ONLINE MAPPING SERVICE FOR FIRE SERVICE FIRST RESPONDERS USING GOOGLE MAPS API

Shunfu Hu (shu@siue.edu)
Department of Geography
Southern Illinois University Edwardsville
Edwardsville, IL 62026

1. INTRODUCTION

Recently there have been many interest in utilizing Google Maps to implement online mapping services (Johnston and Jensen, 2009; Peng and Wu, 2010; Roth and Ross, 2009; Chow, 2008; Pan et al, 2010; Peterson, 2012, Hu, 2012). That is because the launch of Google Maps in 2005 has revolutionized online mapping service applications on the Internet. Based on AJAX (Asynchronous JavaScript and XML), Google Maps introduces an innovative approach that maintains a constant connection between server and client, which results in a fast-response time (Peterson, 2008). In addition, Google also provides programmers free access to its code in the form of an Application Programming Interface (API). The API contains functional libraries (i.e., routines and functions) that can be called by a programmer using JavaScript or similar scripting language (Udell, 2009). The Google Maps version 3 supports both traditional desktop browser applications as well as mobile devices such as the Apple iPad and iPhone. It also supports all current web browsers. These features make Google Maps JavaScript API the most commonly used Maps API for online mapping (Peterson, 2012). There are some other Maps APIs, such as Yahoo! Maps API, Microsoft Bing Maps API, and ESRI ArcGIS API for JavaScript available for the development of online mapping (Hu, 2012).

The existing fire emergency service for a municipality relies on mobile GIS (geographic Information System) which requires server-side mapping applications and wireless data network, and in most part requires the cooperation among the fire department, GIS division (providing street centerlines, parcel data and routing, etc) and 911 Call Center (Luccio, 2010). Such a fire emergency service is still most popular and reliable, but it is no doubt very costly. Professional firefighters for municipalities use vehicles equipped with mobile data terminals (MDT) in which they are able to get access to the Internet. Therefore, it is a logical step to use the Google Maps to provide online mapping service in responding to fire emergencies. The purpose of this paper is to demonstrate the development of an alternative solution to provide cost-effective fire emergency service using Google Maps API. A case study for the city of Wood River, Illinois is demonstrated.

2. METHODOLOGY

2.1. DATA SET

Since the use of Google Maps is free of charge, the only data set that needs to be collected is the locations of the fire hydrants because they are not part of the Google Maps. The fire hydrant data set were provided by the Fire Department of Wood River, Illinois. The original data was in Environmental Systems Research Institute (ESRI) ArcView shapefile format with the spatial reference of State Plane Coordinate System west zone, North American Datum of 1983, and feet. The attribute table contains the following fields: hydrant_id (e.g., H-15-SW-001), municipality (e.g., Wood River), location (e.g., Hillview, Bottom of Hill), make (e.g.,Mueller), year (e.g., 1974), test date (e.g., 5/20/2011), and flow capacity (e.g., 750 gallons per minute). The shapefile was first projected to WGS 1984 Web Mercator (Auxiliary Sphere), which is a required spatial reference for use with Google Maps or any other Maps on the Internet. The projected shapefile was then converted into XML file format for use in the development of the online map service. For information about the tool to convert ArcView shapefile to XML, please refer to https://www.e-education.psu.edu/ geog863/l4_p3.html.

2.2. DATA PREPROCESSING

The converted XML file of the fire hydrant data can be edited/updated in Notepad++. Below is an example of the XML file (e.g., hydrants.xml) that contains the information for one fire hydrant with an ID of H-15-SW-001. For each hydrant, all relevant information is placed in a pair of <pt> and </pt> tags. There are over 450 pairs of such tags in the hydrant XML file.

```
        <hydrants>
        <pt hydrant_id=” H-15-SW-001” municipality =”Wood River” location="Hillview,
Bottom of Hill" make="Mueller" year="1974" test_date="/5/202011" gpm="747.126213"
color="2" x="-90.09211" y="38.87953" </pt>
        </hydrants>
```

Next, with JavaScript code, a programmer can load the xml file onto the Google Maps for display through the call of jQuery. jQuery is a JavaScript library for HTML scripting and is an open source software designed to create dynamic web pages (jQuery, 2011). To use jQuery, the programmer has to attach the external jQuery JavaScript file (e.g., jquery.js) to an XHTML page as follows:

```
        <script type="text/javascript"
        src="http://ajax.googleapis.com/ajax/libs/jquery/1.5.2/jquery.js"></script>
```

