load the most current data using a series of timestamps.

Here's how the site handles the data refresh when a user visits the site:

1. The site loads the report data for the request.
2. A timestamp indicating when the data was last updated for each day is stored in the database.
3. Since each page load requires that the database load the report data, the site also checks to see when the data was last updated.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

4. The site updates today's data, if more than 60 minutes have elapsed between the stored timestamp and the current timestamp.
5. The site updates yesterday's data, if more than 24 hours have elapsed between the stored timestamp and the current timestamp.
6. For historical data, no timestamp checking is done, because the most current data is already loaded, based on previous assumptions.
7. The page displays the data to the user.

This model drastically reduces the traffic and bandwidth sent to the remote SPC web site, and I would recommend using it for any site that uses regularly updated third-party data from remote web sites.

3.5.2. The Hack

When the Google Maps API was finally released, I decided to check out its functionality firsthand. Since the original site was weather related and hurricane season was just beginning to get in the swing of things, a hurricane tracker using the new API was in order.

The Hurricane Track Map uses the Google Maps API to develop a Google Map without jumping through all the hoops that the pre-API XSLT methods required. The catch is that there is quite a bit more JavaScript development that you will need to include. Google has gone to great lengths to try and give enough documentation for a novice developer to get started, but even some veteran programmers have trouble getting it to work correctly.

This code snippet shows just how you can use the `GXmlHttp` class from the Google Maps API to load XML data from a file on your server. As usual, you cannot retrieve remote files from other sites using AJAX; only files within your domain or host can be loaded, which is another good reason to cache the data in your own database first.

```
var request = GXmlHttp.create();
request.open("GET", "data.xml", true);
request.onreadystatechange = function() {
  if (request.readyState == 4) {
    markers = [];
    points = [];
    infoHtmls = [];
    categorypoly = [];
    var xmlDoc = request.responseXML;
    var markers = xmlDoc.documentElement.getElementsByTagName("marker");

// Loop through the XML document and grab the data contained
// with the tags. Store that data into an array.

    for (var i = 0; i < markers.length; i++) {
        points[i] = new GPoint(parseFloat(markers[i].getAttribute("lng")),
                    parseFloat(markers[i].getAttribute("lat")));
        name = markers[i].getAttribute("name");
        type = markers[i].getAttribute("type");
        advisory = markers[i].getAttribute("advisory");
        timestamp = markers[i].getAttribute("timestamp");
        windspeed = markers[i].getAttribute("windspeed");
        pressure = markers[i].getAttribute("pressure");
        moving = markers[i].getAttribute("moving");
        point = points[i];
// Append a hurricane category rating on it based on the
// Wind Speed.
        if (type == "Tropical Depression") {category = "TD";}
        else if (type == "Tropical Storm") {category = "TS";}

        else { // Its a hurricane
            if (windspeed >= 74 && windspeed <= 95) {category = "1";}
            else if (windspeed >= 96 && windspeed <= 110) {category = "2";}
```


Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

```
            else if (windspeed >= 111 && windspeed <= 130) {category = "3";}
            else if (windspeed >= 131 && windspeed <= 155) {category = "4";}
            else {category = "5";}
        }
        categorypoly[i] = category;

// Call to the createMarker function to create the marker
        var marker = createMarker(point, name, type, advisory, timestamp,
            windspeed, category, pressure, moving);

// Overlay the markers on the map
        map.addOverlay(marker);
}
```

The `createMarker( )` function is called in order to build the info window for that particular marker. Also, this function assigns a custom icon to the marker, depending on whether the storm is classified as a tropical storm, a tropical depression, or a category 1 through 5 hurricane.

Next, our code adds a polyline to go along with the markers. This is valuable, because hurricane tracks are very unpredictable, and often make loops and turns before they make landfall or get pushed back out to sea.

```
var pointset = [];
// Loop through the array of points created above. Each point, based
// upon the category it received will receive a line segment color as
// well as the width.
    for (q=0; q < points.length; q++) {
        pointset.push(points[q]);
        if (categorypoly[q] == "TD") {color = "#660099";size = 5;}
        else if (categorypoly[q] == "TS") {color = "#333399";size = 7;}
        else if (categorypoly[q] == "1") {color = "#33FFFF";size = 9;}
        else if (categorypoly[q] == "2") {color = "#33FF66";size = 11;}
        else if (categorypoly[q] == "3") {color = "#FFFF66";size = 13;}
        else if (categorypoly[q] == "4") {color = "#FF9933";size = 15;}
        else {color = "#FF3333";size = 17;}
// Add point to the GPolyline so the map can draw it.
        map.addOverlay(new GPolyline(pointset, color, size));
        pointset = [];
        pointset.push(points[q]);
}
```

3.5.3. See Also

- See http://code.stormreportmap.com/ for more source code and references.
- All source code is released under the GNU General Public License. You may use and publish as you wish without any copyright notice being retained or transferred. If you wish to contact me, I can be reached at webmaster@stormreportmap.com.

—Anthony Petitoa

Hack 21. Track the International Space Station

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

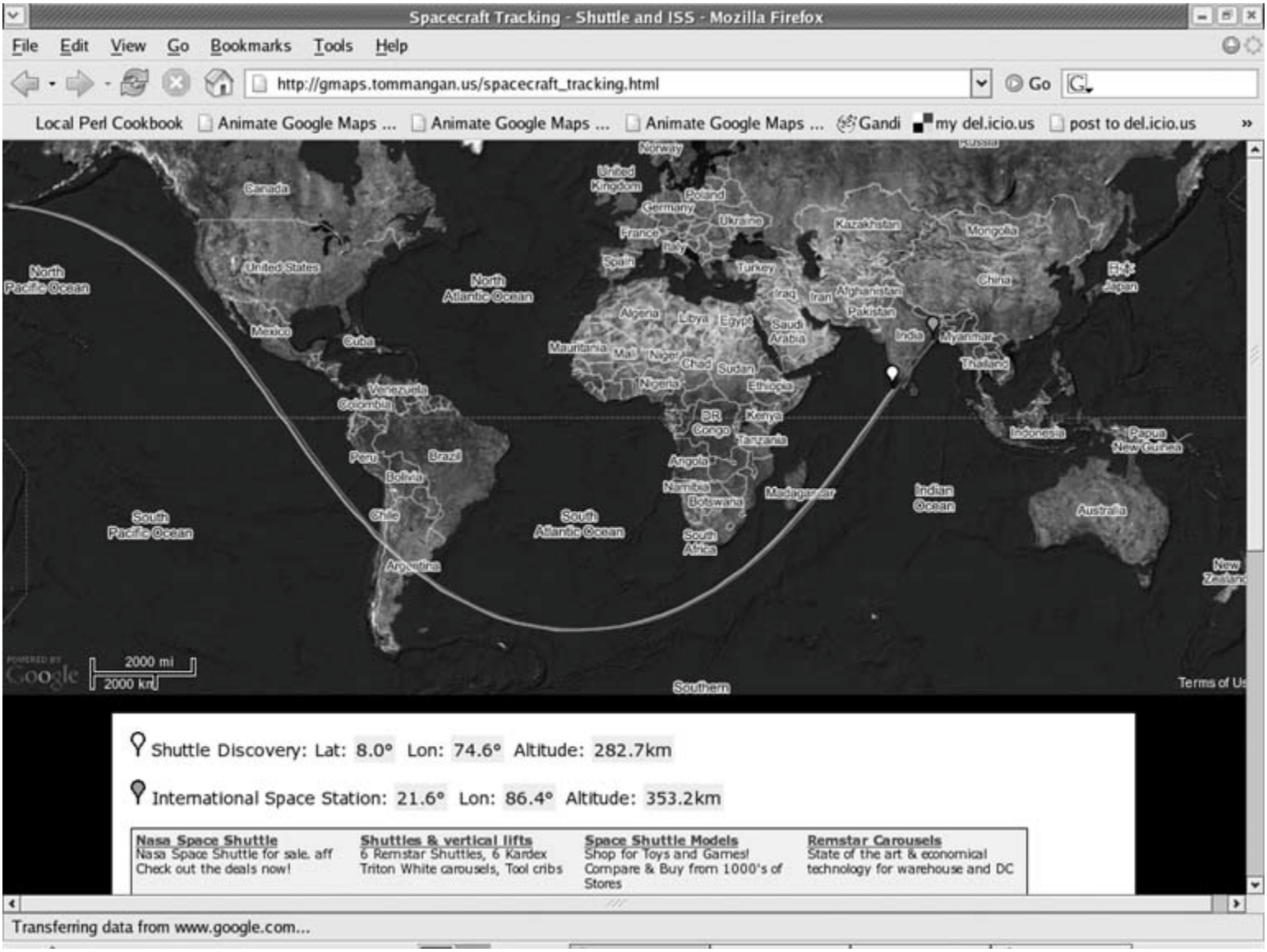
Track the International Space Station and the Space Shuttle in near-real time.

You can track anything on Google Maps—all you need is a source of data. Tom Mangan created his own site to track the Space Shuttle and International Space Station (ISS) at http://gmaps.tommangan.us/spacecraft_tracking.html.

The site tracks the location of the ISS and, when it is in orbit, the Space Shuttle, as shown in Figure 3-11. The excitement can come from being able to spot the Shuttle and ISS from the ground, and in watching on the map as they rendevous. When you first load the page you should get one or two markers, depending on whether the Shuttle is in orbit. If you leave the page open the markers leave a trail of where the ISS and Shuttle have been. Over the course of about 90 minutes (actually, 91.55 minutes for the ISS, according to the Wikipedia page at http://en.wikipedia.org/wiki/International_Space_Station), the characteristic sine wave shape of low Earth orbits appears. No, objects in space don't bounce around like tennis balls; this represents the effects of representing a three dimensional orbit onto a flat map. The first rule of cartography is that you always distort something!

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-11. Shuttle and ISS tracking, with an amusing note



Google's AdSense ads are often the funniest part of a page. Here we have a link to "NASA Space Shuttle for sale. Check out the deals now!" I don't think I want to buy an affordable spacecraft on eBay! Maybe in a few years.

3.6.1. How It Works

Most of the work happens in the JavaScript in the site's `autoUpdate( )` function. `autoUpdate( )` is called as soon as the page loads.

```
<body style="background-color: #000; text-align: center;"
        onload="autoUpdate();">
```


Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

The `autoUpdate` function uses the `GXmlHttp` class from Google Maps to fetch the *iss.js* file from the server. You can take a look at the file at http://gmaps.tommangan.us/iss.js.

```
-48.6;-11.7;369.8;-51.6;-31.9;291.1
```

This shows the latitude and longitude of the ISS first, and then that of the shuttle. The code reads the file, then splits the results into the array `coords` with the `split` method. The rest of the method is housekeeping to do a sanity check on the returned result, then adds the points to the current arrays of results for the Shuttle and ISS.

```
function autoUpdate() {
    var request = GXmlHttp.create();
    request.open("GET", "iss.js", true);
    request.onreadystatechange = function() {
      if (request.readyState == 4) {
        var response = request.responseText;
        coords = response.split(';');
        var valid = (coords[0]&&coords[1]&&coords[3]&&coords[4]
             &&coords[0]>-90&&coords[0]<90&&coords[1]>-180
             &&coords[1]<180&&coords[3]>-90&&coords[3]<90
            &&coords[4]>-180&&coords[4]<180);

        if (valid) {
          map.clearOverlays();
          var sPoint = new GPoint (coords[4],coords[3]);
          var sMarker = new GMarker (sPoint,pinWhite);
          if (coords[4] < -174) { sTrack=[]; }
          sTrack.push (sPoint);
          map.addOverlay(sMarker);

          var issPoint = new GPoint (coords[1],coords[0]);
          var issMarker = new GMarker (issPoint,pinRed);
          if (coords[1] < -174) { issTrack=[]; }
          issTrack.push (issPoint);
          map.addOverlay(issMarker);

          refreshCoords();
          drawTrack();
        }
      }
    }
    request.send(null);
    window.setTimeout ('autoUpdate()', 60000);
}
```

The final trick is the last line of `autoUpdate()`:

```
window.setTimeout ('autoUpdate()', 60000);
```

This tells the window to timeout after 60,000 milliseconds, i.e., one minute. When it times out it calls `autoUpdate()` and starts the cycle again.

There is a script running on the server that queries the NASA site for the current position of the Shuttle and ISS, and then writes that out the *iss.js* file. You can use these same techniques to do dynamic updating of your own maps.

If you are the sort of person who goes in for space shuttle tracking, you'll like a couple of Tom's other projects at http://gmaps.tommangan.us/. He has a map that shows the current known locations of all of the SR 71 Blackbirds and aerial photos of Area 51. Both projects use his TPhoto extension to the Google Maps API, which lets you embed your own images within Google Maps **[Hack #55]**.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

3.6.2. See Also

- NASA's Satellite Tracking Page, which includes applications to track many different satellites, although not with Google Maps: http://science.nasa.gov/realtime/.