Where src links to Google's content distribution network (CDN) to load the jQuery core file. Then, jQuery.get function was used to load the XML file (e.g. hydrants.xml) at the initialization of the Google Maps (i.e., *function initialize()*) and jQuery(data).find(“pt”) to retrieve the information for each hydrant (notice the <pt> and </pt> tags mentioned above). A variable, xmldoc, was declared to withhold all the attribute information (e.g., ID, address, Make, Year, etc.) for each hydrant. A variable, latlng, was declared to withhold only the locational information (i.e. latitude (y) and longitude(x)). A variable, ID, was declared to withhold only the ID information of each fire hydrant. A few other variables were declared to withhold the information for the rest of the attributes as below:

```
        jQuery.get("hydrants.xml", {}, function(data) {
        jQuery(data).find("pt").each(function() {
        var xmldoc = jQuery(this);

                var latlng = new google.maps.LatLng(parseFloat(xmldoc.attr("y")),
                        parseFloat(xmldoc.attr("x")));
                var ID = xmldoc.attr("hydrant_id");
                var address = xmldoc.attr("location”);
                var city = xmldoc.attr("municipality");
                var zip = xmldoc.attr("zipcode");
                var Cindex = xmldoc.attr("color");
                var make = xmldoc.attr("make");
                var year = xmldoc.attr("year");
                var gpm = xmldoc.attr("gpm");
                var LastInspectionYear= xmldoc.attr("test_date");
        });
        });
```

2.3. DISPLAY OF THE FIRE HYDRANTS USING GOOGLE MAPS JAVASCRIPT API

Google Maps API can be easily linked to the web page with just a single XHTML element as below:

```
    <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false">
</script>
```

As you can see, this is a standard XHTML directive to include an external JavaScript file, served by maps.google.com. This element is added to the head section of the page. The next key step is to initialize the Google Maps API and load the map onto the page as follows.

```
    function initialize() {
        // Set the starting map viewport, based on center coordinates and zoom level
        var myLatlng = new google.maps.LatLng(38.86049, -90.07942);
        var myOptions = {
                zoom: 4,
                center: myLatlng,
                mapTypeId: google.maps.MapTypeId.ROADMAP
                      }
        // initialize the core map object
         var map = new google.maps.Map(document.getElementById("map_canvas"),
                              myOptions);

        // add markers to the map and notice the latllng variable declared in jQuery
        var marker = new google.maps.Marker(latlng);
        }
```

2.4. DEVELOPMENT OF SEARCHING AND FILTERING FUNCTIONS

It is required for this project to provide the user with search function such as Find Hydrants by Location so the fire service first responders can have the option to see fire hydrants around that location. Furthermore, search function needs to be accompanied by filtering the type of fire hydrants, a function called Filter by Type. The function, Find Hydrants by Location, was accomplished by using Google Maps API's Geocoding process, which converts an user-provided address (e.g., "501 E Edwardsville Road, Wood River, IL") into geographic coordinates (e.g., latitude 38.8685546, and longitude -90.0884099), which the programmer can use to place a marker or to position the map. The Google Maps API provides a geocoder class for geocoding addresses dynamically from user input. The programmer has to create a new geocoder object within the Google Maps *function initialize()* as follows:

```
        var geocoder = new google.maps.Geocoder();
```

and then create a new *codeaddress ()* function as follows:

```
        function codeAddress() {

                // Get the user's address input and the user's selection of hydrant type
                var address = document.getElementById("address").value;
                var selType = document.getElementById("select_type").value;

                // Google standard geocoding function
                 geocoder.geocode( { 'address': address}, function(results, status) {
                          if (status == google.maps.GeocoderStatus.OK) {
                          map.setCenter(results[0].geometry.location);
                        map.setZoom(11);
                          var marker = new google.maps.Marker({
                          map: map,
                          position: results[0].geometry.location
                    });
                  } else {
                  alert("Geocode was not successful for the following reason: " + status);
                          }
        });

        // Function for Filter by Type
                for (var i=0; i<markers.length; i++) {
                        if (selType == "0"){
                        markers[i].setVisible(true);
                        }
                        else if (selType != "0" & markers[i].mycategory == selType) {
                        markers[i].setVisible(true);
                        }
                        else {
                        markers[i].setVisible(false);
                        }
                }
                }
```

2.5. USE OF JAVASCRIPT AND CSS TO DESIGN THE LAYOUT OF THE WEB SITE

In the design of the web application, a single column layout design was adopted, which is mainly used for a map container. In the map container, the Google map or satellite imagery is displayed, and points of interest (e.g., fire hydrants) are marked with the customized fire hydrant markers – the red color for fire hydrants with a flow capacity of less than 500 gallons per minute (GPM), the orange color for fire hydrants with a flow capacity between 500 and 999 GPM, the yellow color for fire hydrants with a flow capacity between 1000 and 1499 GPM, and the cyan color for fire hydrants with a flow capacity greater than or equal to 1500 GPM.

User interaction with the map is achieved by adding standard Google Maps controls, such as Pan and Zoom controls; Map Scale control; and Map Type control: Satellite and

Roadmap. In addition, tooltips (e.g., hydrant ID) to the markers are provided, along with clickable marker icons with Google Maps API's standard Infowindow, which provides the information about each fire hydrant (i.e., ID, address, year and make, flow capacity, test date, etc.).

In addition, in order to allow the user to link the search function (e.g, Find Hydrant by Location) with the map display, the author decided to provide tab panels as a user interface. The Adobe's Spry 1.6 framework for Ajax was employed to develop such interface. To do so, the programmer has to download SpryTabbedPanels.css and SpryTabbedPanels.js (all open sources) from the Adobe Labs (Adobe. 2011). The former links the CSS style sheet that styles the tabbed panel; the latter links the Spry TabbedPanels JavaScript library with the search function. Both of them need to be placed in the head section as follows:

```
<link href="SpryTabbedPanels.css" rel="stylesheet" type="text/css" />
<script src="SpryTabbedPanels.js" type="text/javascript"></script>
```

Working in conjunction with the tab Find Hydrant by Location is the HTML <input> tag and input field that allow the user to type in address or zip code from the keyboard. The HTML code looks as below:

```
<input id="address"  size="40" type="textbox"
value="Please type your address or zipcode here">
```

Then, the <select> tag is used to create a select list (drop-down list) and the <option> tags inside the select element define the available options in the list. For instance, Filter by Type allows the user to choose one of the four hydrant types with 0 as a default value for no filtering. It was implemented with the following HTML code:

```
<select id = "select_type" name="hydrant_Type">
        <option value="0" selected>All Types</option>
        <option value="1">0 - 499 GPM</option>
        <option value="2">500 - 999 GPM</option>
        <option value="3">1000 - 1499 GPM</option>
        <option value="4">1500 GPM</option>
</select>
```

Finally, all the pieces need to come together. JavaScript is the scripting language for Google Maps. In addition, XHTML (eXtensible HTML) is the building block of Google Maps with Cascading Style Sheet (Udell, 2009). Therefore, JavaScript, XHTML, and CSS are used to develop all the functionalities, including uploading XML data file via jQuery, displaying points of hydrants using customized icons via Google Maps API, and the search function via Google Geocoderer. Figure 1 illustrates a conceptual framework for the integration of the Google Maps API and other JavaScript libraries in the World Wide Web environment.

3. RESULTS

The use of Google Maps API V3 provides very efficient way to disseminate digital cartographic information to the user on the Internet faster and more user friendly. With Google Maps standard Map Type control, the user is able to choose one of the two map types:

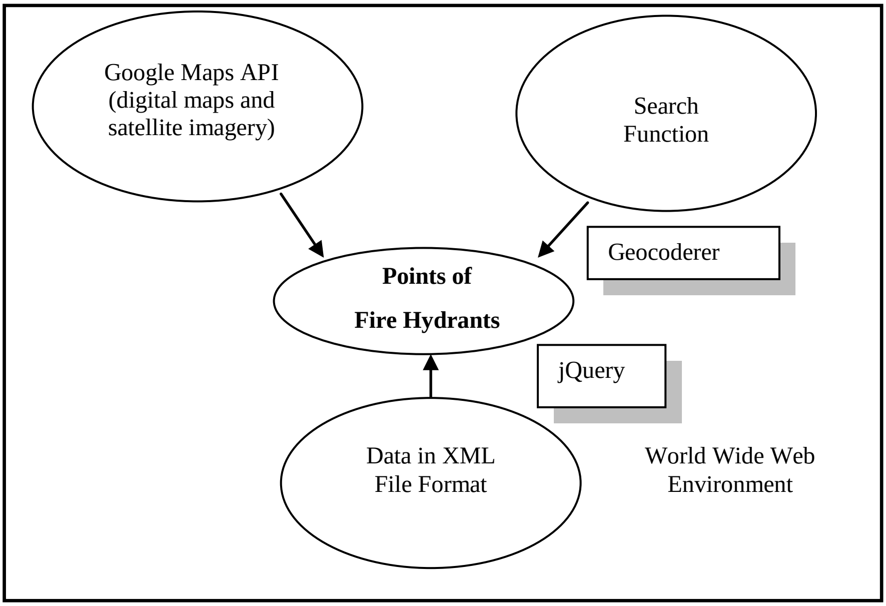
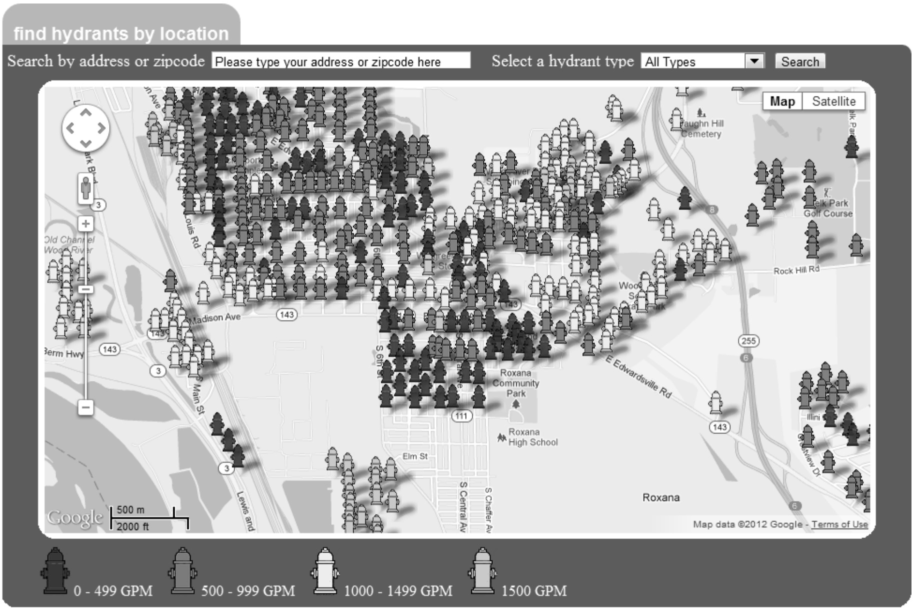


FIGURE 1
A CONCEPTUAL FRAMEWORK OF MAPPING FIRE HYDRANTS USING GOOGLE MAPS API THAT INCLUDES DATA UPLOADING FUNCTION, DISPLAYING FUNCTION AND SEARCH FUNCTION.

Roadmap and Satellite. Figure 2 shows an outlook of the online map service for the fire hydrant project in Google Chrome. At the initial launch of the web page, customized markers for the fire hydrants are displayed within the map container with a map legend below it. The Search function located above the map container allows the user to locate all fire hydrants by typing a street address in the Search box. If the user would like to display a particular type of the fire hydrants (e.g., 0-499 GPM), he/she can select that type from the drop-down list and click the Search button (Figure 3).