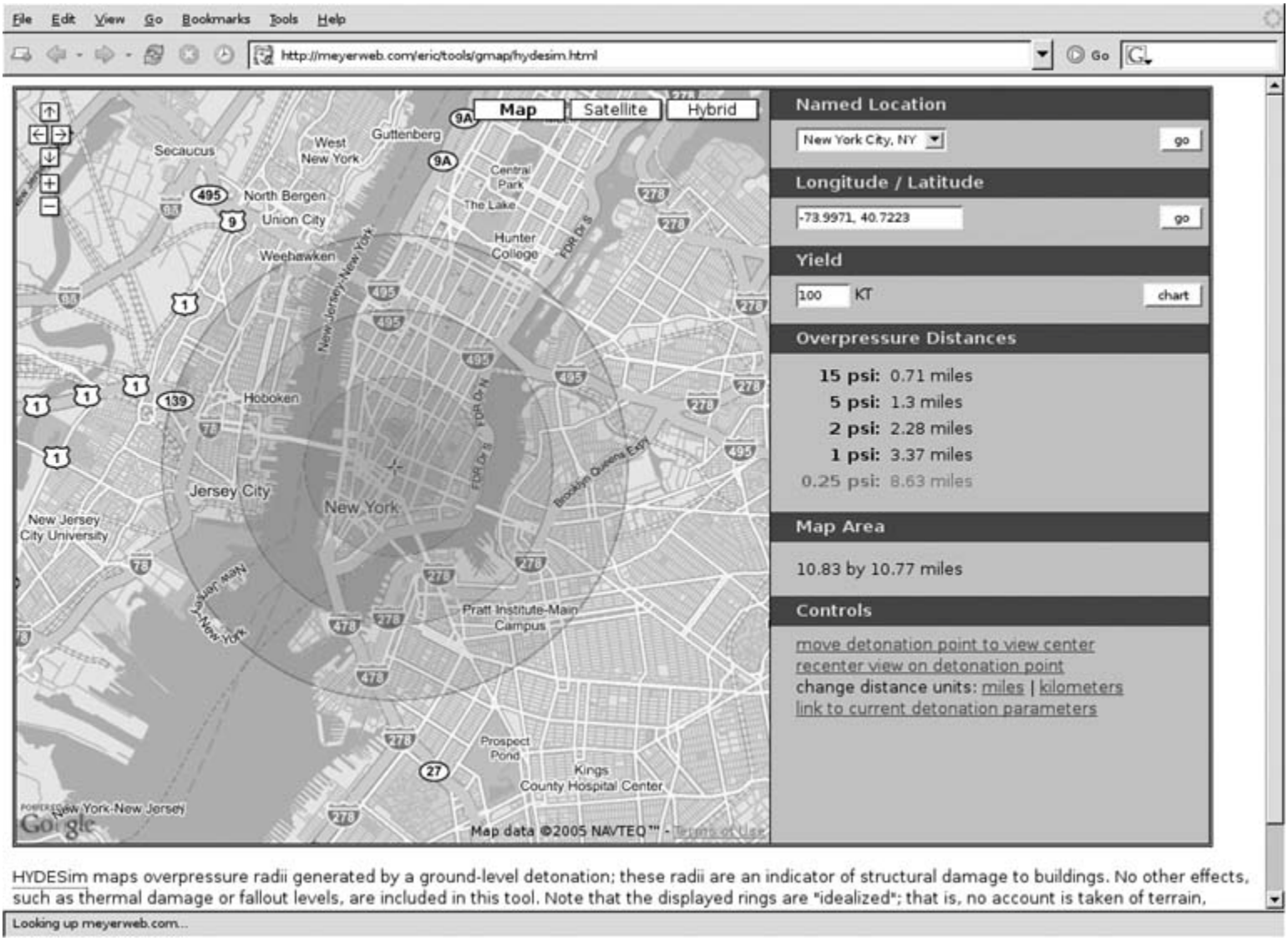
Hack 22. Witness the Effects of a Nuclear Explosion

Sometimes a map can reveal truths we'd rather not know.

For over 50 years, the human race has lived under the shadow of the threat of nuclear war. Eric A. Meyer's HYDESim (High-Yield Detonation Effects Simulator) web site, which uses Google Maps in a somewhat novel way to illustrate the blast effects of a nuclear detonation. You can see the results for yourself at http://meyerweb.com/eric/tools/gmap/hydesim.html. Figure 3-12 shows the default view, which illustrates the effect of a 150 kiloton explosion at ground level in downtown Manhattan.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-12. Depiction of the blast wave of a 150-kT nuclear explosion in downtown Manhattan



The blast effect of a nuclear explosion is usually reckoned in terms of *overpressure*, which is a measure of how much force is exerted on people and buildings at a given distance away from ground zero. On the map, this is shown as four concentric rings of decreasing intensity, which spread outward from the hypothetical explosion site, representing overpressures of 15 psi, 5 psi, 2psi, and 1 psi, respectively. The display on the left shows the blast radius for .25 psi overpressure as well, although this ring is not shown on the map.

What exactly do these figures mean, though? The descriptions shown in Table 3-1 are taken from section 5 of the Nuclear Weapons FAQ at http://nuclearweaponarchive.org/Nwfaq/Nfaq5.html.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Table 3-1. Destructive effects of atmospheric overpressure

Overpressure	Structural effects	Human injuries
1 psi	Window glass shatters.	Light injuries from fragments occur.
3 psi	Residential structures collapse.	Serious injuries are common, fatalities may occur.
5 psi	Most buildings collapse.	Injuries are universal, fatalities are widespread.
10 psi	Reinforced concrete buildings are severely damaged or demolished.	Most people are killed.
20 psi	Heavily built concrete buildings are severely damaged or demolished.	Fatalities approach 100%.

Although the results are simplified, they sure don't look pretty—at that location and yield, such a nuclear explosion would literally wreck all of downtown Manhattan. What's worse, this map doesn't take the effects of heat or radiation into account. (On the other hand, this map doesn't take the attenuating effects of terrain and weather into account, either.) Although the destructive effects of nuclear weapons are hardly news to anyone, it is still kind of morbidly interesting to be able to see them on a map. Additionally, the site allows you to see the effects on certain other U.S. cities listed in the drop-down box at the upper right, and, if you happen to know the latitude and longitude of a place that particularly interests you, you can enter them into the coordinates box below that.

If you live in the States and don't happen to know the coordinates of, say, your hometown, you can look up a specific address on the Geocoder.US web site at http://geocoder.us/.

Finally, you can adjust the yield of the hypothetical explosion, which is measured in kilotons of TNT. By experimenting, we can see that a 1 megaton nuclear explosion over the Brooklyn Bridge would destroy most of Manhattan, Queens, and Brooklyn. Fortunately, these high-yield nukes have been phased out of most of the world's military stockpiles. On the other hand, we can see that even a relatively "small" detonation on the order of 5 kilotons could wreak utter mayhem in significant parts of the city.

3.7.1. The Code

From a technical standpoint, what makes this hack interesting is this bit of JavaScript code, which you can find for yourself by viewing the source of the aforementioned web page:

```
var base = new GIcon();
base.image = "radii.png";
base.shadow = 't.png';
base.shadowSize = new GSize(1, 1);

var GZ = new GIcon(base);
GZ.image = "crosshair.png";
GZ.iconSize = new GSize(13, 13);
GZ.iconAnchor = new GPoint(6, 6);
GZ.infoWindowAnchor = new GPoint(6, 6);
```


Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

```
var p15 = new GIcon(base);
p15.iconSize = new GSize(det.radius.p15*2/mpp, det.radius.p15*2/mpp);
p15.iconAnchor = new GPoint(det.radius.p15/mpp, det.radius.p15/mpp);
p15.infoWindowAnchor = new GPoint(det.radius.p15/mpp,
    det.radius.p15/mpp);

var p5 = new GIcon(base);
p5.iconSize = new GSize(det.radius.p5*2/mpp, det.radius.p5*2/mpp);
p5.iconAnchor = new GPoint(det.radius.p5/mpp, det.radius.p5/mpp);
p5.infoWindowAnchor = new GPoint(det.radius.p5/mpp, det.radius.p5/mpp);
```

This code, which you'll find in the `buildOverlays( )` function, uses the standard GIcon marker object from the Google Maps API to render the blast radius rings using the same semi-transparent *radii.png* image. Each one is sized separately, according to the blast radii calculated from the detonation yield in the `Detonation( )` constructor (not shown here), and the `mpp` variable, which stores the map scale at the current zoom level. Additionally, a crosshair icon is created to represent ground zero itself.

The upshot is that when the map is loaded, or whenever the detonation location is moved, the individual blast radius markers are stacked on top of one another at the same location on the map. The semi-transparent circles then give the visual impression of blast intensity decreasing, as it moves away from ground zero. The result, which perfectly conveys the desired information, is a very clever use of the Google Maps marker icons, which are usually used to represent grocery stores or yard sales! The same technique could be used to represent any kind of data via Google Maps that involves concentric radii of decreasing intensity. One immediately thinks of volcanic explosions or earthquake damage as candidates for this kind of interface, but there are probably less destructive topics that could be illustrated the same way.

Certainly, the prospect of nuclear war, or even of an isolated nuclear explosion in a populated area, is a terrifying one. So far, humanity has managed to show considerable restraint in its application of nuclear weapons, but an estimated 20,000 nuclear warheads still exist in the world's military arsenals. Ridding ourselves of this menace remains one of the most important outstanding issues in international politics. Hopefully, access to the kind of information offered by the HYDESim site will bring home to people how tragic the possibilities are, and just how imperative it is that the menace never becomes a reality.

3.7.2. See Also

- The Nuclear Weapons FAQ (http://nuclearweaponarchive.org/Nwfaq/Nfaq0.html) answers a lot of common questions about nuclear weapons.
- The Atomic Archive's New York example (http://www.atomicarchive.com/Example/Example1.shtml) illustrates a situation much like the one shown in Figure 3-12, only in much more detail.
- Wikipedia's List of Nuclear Accidents at http://en.wikipedia.org/wiki/List_of_nuclear_accidents makes for an interesting read and illustrates just how close we've come over the years to serious accidental nuclear explosions.

Hack 23. Find a Place to Live

Why slog through endless listings of apartments that all look the same, when you can pick and choose based on where you actually want to live?

Finding a place to live, particularly in large cities, can be a huge pain. Locating a place at a price you can afford is enough of a challenge, but, even when you can locate such places, they're often not situated where you actually want to live. Real estate and rental listing sites like Craigslist.org go a long way towards easing this pain by offering searches against listings based on keywords and price brackets, but often

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

even the search results themselves are still daunting. Where the heck are all these places? How close are each of these apartments to the grocery store? To public transit? To my friends' houses? The listings often include neighborhood names, or even street addresses or nearby intersections, but if you're not particularly familiar with the area, these names might not mean anything to you.

Paul Rademacher's HousingMaps site offers an inventive start on a solution to this problem: take one part Craigslist.org real estate listings, one part Google Maps API, and stir!

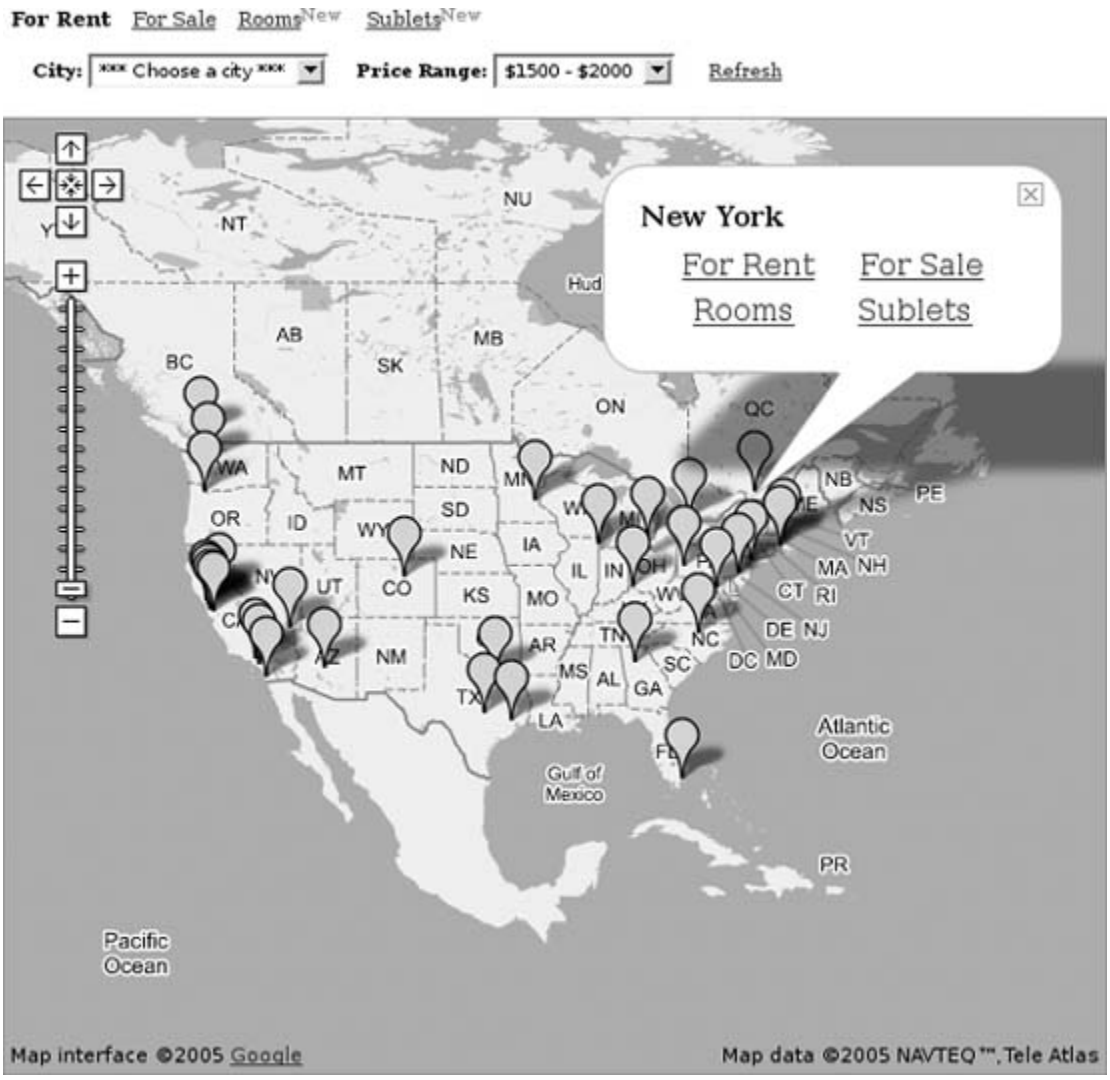
3.8.1. What to Do?

The front page of HousingMaps, which you can find at http://housingmaps.com/, shows a Google Maps view of the United States and Canada, as seen in Figure 3-13. The green icons on the map mark many of the major urban areas served by Craigslist, and clicking one opens a call-out, from which you can select real estate for sale, rentals, rooms for rent, and sublets or temporary housing for that city.

If you find this view a bit confusing—or find the icons hard to click, as in the northeastern United States, where the icons are bunched a bit close together—then you can select the metropolitan area of your choice from the drop-down box at the top left, and then use the links above to narrow your search. (Alternatively, you can recenter with a mouse drag or double-click, and then zoom in! This is Google Maps, after all.) Finally, another drop-down box at the top allows you to indicate price ranges of properties to show on the map.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-13. Why live anywhere when you could live, er, anywhere?



After selecting a type of listing for a given city, you're taken to a view of the available properties of that type in that city. Since I've been looking for a cheap short-term sublet in north Brooklyn, I select *sublets* in New York City, and then use the zoom and pan controls to focus in on that area. Yellow icons identify listings with photos, while red icons identify those without. I started by clicking one of the yellow icons on the map, as shown in Figure 3-14.A Google Maps info window pops up, showing the details of the property, including contact details and any photos.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-14. A search for rentals in north Brooklyn using HousingMaps

A list of identifiable properties for that city and type is shown on the right side, sorted by price. Clicking on one of the icons recenters the map on that property and opens its call-out, while clicking on the adjacent link takes you directly to that entry in Craigslist. Additionally, you can apply the same filters to your search that Craigslist offers, such as keywords, pets permitted, and the presence of photos in a listing. Finally, there's a "permalink" option that allows you to bookmark a particular search for future use once you've zeroed in on the part of town and the search filters you're interested in. Does this beat paging through mind-numbing lists of rentals that all look the same or what?

3.8.2. How It Works

How does this miracle of modern technology function? Part of the answer lies in the RSS feeds offered by Craigslist for each category they provide. The Brooklyn sublets, for example, are listed at http://newyork.craigslist.org/brk/sub/. At the bottom of the page, there's a link labelled RSS, which points to http://newyork.craigslist.org/brk/sub/index.rss. This file contains an XML document that provides a machine-readable version of the 15 most recent listings for that category. Here's a snippet:

```
<item rdf:about="http://newyork.craigslist.org/brk/sub/87294481.html">
<title>***Bedroom Avail in Apt with Garden- Aug Sublet*** (Williamsburg)
$650 3bd</title>
<link>http://newyork.craigslist.org/brk/sub/87294481.html</link>
<dc:rights>Copyright 2005, craigslist.org</dc:rights>
<dc:language>en-us</dc:language>
<dc:date>2005-07-27T14:52:39-04:00</dc:date>
```


Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

```
<dc:type>text</dc:type>
</item>
```

The main thing to note, of course, is the URL in the `<link>` element. That HTML page is the rental listing itself and, buried within it, are a few choice bits of HTML comments that Craigslist puts in every page:

```
<!-- START CLTAGS -->
<!-- DO NOT EDIT these unless you're really feeling brave and want your
posting
messed up. You have been warned. -->
<!-- CLTAG xstreet0=Boerum Street --><!-- CLTAG xstreet1=Bushwick Ave -->
<!-- CLTAG city=Brooklyn --><!-- CLTAG state=NY -->
```

These comment tags, of course, contain enough information to plot the approximate location of the rental on a map! We'll show how to do this for U.S. street addresses with Geocoder.us in "Find the Latitude and Longitude of a Street Address" **[Hack #62]**. The practical upshot is that, on the server side, the HousingMaps web site periodically spiders the Craigslist RSS feeds, finds new listings, and scrapes the location data out of each one. (The downside is that if there isn't any location information in the listing, it's awfully difficult to show it on the map!)

Finally, the site produces its own data file containing the listing information, links to photos, and, of course, the location data, which is then fed into the Google Maps interface in your web browser to produce the lovely maps you see before you, using `XMLHttpRequest`. Of course, now that Google has released a proper API for Google Maps, there are easy ways to do this yourself, as we'll also see later on in the book.

3.8.3. See Also

- Monkey Homes (New York, NY): http://monkeyhomes.com/map/maps.php
- Colorado Future (Denver, CO): http://www.coloradofuture.com
- ApartmentRatings.com (nationwide U.S.): http://www.apartmentratings.com

Hack 24. Search for Events by Location

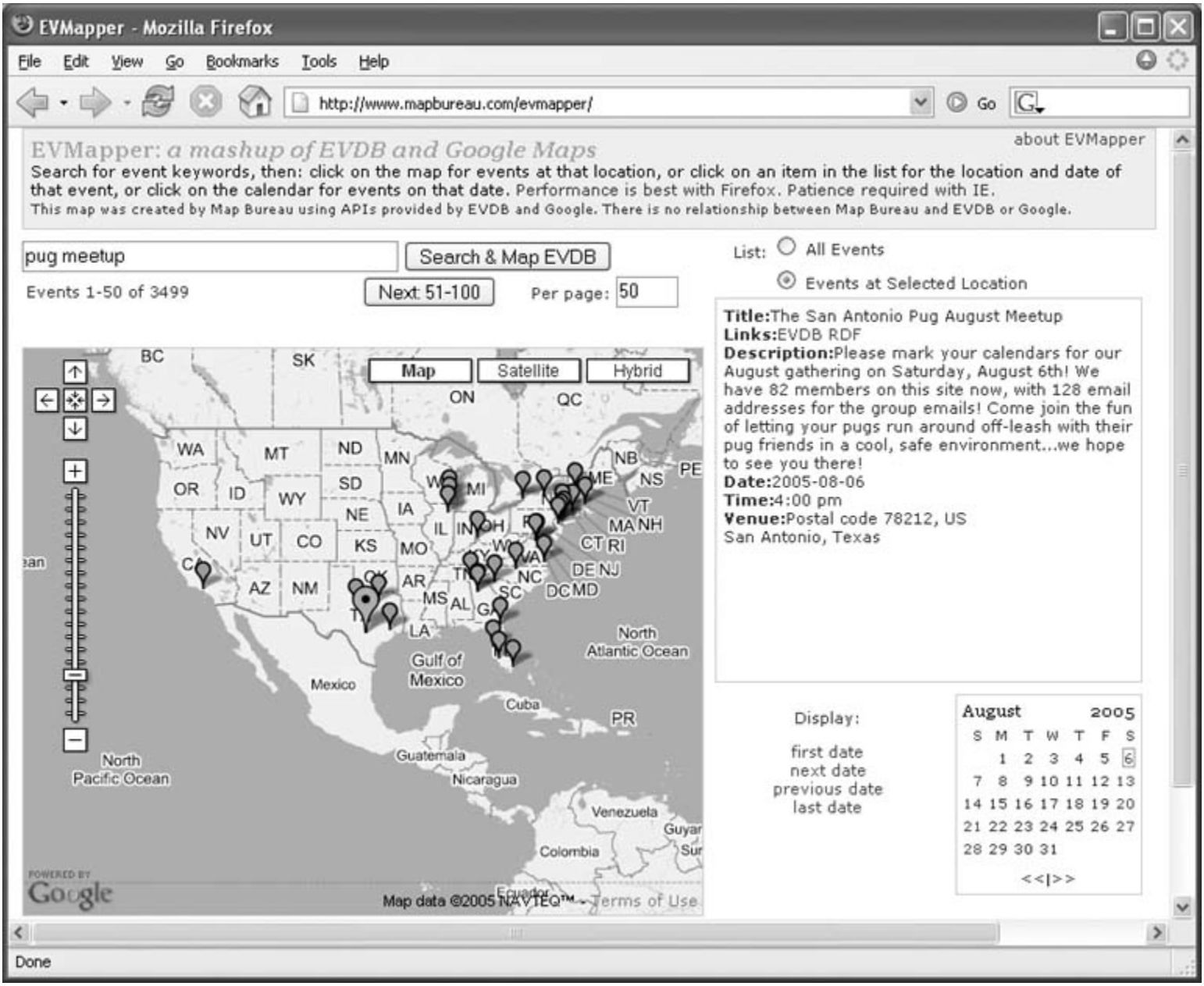
Events listed in the EVDB event database can easily be plotted on a Google Map.

Using a generic search engine isn't a good way to find out about events such as musical performances or garage sales. You can't specify that only event descriptions are wanted, and even if you could, the results aren't displayed in a way that organizes them usefully in time and space.

Wouldn't you like to be able to type keywords ("U2", "pug meetup", etc.) into a specialized event service, and get back not only a list of textual descriptions, but also a map showing event locations and a calendar highlighting event dates? Well, that's what EVMapper does. You can try EVMapper for yourself at http://mapbureau.com/evmapper/. Figure 3-15 is a screenshot of the results of the "pug meetup" search.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-15. Pug meetup locations



You can click on a map dot, or an event listing, or a highlighted calendar date, and details about the event at that place or time will be displayed. If there is more than one, a list is displayed instead. Whenever an event is selected, and however the selection took place, its location will be marked on the map by the big balloon and its date will be marked by a little red square on the calendar. You can also click your way through the events in time.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

3.9.1. How It Works

EVMapper is a mashup of two services: EVDB and Google Maps. EVDB (http://evdb.com) is the Events and Venues Database, a free site that allows anyone to submit or search for events. An EVDB search displays a list of events; events can also be organized into personal calendars. However, maps are not currently part of the experience at the EVDB site.

EVDB provides a RESTful API to the event database (http://api.evdb.com). When a user enters a query into EVMapper, an `/events/search` call is made to the API; for example:

```
http://api.evdb.com/rest/events/search?app_key=…&keywords=pug+meetup
```

The XML returned by EVDB contains descriptions of the matching events. The descriptions specify place by name (e.g., city, state, and sometimes ZIP Code or street address), not latitude/longitude, so EVMapper needs to do the geocoding. EVMapper implements the simplest possible geocoding method, which is adequate to demonstrate the idea of event mapping: it looks up city/state in a GNIS database covering the United States only. The Geographic Names Information System is the official repository of place names in the United States. You can query the GNIS at http://geonames.usgs.gov/. As of this writing, EVDB has recently begun to provide lat/long coordinates with some event descriptions, and EVMapper will soon exploit this development with dispatch.

EVMapper uses RDF, rather than EVDB's specialized XML, for its internal data. RDF makes EVMapper's representation and processing of events fully extensible, since RDF is built from an open-ended series of vocabularies, each for its own application domain. RDF datasets freely mingle vocabularies for as many domains as are relevant to the application at hand. In future, EVMapper may aggregate events from a variety of sources, not just EVDB. New bits of RDF vocabulary, asserting things like tonnage of ships for shipwrecks or Richter Scale values for earthquakes, will be added as needed, with no need to disturb the underlying implementation. Of course, translators from other formats may be required, as was the case for EVDB.

EVMapper is implemented in Fabl (http://fabl.net), a programming language for which RDF is the native data representation.

3.9.2. See Also

- The World Wide Web Consortium's RDF Primer at http://www.w3.org/TR/rdf-primer/.

  —Chris Goad

Hack 25. Track Your UPS Packages

With Google Maps and a simple Greasemonkey user script, you can watch your UPS packages travel across the country.

Anyone who's received a package delivered by UPS, or any other large shipping company, probably has had the experience of wondering where the heck the package is right now, and when exactly it's going to arrive. By entering the package's tracking number into a form on the UPS web site, you can get back a list of the cities which the package has traveled through to date. Of course, this is enough information to allow us to visualize the package's progress on a map!