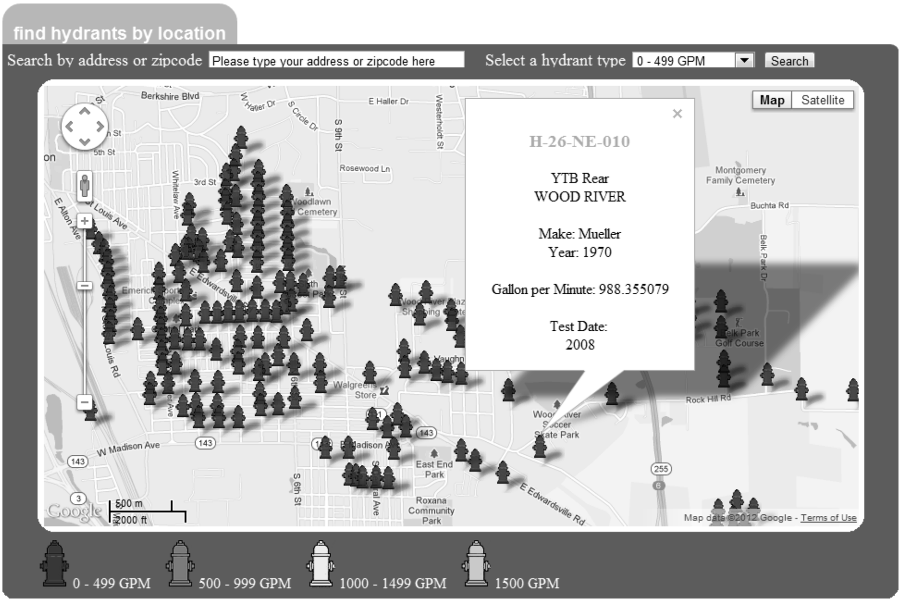
This online mapping service also provides routing capability. The default routing provides the driving directions, along with the guided turns and distances, on the map, from a fire department or fire house to a fire location (the address of a 911 caller). Similarly, if the fire truck is located anywhere other than at a fire department, the routing will start from the existing location of the fire truck (if it is equipped with MDT) (Figure 4).



find hydrants by location
Search by address or zipcode
Please type your address or zipcode here
Search
Map
Satellite
0 - 499 GPM
500 - 999 GPM
1000 - 1499 GPM
1500 GPM
Map data ©2012 Google - Terms of Use

Select a hydrant type: All Types

FIGURE 2
LOCATIONS OF ALL FIRE HYDRANTS WITH FOUR DIFFERENT TYPES



find hydrants by location
Search by address or zipcode
Please type your address or zipcode here
Search
Map
Satellite
0 - 499 GPM
500 - 999 GPM
1000 - 1499 GPM
1500 GPM
Map data ©2012 Google - Terms of Use