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

3.10.1. The Hack

As usual, the trick of mashing up Google Maps with information from another site, such as that from the UPS tracking form, involves a bit of contortion, to get around security restrictions in the browser. One solution to this problem (at least for Mozilla Firefox users) is to use a Greasemonkey user script to modify the contents of a web page to include a link to a map of the things on that page. The Greasemonkey approach **[Hack #27]** is exactly the one taken by Matthew King, when he decided he wanted to visualize the path traversed by his new laser printer, on its way from the warehouse to his hometown.

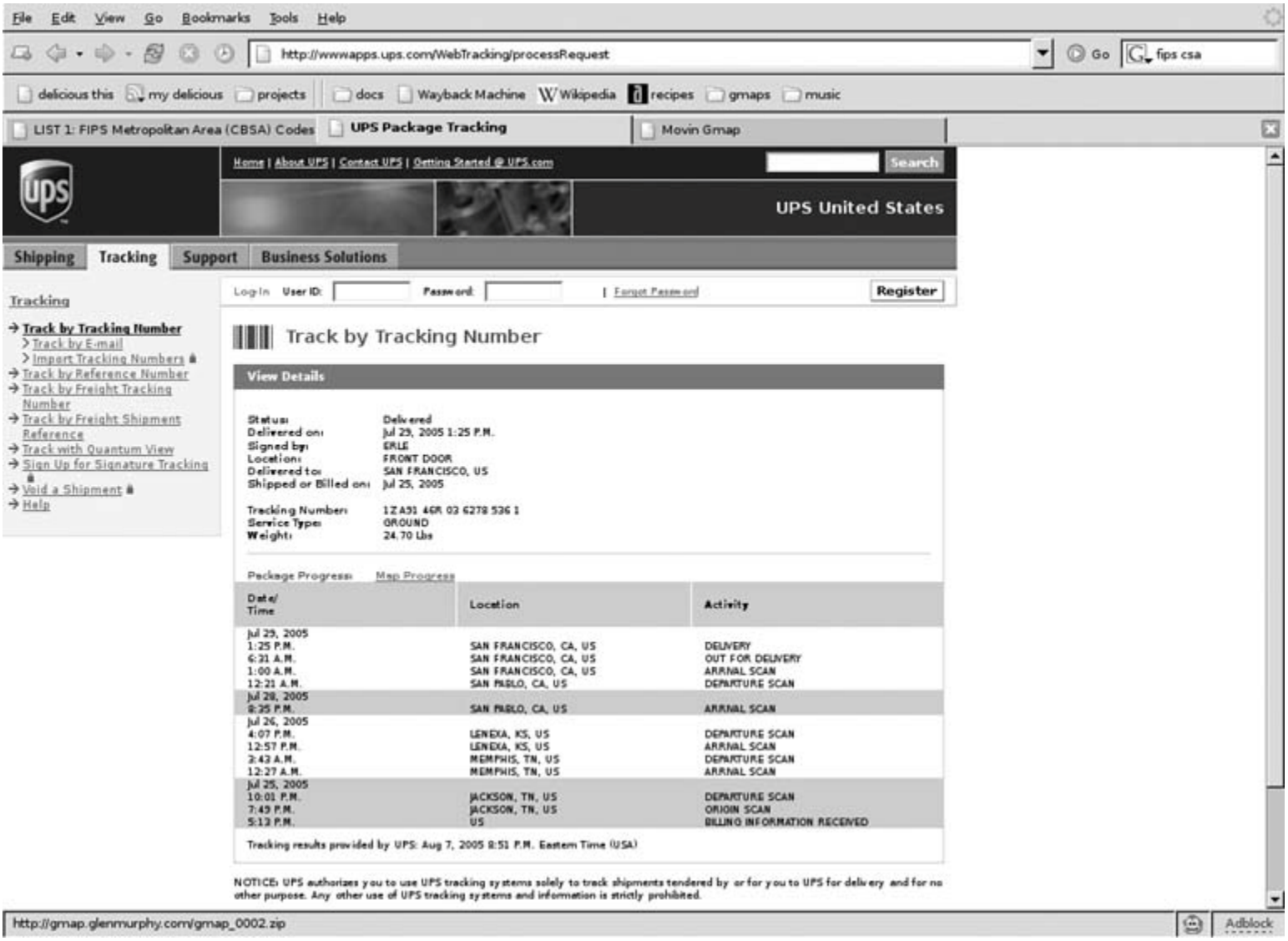
First, you'll need to be running Mozilla Firefox. You'll also want to install the latest version of the Greasemonkey extension from http://greasemonkey.mozdev.org/, if you haven't done this already; see "Add Google Maps to Any Web Site" **[Hack #27]** for more information on how this works. To install the UPS tracking user script, visit http://www.thrall.net/~mking/maps/upstrack.user.js in Firefox, and then select Tools → Install User Script from the menu bar. A confirmation window will pop up, in which you can simply click OK.

Now you're ready to track your UPS packages! Visit the UPS package tracking form at http://www.ups.com/content/us/en/index.jsx, enter a tracking number, click the checkbox to accept the terms and conditions of use, and then click the button marked Track. A summary page will load, with a "View package progress" link. Click this link as well. You should get a results page that looks something like Figure 3-16. This particular example shows the course of a box of O'Reilly's *Mapping Hacks* sent from the company's warehouse in Tennessee to our old house in San Francisco.

So far this looks just like the regular UPS package details page, with a list of cities, dates, and status messages. However, if you look closely, you'll see that the Greasemonkey script you installed earlier has added a special link to this page that reads simply "Map Progress." Go on, click it! A new window should open, showing a map of your package's progress across the country, as shown in Figure 3-17.

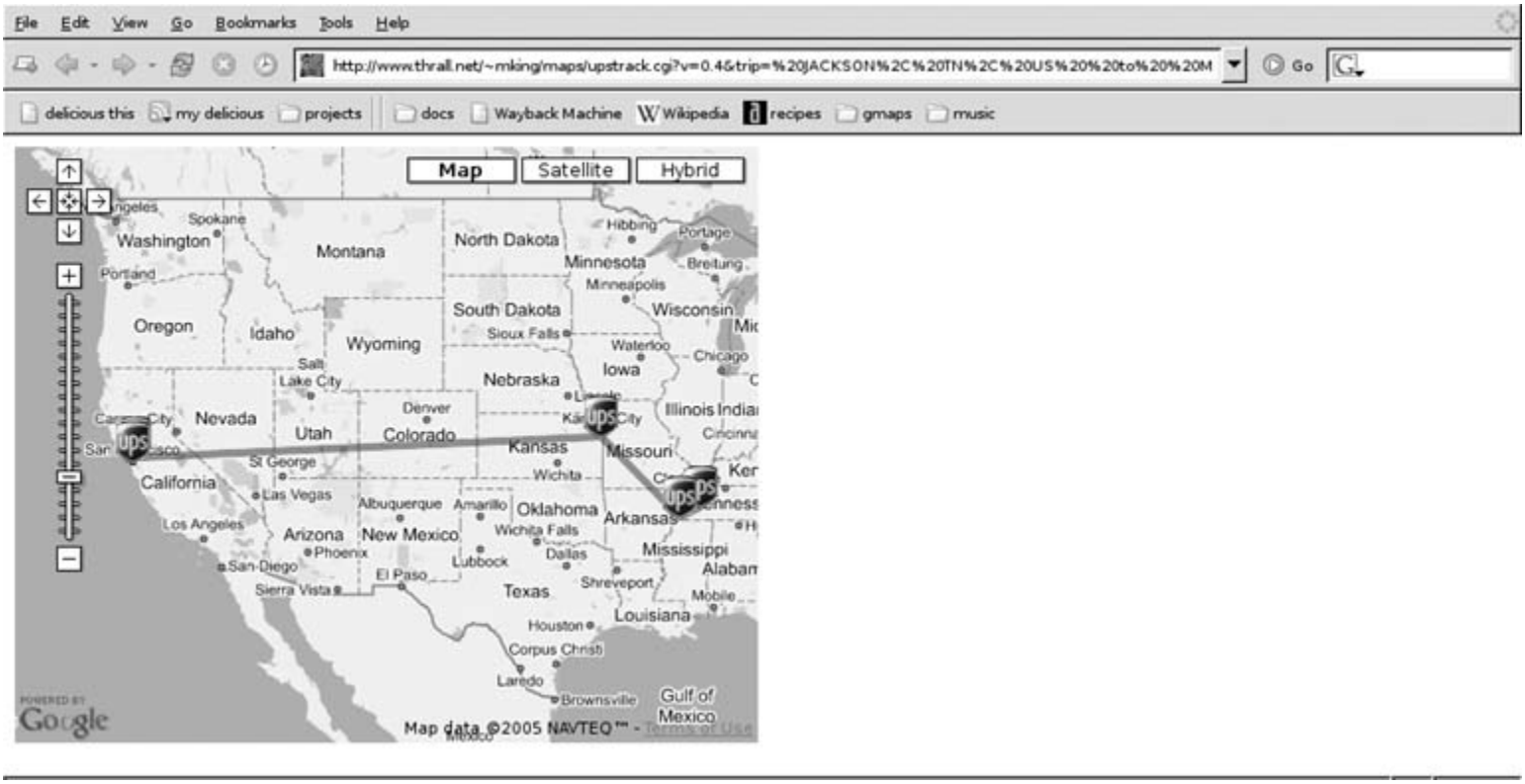
Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-16. The UPS package tracking details page, augmented by Greasemonkey



Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-17. The Google Maps representation of a package's travels



If you don't have a package of your own to track, you can always try out Matt King's example page at http://www.thrall.net/~mking/maps/ups_sample.html.

3.10.2. The Code

The Greasemonkey code for this hack is actually really simple and offers a good example of how to use Greasemonkey to pick elements from an existing page on the Web and use them to insert new elements into the same page. This first chunk of code from *upstrack.user.js* extracts all the locations from the UPS detailed results page:

```
var lastLoc = null;
var loc = null;
var locations = new Array;
var allDivs, thisDiv;
allDivs =
    document.evaluate("//div[@class='modulepad']", document, null,
                      XPathResult.UNORDERED_NODE_SNAPSHOT_TYPE, null);
for (var i = 0; i < allDivs.snapshotLength; i++) {
    thisDiv = allDivs.snapshotItem(i);
    var html = thisDiv.innerHTML;
    html = html.split(/[\t\n\r]+/).join(' ');
    html = html.replace("<br> ", '');
    if (html.indexOf(', US') == -1)
        continue;
    loc = html;
```


Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

```
        if (loc == lastLoc)
            continue;
        locations[locations.length] = loc;
        lastLoc = loc;
    }
```

The code starts by creating an empty array of locations and then passes an XPath to Firefox's `document.evaluate( )` method to find all of the HTML `div` elements of the class `modulepad`, which apparently contain the location strings. The script then iterates over each `div` node, extracting the location string, cleaning it up a bit, and then pushing it on to the array of locations, checking each one to make sure that it's not redundant with the locations already stored.

The next bit of code in the script handles the insertion of the Map Progress link into the results page:

```
    if (locations.length > 1) {
        locations.reverse();
        var locStr = locations.join(" to ");
        allDivs =
            document.evaluate("//span[@class='brownbold']", document, null,
                              XPathResult.UNORDERED_NODE_SNAPSHOT_TYPE, null);
        for (var i = 0; i < allDivs.snapshotLength; i++) {
            thisDiv = allDivs.snapshotItem(i);
            var html = thisDiv.innerHTML;
            var estr = escape(locStr).replace(/\ //g, '%2F');
            html += '     '
                 + '<a href="http://www.thrall.net/~mking/maps/upstrack.cgi?'
                 + 'v=0.4&trip=' + estr +
                 '" target="_blank">Map Progress</a>';
            thisDiv.innerHTML = html;
        }
    }
```

In this section, the code checks to see if any locations were found, and, if so, joins them with the string "`to`", yielding something like `NASHVILLE TN to KANSAS CITY MO to SAN FRANCISCO CA` in the `locStr` variable. The script then looks for `div` elements of the class `brownbold`, which presumably holds the "Package Progress:" text displayed on the page above the list of locations, and inserts a link to Matt King's http://www.thrall.net/~mking/maps/upstrack.cgi script into the `div's innerHTML`, passing a URL-escaped rendition of `locStr` as an HTTP GET parameter. This results in the Map Progress link that you can click to show the map in a new browser window. The server-side script geocodes each location and returns a Google Map with markers based on the UPS logo over each location, with colored polylines connecting them—thus describing the path of your package.

3.10.3. Hacking the Hack

You'll note that this code also ensures that the location is within the United States, which means it won't work if the package's origin or destination is outside the States. If you live outside the U.S. (and are feeling adventurous), you might try removing or commenting that line out in your local copy of *upstrack.user.js* (buried somewhere in your Mozilla user directory) and then restarting Firefox to see if it does indeed work for you.

Hack 26. Follow Your Packets Across the Internet

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Ever wonder where your network traffic goes when you visit a site on the 'Net?

Not long after the Internet began to be widely used in our society, a new word began to gain currency among the public to evoke the experience of being able to find information and communicate with people from all over the globe, and that word was *cyberspace*. In fact, the word "cyberspace," taken from the term *cybernetics*, a technical term for human-computer interfaces, has been so overused that it comes across as trite or hackneyed today. All the same, the word conjures up an image of sweeping digital vistas, waiting to be explored and homesteaded, and so has a great deal of potency—which is probably why the word became a cliche in the first place.

The fact, however, is that the Internet works so beautifully and, usually, so transparently, that most people don't take the time to consider that cyberspace and meatspace (as we hackers sometimes jokingly refer to the Real World) are actually connected. Obviously, every web server, DSL router, cable modem, dial-up service, and so on, is located somewhere on the planet. But who knows where?

3.11.1. From Clicks to Bricks

As it happens, the Whereis service at http://www.parsec.it/whereis/ knows where Internet addresses are hosted in the real world, sometimes with astonishing accuracy. The front page of the site, clearly modeled after Google's, offers a simple search box, where you can type in an Internet domain name or an IP address in dotted-quad format (e.g., `192.168.1.1`). Clicking the locate button takes you to the view shown in Figure 3-18, with a Google Map showing a marker over the most probable physical location of that Internet address. Clicking on the marker pops up some basic information about the address, including the country and locale that it's believed to be physically located in or near.

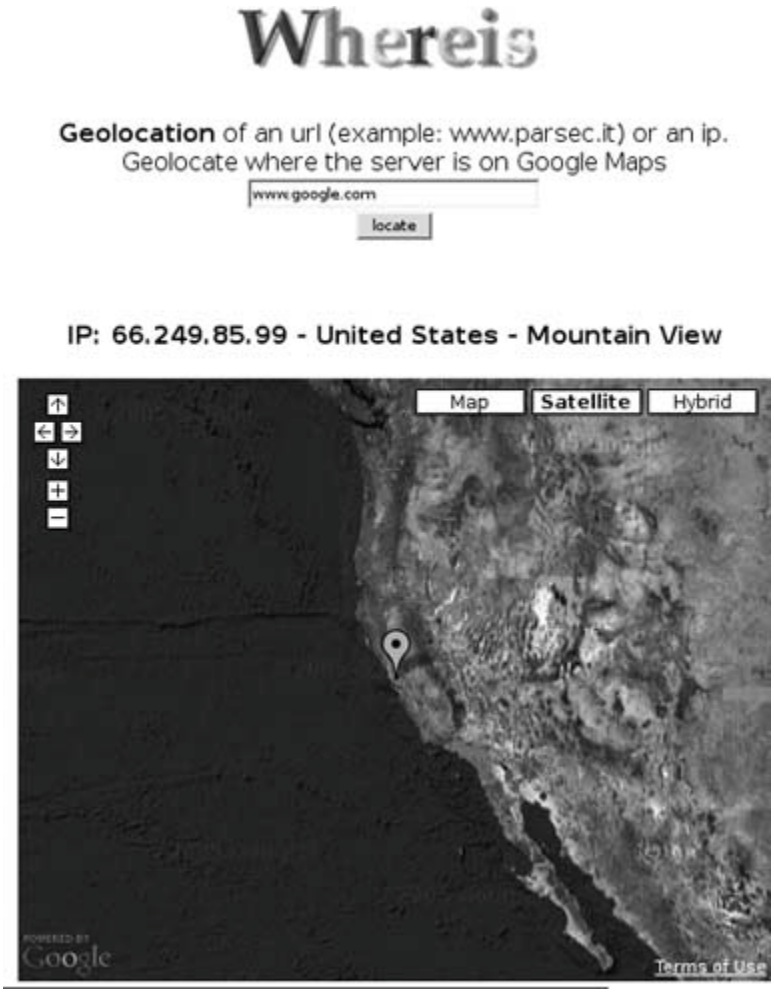
Whereis uses the standard Google Maps API to display the map on the results page. Embedded within a JavaScript block on the results page is a call to the `GMarker( )` constructor, which specifies the physical coordinates of the Internet address and generates the marker that you see on the map.

What's particularly interesting about Whereis is how accurately it identifies the approximate location of high-speed residential connections, such as DSL and cable modems. If you have such a connection, try putting in your own IP address at home. If you don't happen to know what your IP address is at home—and it may be assigned dynamically—you can use an online service like http://www.whatismyipaddress.com/ to find out what public IP address you're appearing from, and then cut and paste that address into the Whereis search box.

What's even more interesting about Whereis is that when it fails, as it might if your Internet provider uses an upstream web proxy (which AOL has been known to do). As a result, Whereis may decide that you're in, say, Reston, Virginia, even though the sign outside your house says "Welcome to Rapid City, Iowa!" Note also that what Whereis tries to return is the physical location of the hardware hosting the domain, not the place that the web site or even the domain name purports to represent, which is why web sites such as *zooleika.org.uk* and www.freemap.in turn up in Fremont, California, rather than in London or Mumbai. Also, it's conceivable that some large web sites might be hosted in different locations, with different IP addresses for the same domain name, which might result in different locations being returned on different tries for a single address.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-18. Whereis correctly places www.google.com in Mountain View, California



3.11.2. How It Works

The fact that Whereis does as well as it does seems nothing short of miraculous, under the circumstances. The Internet Address and Naming Authority (IANA), licenses a number of organizations around the world to manage the allocation of IP addresses and domain names. Each of these organizations maintains its own public database of address assignments, which typically can be accessed through the *whois* service on the 'Net. The problem is that each *whois* database returns results that have different information and are formatted in different ways. Not only that, but even once you've got, say, a mailing address for the owner of a given range of IP addresses, you still don't necessarily know where that place is in the world, in terms of latitude and longitude, which means you can't yet put it on a map or say what else is nearby.

So, the problem of physically locating an IP address turns out to be quite difficult—so difficult, in fact, that there aren't any worthwhile free-as-in-freedom sources of this information on the 'Net that are more precise than the country level. Mapping IP addresses to countries can be useful for collecting statistics on international visitors to your weblog, but it's no good for making decent maps. Instead, Parsec Tech s.r.l., the maintainers of the Whereis service, would seem to be using MaxMind's GeoIP database, which, as you can see, does offer pretty impressive results. You can learn about MaxMind's products, which, interestingly enough, are used first and foremost for credit card fraud prevention, at http://www.maxmind.com/.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

3.11.3. Hacking the Hack

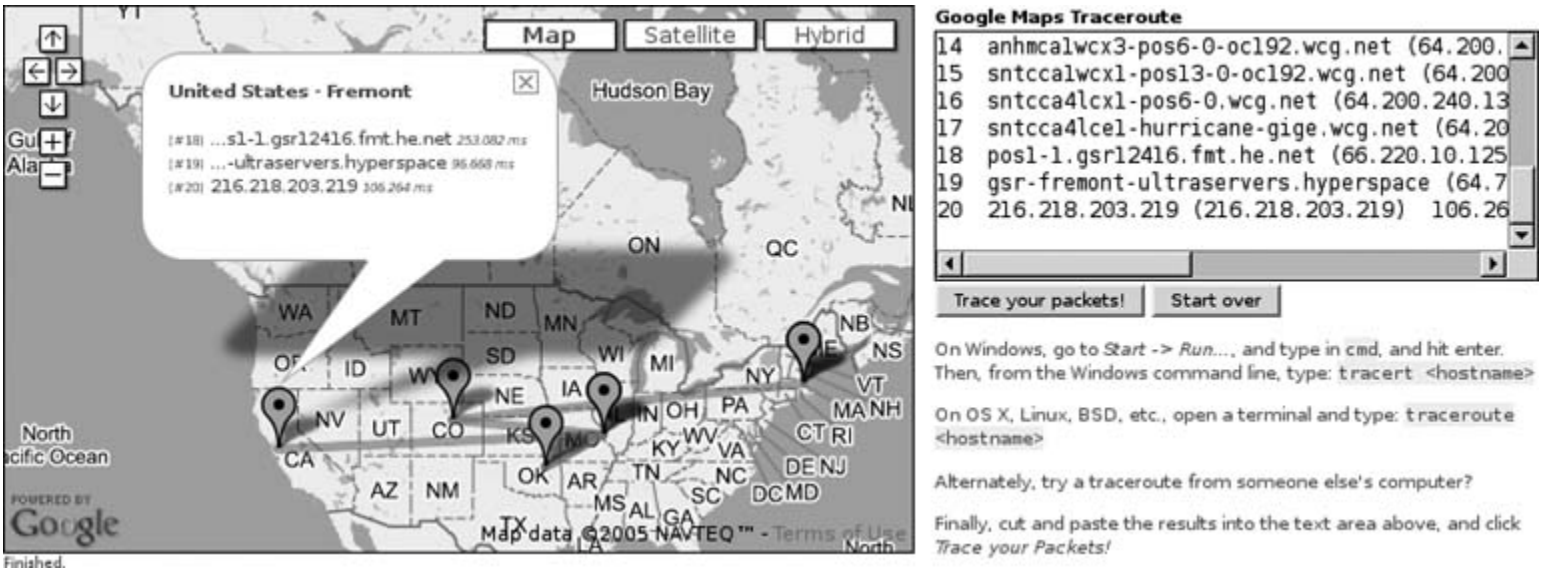
Seeing where your favorite web sites actually live in the real world can be quite fascinating, but doesn't it make you curious to see how your requests get there in the first place? It definitely did for me, so I decided to hack the hack, by building a traceroute mapping service on top of Whereis. In technical parlance, the term traceroute is used to describe an attempt to discern which computers an Internet Protocol packet travels through on the way to its destination, after the Unix network diagnostics tool designed for that purpose. (A very similar tool ships with Windows, but its name has been abbreviated to *tracert*.)

The *traceroute* utility works as follows: all traffic sent over the Internet is broken up into packets, and each packet that's sent is marked with the IP address of the sender and the intended receiver. Additionally, each IP packet is marked with a Time-to-Live (TTL), which specifies how many network hops the packet can travel through. Each time a computer forwards the packet towards its destination, the TTL value in the packet header is decremented by one, and if it ever reaches zero, a message is sent back to the sender informing it that the receiver was unreachable. This feature of the Internet Protocol is designed to allow network engineers to detect loops and other routing problems.

traceroute piggybacks on this process, by first sending out a test packet to a given destination with a TTL of 1, and then a packet with a TTL of 2, and so on. Each time, the computer at each successive network hop along the way drops the TTL to zero and bounces the packet, thereby revealing its IP address. The process continues until the TTL value reaches the number of network hops to the destination, at which point the entire route is known. The time taken to perform each step provides an estimate of round-trip network latency and can be used to identify bottlenecks in a network route.

Fortunately, *traceroute* has a sufficiently simple output format that the results can easily be parsed in JavaScript by a web browser—and that's exactly what the Google Maps traceroute at http://mappinghacks.com/projects/gmaps/traceroute.html does. Start by running `traceroute <hostname>` from your *nix or OS X terminal, or `tracert <hostname>` from the Windows command line, where *<hostname>* is an Internet domain name or an IP address. Copy the output to the clipboard, then go back to your web browser and paste the results into the text box on the right, as shown in Figure 3-19. Finally, click "Trace your packets!" and watch the hosts between you and your chosen destination appear on the map, one by one. Clicking on any of the markers that show up opens an info window that shows the country and locale, the servers hosted there, and the reported network latency to that host.

Figure 3-19. The route from New York City to googlemapshacks.com



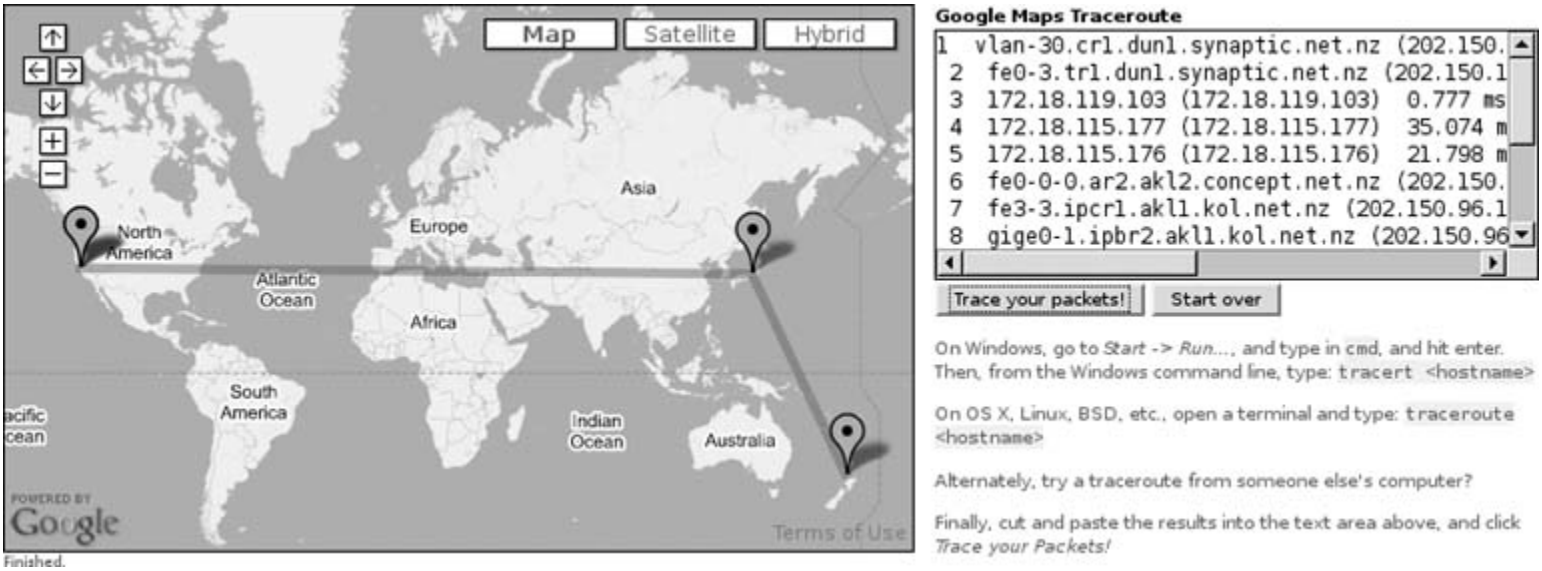
Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

On Windows, you can get to the command-line interface by going to Start → Run, typing in `cmd`, and then clicking OK.

Under the hood, when you click Trace, a JavaScript function parses the individual entries from the supplied *traceroute* output by looping over it with a regular expression. Next, it sends each IP address to Whereis via asynchronous `XMLHttpRequest()`, and then uses another regular expression to scrape the coordinates out of the JavaScript in the returned HTML page. As the results come back, a marker is created on the map for each unique location, using Google Maps `GMarker` objects. Some care is taken to note when Internet hosts are listed as being at the same location, so as to avoid redundant overlapping markers. Additionally, a line is drawn between each successive location with `GPolyline` overlays, to mark out the path of your packets as they speed through the ether. The code runs to over 180 lines, so we don't have room to print it here, but you can always view the source of the page in your browser if you're curious to see how it works.

Try it with a few different hosts and see what sort of results you get. You might see certain patterns emerge, such that your packets may have to run through a number of specific hops just to get out of your ISP. For variety's sake, you might try mapping traceroute results from any of the online traceroute services listed at http://traceroute.org/. Finally, if you map network routes that cross the Pacific Ocean, you may discover an interesting flaw in Google Maps polylines, as shown in Figure 3-20, which is that they can't cross the International Date Line! Oops!

Figure 3-20. Hey, wait! You're going the wrong way!



Hack 27. Add Google Maps to Any Web Site

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Greasemap adds informative Google Maps to almost any page on the Web.

There are lots of sites on the Web that don't have maps but could really use them. Any web page with an address or a set of latitude and longitude coordinates is a candidate for a potentially useful map. Let's say, for example, that you want to visit O'Reilly's global headquarters to purchase the next hot Animal book straight from the bookstore in the reception lobby. You could go to http://www.oreilly.com/oreilly/contact.html and get the address, but what good does that do you? At the very least, you'll have to copy the address to your clipboard, and then go to http://maps.google.com/ in another browser window, paste the address in there, and so on. Isn't there some way your web browser could just do this for you?

3.12.1. Greasemap to the Rescue

As it happens, there is a way—if you're running Mozilla Firefox. Firefox supports an extension called Greasemonkey, which allows custom JavaScript-based *user scripts* to, well, monkey with the contents of any page you view in your browser. As you can imagine, there is a wealth of possibilities inherent in the Greasemonkey concept.

Vinq.com's Greasemap is one such user script for Greasemonkey. Greasemap augments any web page you visit, by searching the page for street addresses and clearly marked geographic coordinates. If it finds any, Greasemap provides a clickable message bar across the top of the page, notifying you that a map is available. Clicking on the message bar adds an IFrame to the top of the page, in which a Google Map is loaded, displaying all the locations found on that page.

The maps from Google remain interactive; you can zoom in, pan around, and so on. The map will be one wide map if there are multiple locations found, or two smaller maps—an overview map and a zoomed in map—if there is just one location found on the page.

3.12.2. Installing Greasemap

Installing Greasemap takes a few short steps and should be almost painless.

1. Download and install Greasemonkey (http://greasemonkey.mozdev.org/). When you first try to install Greasemonkey, you may see a thin bar at the top of Firefox that says "Firefox prevented an extension from loading." Click the button at the right side of that bar, then "Allow extensions from that site." Once you've done this, try installing Greasemonkey again; it should work the second time around.
2. Restart Firefox. This allows Firefox to load the newly installed extension.
3. Visit.the Greasemap homepage at http://www.vinq.com/greasemap/. About halfway down the page, you'll see a link to install the Greasemap script. Right-click this link, and then choose the Install User Script option to install the Greasemap script in your browser.

To uninstall Greasemap, go to the Tools → Manage User Scripts menu in Firefox, select Greasemap, and click the Uninstall button. You can also use this panel to temporarily disable it.

You can also temporarily disable Greasemap by disabling Greasemonkey itself, by clicking the Greasemonkey icon in the lower-right corner of your browser. The icon should turn gray, indicating that Greasemonkey is deactivated. Clicking the icon again reactivates Greasemonkey.

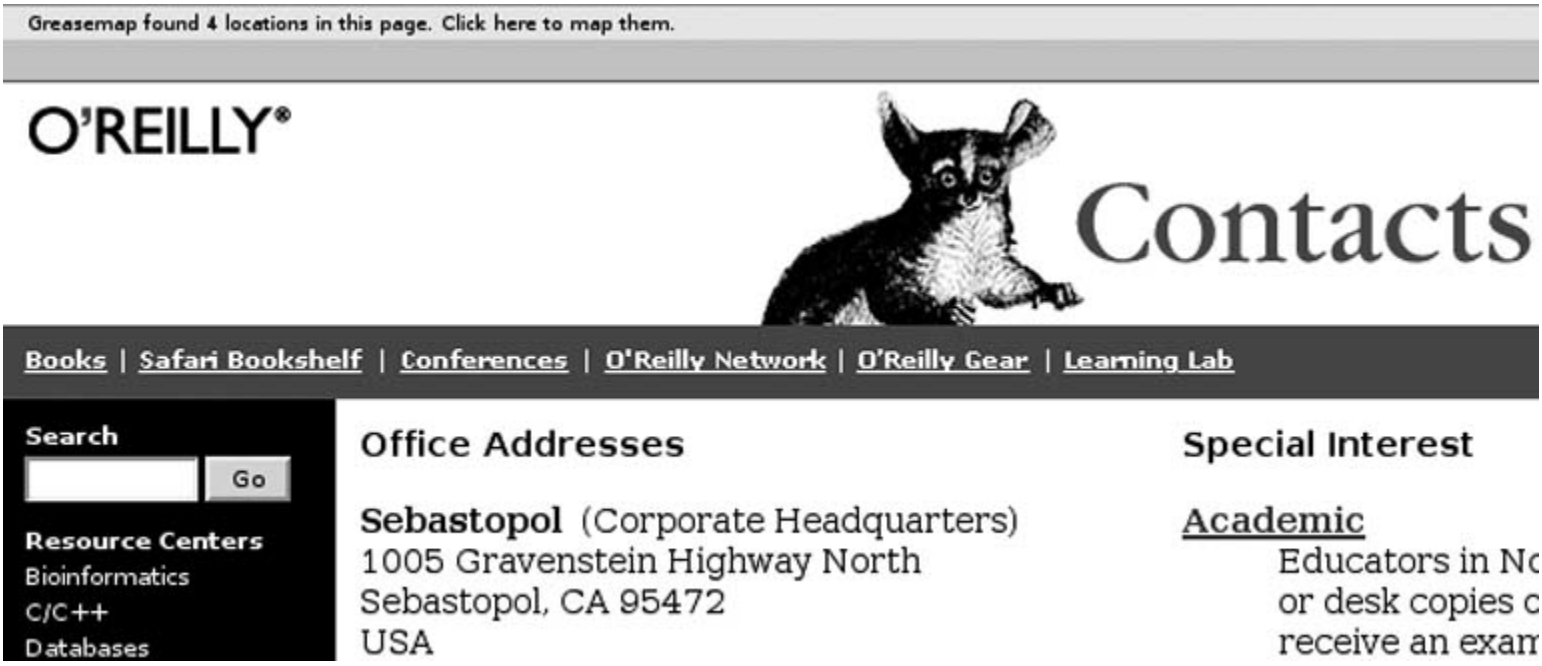
To upgrade Greasemap, first uninstall it, then follow Step 3 above to install the latest version. Unfortunately, there's no way to avoid uninstalling first—this is a limitation of Greasemonkey.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

3.12.3. Taking Greasemap Out for a Spin

Once you've got Greasemap installed, getting a map for any location shown on a web page becomes a snap. Figure 3-21 shows the Greasemap version of O'Reilly Media's contact page, with the message bar at the top of the page.

Figure 3-21. O'Reilly Media's contact page, with the Greasemap message bar



Clicking on the message bar loads a Google Map in an Iframe at the top of the page, with a marker placed on the map for each location found on the page, as shown in Figure 3-22. Clicking a marker on the map highlights the matching location on the right side of the frame, and clicking the "Big map in new window" link does just what it says: it opens a larger version of the same map in a new browser window. From this larger map, you can click on the markers to get an info window that allows you to search for directions to that place, as shown in Figure 3-23.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-22. O'Reilly Media's contact page plus the Greasemap frame

Figure 3-23. Search for driving directions in a new browser window with a single click

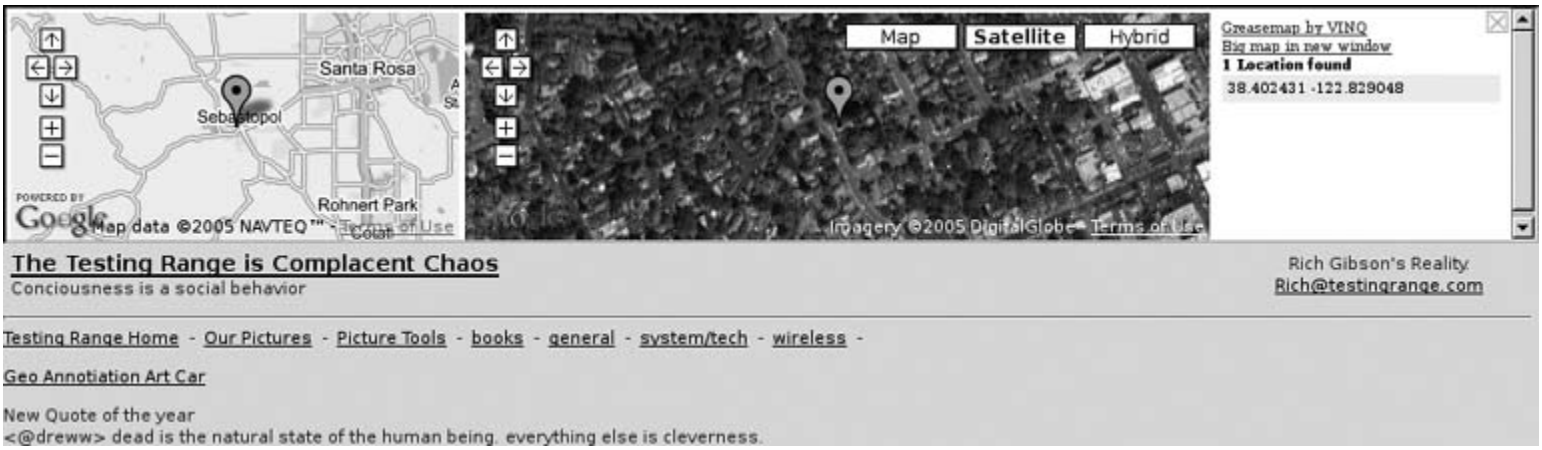
Instead of a whole lot of frustrating cutting and pasting, and opening new browser windows, Greasemap turns the process of finding directions to a location shown on a web page to a mere three clicks of the mouse button.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Sometimes Greasemap reports more locations in the message bar than it actually shows on the map. The reason for this is that although Greasemap is fairly liberal about identifying locations on a web page, occasionally not all of them can be identified with actual geographic coordinates when it comes time to generate the map.

Addresses aren't the only thing that Greasemap can illustrate on a map. Several other types of location, such as *geourl.org*-style ICBM tags can be detected and mapped, as well. For example, Figure 3-24 shows the "location" of Rich Gibson's home page at http://www.testingrange.com/. Although there aren't any addresses directly visible on Rich's site, there *is* an HTML `meta` element embedded in the page that specifies a latitude and longitude, that Greasemap can identify and plot on a Google Map.

Figure 3-24. Rich's home page, automatically identified and mapped by Greasemap



3.12.4. How Greasemap Identifies Locations

Greasemap uses JavaScript *regular expressions* to identify locations in a web page. The following styles of location reference are currently supported:

`1600 Amphitheatre Parkway, Mountain View, CA 94043`

At the moment, only U.S. addresses are supported. The ZIP Code is optional.

`<meta name="geo.position" content="41.328,-110.292">`

This style of HTML meta element is used by sites that are indexed from *geourl.org*.

`<meta name="ICBM" content="41.328,-110.292">`

This is an alternate form of the same.

`N 42 25.159 W 071 29.492`

Degrees and decimal minutes are often used as geocache coordinates on *geocaching.org*.

`geo:lat=…and geo:long=…`

This W3C-derived style of geotagging is frequently used on sites such as del.icio.us and Flickr. See http://www.flickr.com/photos/tags/geotagged/ for examples.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Since the Google Maps API doesn't offer address lookups, Greasemap currently turns U.S. street addresses into latitude/longitude coordinates on the server side, using the Census Bureau's TIGER/Line data with the `Geo:: Coder::US` Perl module. Canadian and UK addresses should be coming soon.

If you're interested in having Greasemap identify other kinds of addresses or coordinates, have a look at the JavaScript source at http://www.vinq.com/greasemap/. If you wind up modifying the Greasemap source to include your favorite way to specify a location, please send us the code! We're always looking for new ways to improve the usefulness of this tool.

3.12.5. See Also

- `Geo::Coder::US` can be found on the Web at http://geocoder.us/. See "Find the Latitude and Longitude of a Street Address" **[Hack #62]** for more info on how to use the web service based on it.
- Mark Pilgrim's *Dive Into Greasemonkey* provides an excellent introduction to creating and improving Greasemonkey scripts. You can read the entire book online at http://www.diveintogreasemonkey.com/.

—*Mark Torrance*

Hack 28. How Big Is That, Exactly?

Explore the size of geographic regions in terms of those you already know.

A Google search for "roughly the size of" returns 398,000 results. Certainly this suggests that, when it comes to visualizing the size of faraway places, we feel the need to refer to the unknown in terms of the familiar. Take this newspaper quote, for example:

> …when the Amazon shrank a record 11,200 square miles, an area roughly the size of Belgium, or the American state of Massachusetts….

I read that and get a vague sense of confusion and think, "Wow, I bet the people in Belgium and Massachusetts know what that means, but I sure don't." There's more:

> "Huge iceberg menaces Antarctica…roughly the size of the island of Jamaica." (http://www.theregister.co.uk/2005/05/20/iceberg_penguin_menace/)
>
> "The largest-known iceberg was from this region. It was roughly the size of the state of Rhode Island." (http://www.factmonster.com/ipka/A0781668.html)
>
> "Months later, after I discussed what I had seen with the oceanographer Curtis Ebbesmeyer, perhaps the world's leading expert on flotsam, he began referring to the area as the 'Eastern garbage patch.' But 'patch.' doesn't begin to convey the reality. Ebbesmeyer has estimated that the area, nearly covered with floating plastic debris, is roughly the size of Texas." (http://www.naturalhistorymag.com/1103/1103_feature.html)

A search for "roughly the size of" reveals interesting things. The fact that the writer needed to find something, anything, to use to provide a measure of spatial context hints at something odd and interesting. An "Eastern garbage patch…roughly the size of Texas"? The image is vivid.

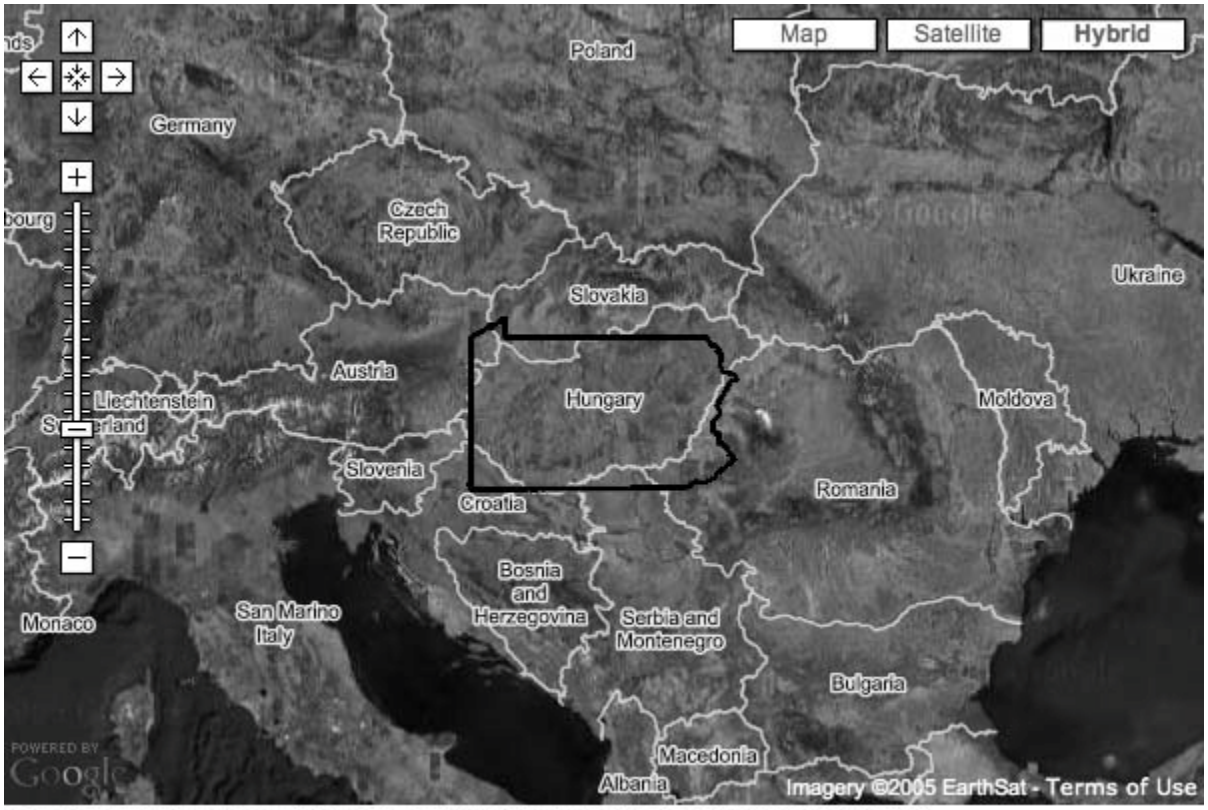
Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

3.13.1. The Hack

The statement "roughly the size of…" demands the ability to compare the unknown with a known quantity. At http://www.mappinghacks.com/projects/gmaps/size_of.html you can select the outline of a state or country and pin it onto the map. Pick the shape from a drop-down box then scroll around on the map until you find a familiar place and click. Figure 3-25 shows an outline of Pennsylvania on top of Hungary.

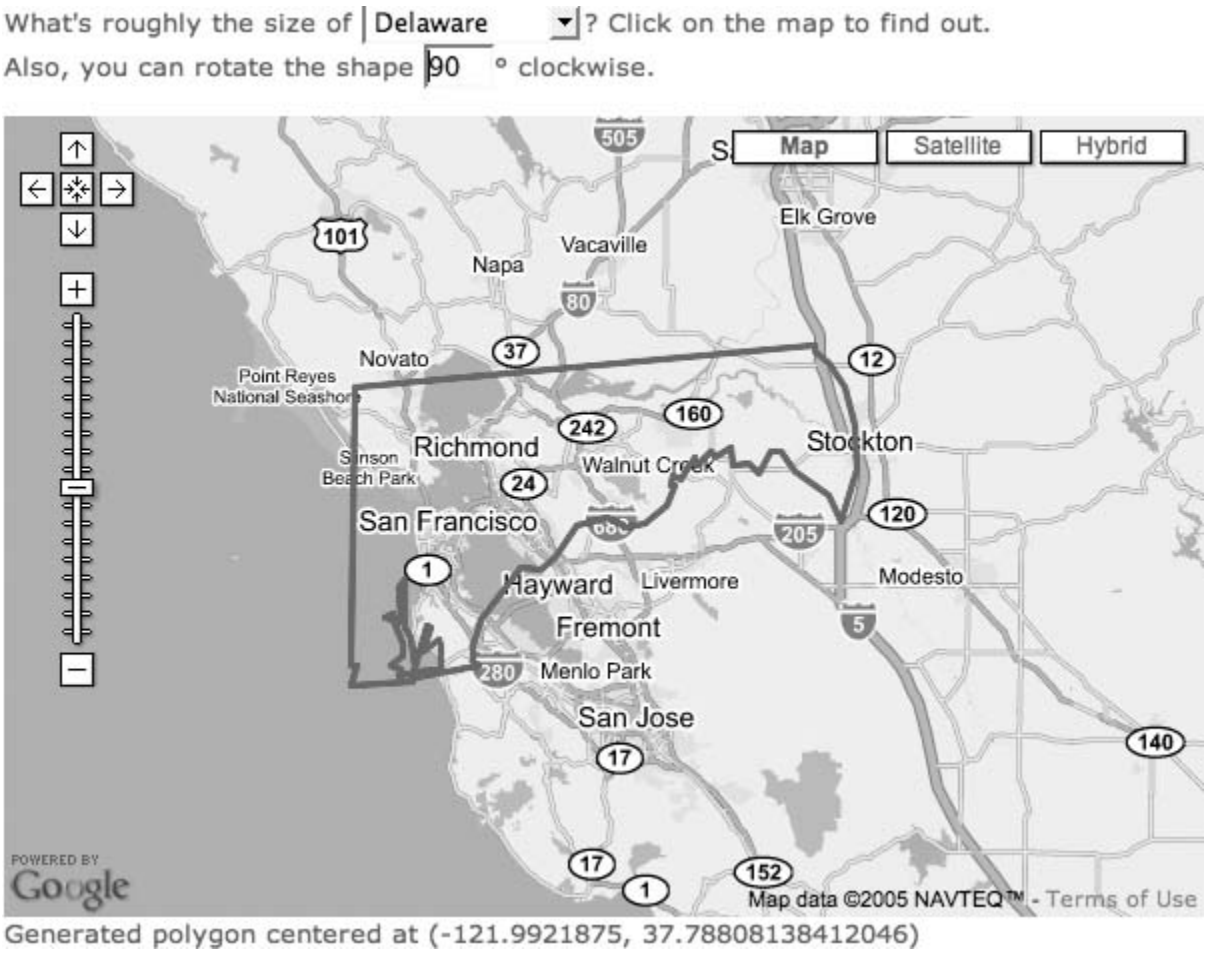
State boundaries can have an historical basis, but often are rather arbitrary. There is no particular reason for us to respect the work of our forebearers, and with Google Maps we don't need to! Figure 3-26 shows an outline of the state of Delaware, rotated 90 degrees clockwise and pinned over the San Francisco Bay area.

Figure 3-25. Hungary is roughly the size of Pennsylvania



Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-26. Delaware is roughly the size of the Bay Area

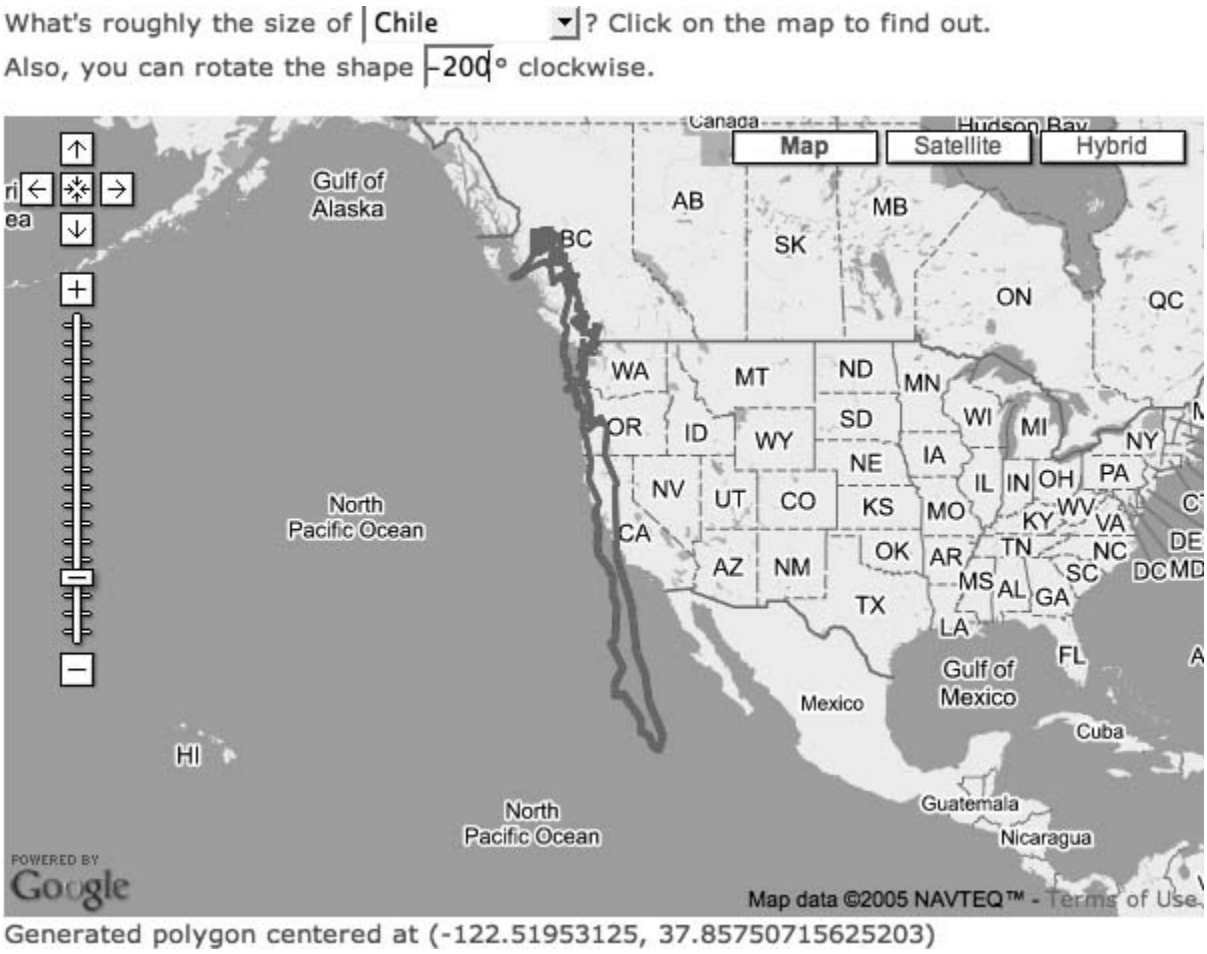


Depending on how you maneuver the outline, Delaware is roughly the size of the San Francisco Bay Area. Or perhaps the Bay Area is roughly the size of Delaware. Once you start doing comparisons like this, you need a fully declensed language, like Latin, so you can specify which place is nominative and which one accusative. (Oddly, neither would be locative.)

Pinning a state or country onto the world expands our understanding of geography. For example, people don't appreciate how very long Chile is. Figure 3-27 shows Chile on top of the west coast of California—it goes from the southern tip of Alaska to the bottom of Baja California!

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-27. Chile flipped over and pinned to the west coast of North America

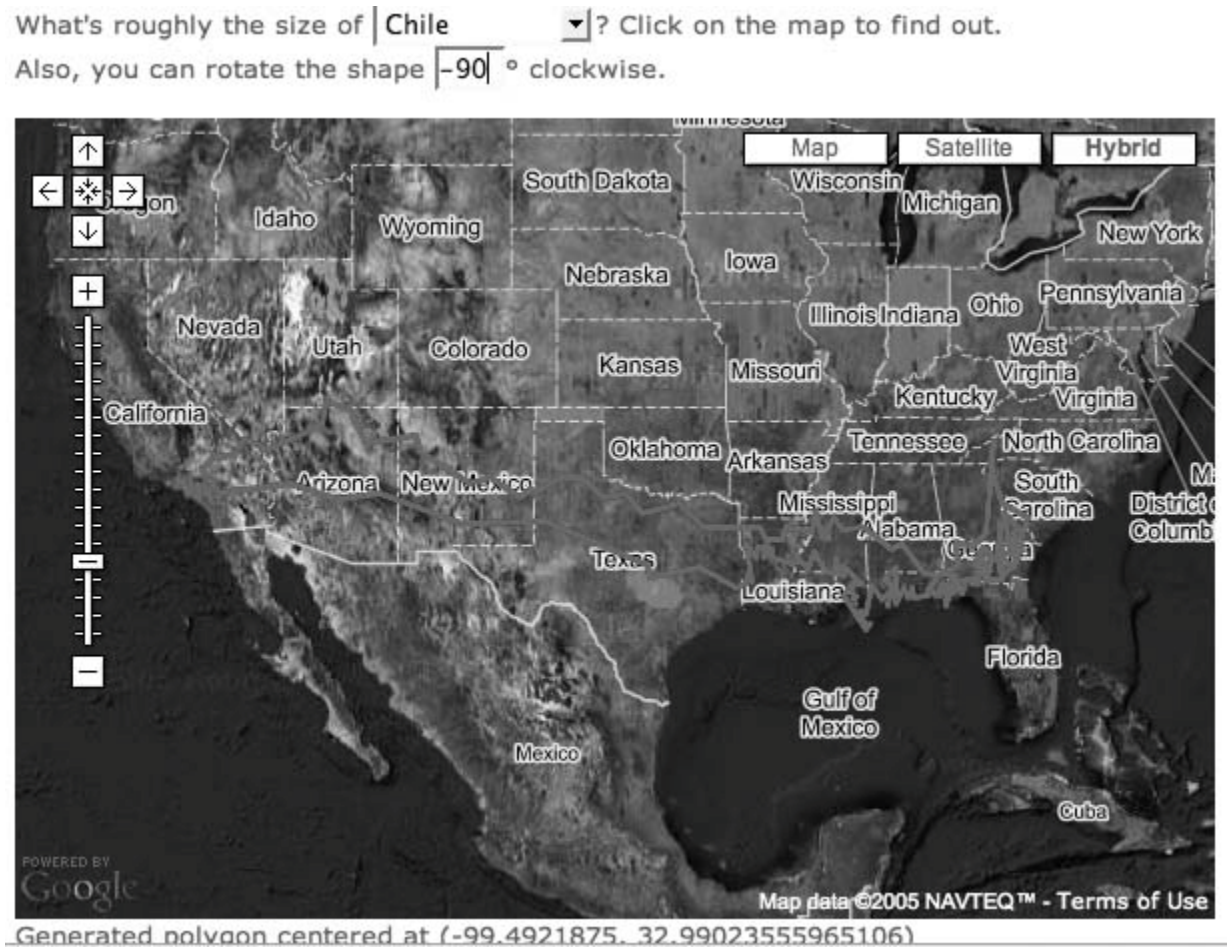


This image of Chile has been flipped, so southern Chile is near Alaska and northern Chile over Baja. The latitudes roughly match, and there is a general correspondence in climate. The north of Chile includes the Atacama desert, the driest place on earth. In parts of the Atacama, there has been no rainfall in recorded history. Baja California is not quite that dry, but it is pretty darn dry! The central part of Chile is a ripe agricultural region, analogous to California, then tending toward forests in the Southern Lakes district, which is evocative of the Inland Passage of Canada and Alaska. (Again, we have that nominative and accusative question: which one is analogous to the other, and which one is the defining instance?)

When we rotate Chile and lay it over the whole of the United States, it reaches most of the way from California to Florida, as shown in Figure 3-28.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

Figure 3-28. Chile shows her length



You don't need Google Maps to demo this, but the ease with which the Google Maps API handles everything on the client side (from rendering the map background to overlaying the boundary outlines to handling the user interaction), made this interesting idea an implementation in just a few hours.

3.13.2. The Code

We started by taking outlines of our example regions from common sources in ESRI Shapefile format. We then used a simple Perl script to extract these outlines and calculate the bounding box of the coordinates in the outline. Taking the average of the corners of the bounding box, we estimated a centerpoint for each region, and then calculated the offset of each coordinate in the perimeter from that center point. This allows us to draw the outline of the country or state around any arbitrary point that a user might click on the map.

Also, we took this opportunity to quantize the coordinate values—essentially snapping them to a grid of an arbitrary size—and then removed the duplicate coordinates to get the region's outline down to a number of points that's manageable for Google Maps. Additionally, we scaled the longitudinal distances by the cosine of the center latitude, so that the areas display at more or less the right size anywhere on the map. These doctored outlines were then put in individual text files on our server. The JavaScript uses the `GXmlHttp` class from the Google Maps API to fetch the file for a given country or state from our server:

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

```
function loadPolygonData (url, displayNow) {
    if (!url)
        url = document.forms['selectPoly'].area.value;
        var xmlHttp = GXmlHttp.create();
        msg.innerHTML = "Fetching " + url;
        xmlHttp.open('GET', url, true);
        xmlHttp.onreadystatechange = function () {
        msg.innerHTML = "Fetching " + url + "(readyState "
        + xmlHttp.readyState + ")";
        if (xmlHttp.readyState == 4) {
            shape = parsePolygonData( xmlHttp.responseText );
            if (displayNow) {
                var center = map.getCenterLatLng();
                drawPolygon(null, center); }
            }
        }
    }
    xmlHttp.send(null);
}
```

The URL of the file that gets fetched is pulled from the following select box:

```
What's roughly the size of
    <select name="area">
    <option value="size_of/california.txt">California</option>
    <option value="size_of/texas.txt">Texas</option>

    <option value="size_of/minnesota.txt">Minnesota</option>
    <option value="size_of/pennsylvania.txt">Pennsylvania</option>
    <option value="size_of/delaware.txt">Delaware</option>
    <option value="size_of/rhode_island.txt">Rhode Island</option>
    <option value="size_of/chile.txt">Chile</option>
    <option value="size_of/italy.txt">Italy</option>

    <option value="size_of/mexico.txt">Mexico</option>
    <option value="size_of/australia.txt">Australia</option>
    <option value="size_of/india.txt">India</option>
    </select>? Click on the map to find out.
```

As one example, the points file for California is at http://mappinghacks.com/projects/gmaps/size_of/california.txt and starts out like this:

```
-3.93254 4.72960
-0.58090 4.72519
-0.58141 1.73040
3.44982 -2.02319
4.08442 -2.96202
3.77594 -3.31706
3.77654 -3.71225
```

This data is processed with the `parsePolygonData( )` function:

```
function parsePolygonData (txt) {
    var rePoint = /^(\S+)\s+(\S+)$/igm;
    var match, pts = [];
    msg.innerHTML = "Parsing polygon data…";
    while (match = rePoint.exec(txt)) {
        var lon = parseFloat(match[1]),
            lat = parseFloat(match[2]);
```


Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

```
            pts.push(lon, lat);
        }
        msg.innerHTML = "Click anywhere on the map to continue.";
        return pts;
    }
```

This bit of code defines a regular expression to read the pairs of points as longitude and latitude, and push them onto an array called `pts[]`. Next, the `drawPolygon( )` function is called when a user clicks on the map:

```
    function drawPolygon (overlay, point) {
        msg.innerHTML = "Got click at " + point;
        if (point && shape) {
            var angle = parseInt(document.forms["selectPoly"].rotate.value);
            var poly = makePolygon( point, angle, shape );
            // msg.innerHTML = "Showing poly " + poly + " at " + point;
            map.centerAndZoom(point, 13);
            map.clearOverlays();
            map.addOverlay(poly);
        }
    }
```

The `drawPolygon( )` function takes the angle that you are applying to turn the state or country outline, and then calls `makePolygon ( )` to generate a polygon represents the state or country borders. Finally, the polygon is added to the map with `map.addOverlay( )`, and the map is recentered and zoomed in. The mathematical core of the code is in the `makePolygon( )` function.

```
    function makePolygon (center, angle, ptsIn) {
        var ptsOut = [];
        if (isNaN(angle))

        angle = 0;

        var theta = - angle * Math.PI / 180;
        for (var i = 0; i < ptsIn.length; i += 2) {
            var x = ptsIn[i] * Math.cos(theta)
              - ptsIn[i+1] * Math.sin(theta)
              + center.x;
            var y = ptsIn[i] / phi * Math.sin(theta)
              + ptsIn[i+1] * Math.cos(theta)
              + center.y;
            var pt = new GPoint(x, y);
            ptsOut.push
        }
        var poly = new GPolyline(ptsOut, 'ff0000', 3, 1);
        msg.innerHTML = "Generated polygon centered at " + center;
        return poly;
    }
```

The `makePolygon( )` function takes the requested angle, converts it from degrees to radians, and stores it in `theta`. The `x` and `y` coordinates are composed by scaling each point from the geographic outline by the horizontal and vertical components of `theta`, and then offsetting that point from the point where the user clicked on the map. These coordinates are used to create `GPoint` objects that are pushed onto an array, and then assembled into a `GPolyline`, which is returned and then displayed on the map by `drawPolygon()`.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006

3.13.3. How Big Is "That Big"?

We would hope that this tool helps provide a bit of spatial perspective to address a weakness in geographical literacy. Problems of cultural referents remain. For example, it is said that one difference between America and England is that Americans think 100 years is a long time, and the English think 100 miles is a long way. Perhaps the tool needs the addition of a set of radio buttons, modeled after a Slashdot poll.

When faced with a 100-mile trip, I will:

- Happily drive down in the morning and back that night
- Try to schedule it for the best time
- Go for a week-long trip
- Consider it an absurdly extreme ordeal
- The Bentley hardly gets up to speed in 100 miles
- Cowboy Neal/I'm on a home release program, you insensitive clod

Humor aside, calibrating perceived distances is an important task. The Jhai Project http://www.jhai.org/ is installing computers and Wi-Fi relays to allow people in remote villages to connect with each other and the world. Farmers can see what the current market prices are before deciding to take their goods to market, because a 10-mile trip can take hours, and the the difference between selling on a bad-price day and a good-price day has an enormous effect on the well-being of an extended family. By getting current market prices and dealing directly, farmers and craftspeople are able to double their income.

I can drive from Sebastopol, CA down to San Francisco International Airport, fly to Los Angeles, and *return* in less time than it takes to get to the nearest market town in the mountains of Laos. A mile (or kilometer) may be the same distance the world round, but it doesn't mean the same thing to me as it does to a Laotian farmer. Even though (for some people) our world is getting smaller every day, areas and distances still matter to humanity a great deal.

Google Maps Hacks By Schuyler Erle, Rich Gibson
ISBN: 0596101619 Publisher: O'Reilly Print Publication Date: 1/1/2006