Select a hydrant type: 0 - 499 GPM

H-26-NE-010

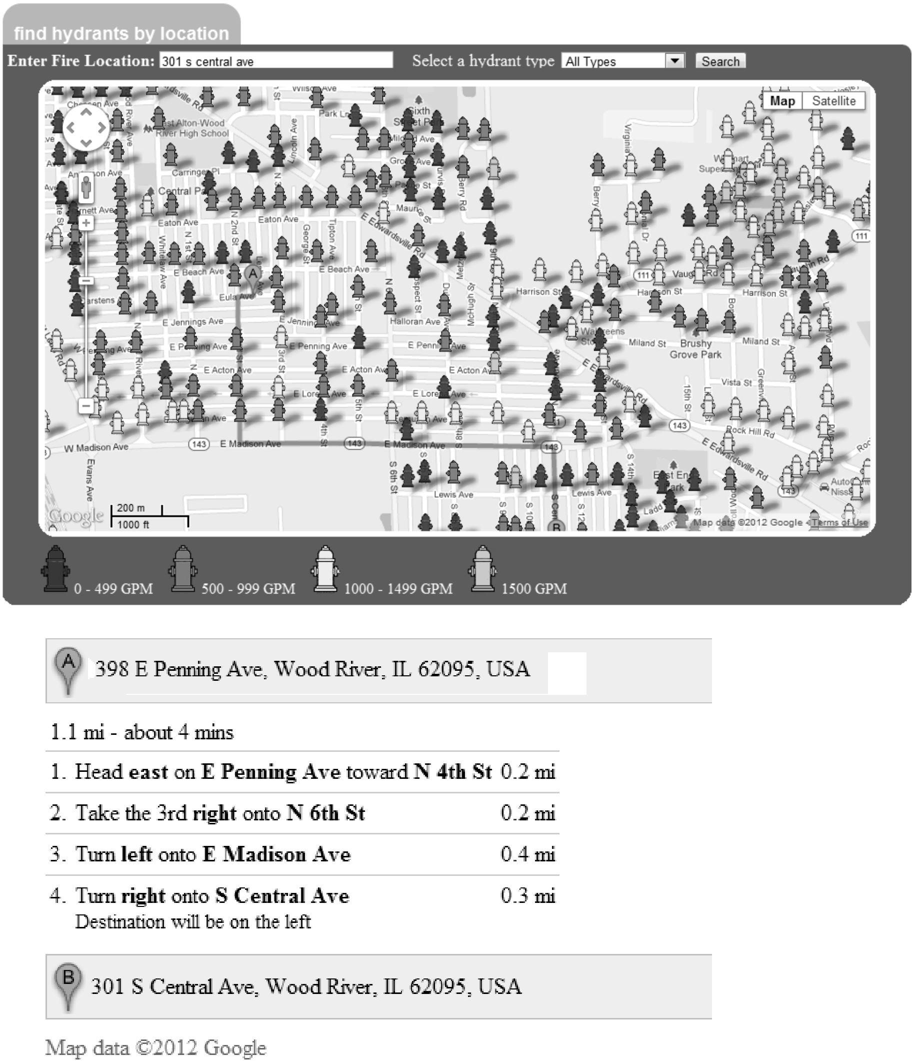
YTB Rear
WOOD RIVER

Make: Mueller
Year: 1970

Gallon per Minute: 988.355079

Test Date:
2008

FIGURE 3
THE USER CAN SELECT ONE HYDRANT TYPE; THE USER CAN ALSO CLICK ON THE ICON OF A HYDRANT TO GET RELEVANT INFORMATION.



find hydrants by location

Enter Fire Location: 301 s central ave

Select a hydrant type: All Types

A 398 E Penning Ave, Wood River, IL 62095, USA

1.1 mi - about 4 mins

1. Head **east** on **E Penning Ave** toward **N 4th St** 0.2 mi
2. Take the 3rd **right** onto **N 6th St** 0.2 mi
3. Turn **left** onto **E Madison Ave** 0.4 mi
4. Turn **right** onto **S Central Ave** 0.3 mi
   Destination will be on the left

B 301 S Central Ave, Wood River, IL 62095, USA

Map data ©2012 Google

FIGURE 4
GPS-GUIDED ROUTING FROM THE CURRENT LOCATION OF THE FIRE TRUCK (POINT A) TO A FIRE LOCATION (POINT B)

4. CONCLUSION AND DISCUSSION

This paper has demonstrated a cost-effective online mapping service for use by fire service first responders. It was successfully carried out using Google Maps API v3, XML, and Google Geocoding. The case study presented in this article provides the advanced functionality to display the locations of hundreds of fire hydrants on the Internet with customized icons and map legend. It also provides the sophisticated functionalities for searching, filtering, and tabbed interface that offer the user the capability to manipulate the data.

This online mapping service offers several benefits to the fire service first responders. First, the Search function allows the first responders or dispatchers to type in the address that a 911 caller provides and it will provide an instantaneous route from the fire department or a fire house, along with detailed turns and distances. They also can look for the closest fire hydrants around that address on the Google map before the fire truck arrives at the scene. If the fire truck is equipped with MDT, the fire responders in the fire truck can quickly find the route from the existing location of the fire truck to the fire location and can search for the closest fire hydrant before it arrives at the scene.

Secondly, in most cases, each fire district relies on its own district's parcel data, hydrant data, and road network data to respond a fire emergency call. However, professional firefighters outside the fire district will not have those data sets for a neighboring fire district (unless the two fire districts have a special agreement). This online mapping service would help professional firefighters outside the fire district to provide assistance when responding to a fire emergency call from an address outside their own district.

One additional benefit is that the online map service showing the locations of all the fire hydrants in a municipality gives the general public a tool to report any problem related to a fire hydrant such as leaking or damages.

The online mapping application for fire service first responders requires a constant access to the Internet. That means only the fire trucks that are equipped with MDT can use such online service. To remedy this issue, a better solution is to develop a similar online map service application for use on the Smartphone such as Adroid phones or iPhones.

5. REFERENCES

Adobe. 2011. Working with Spry 1.6. http://livedocs.adobe.com/en_US/Spry/SDG/help.html. Last accessed on May 10, 2011.

Bildirici, I. O., and N. N. Ulugtekin. 2010. Web mapping with Google Maps Mashups**:** overlaying geodata. *Proceedings of the Special Joint Symposium of ISPRS Technical Commission IV & AutoCarto in Conjunction With ASPRS/CaGIS 2010 Fall Specialty Conference*. Orlando, Florida. November 15-19, 2010.

Chow, T. E. 2008. The potential of Maps APIs for Internet GIS Applications, *Transactions in GIS* 12(2): 179-191.

Haubrock, S., T. Wittkopf, G. Grünthal, and D. Dransch. 2007. Community-made earthquake intensity maps using Google's API, *Proceedings of the 10th AGILE International Conference on Geographic Information Science*. Aalborg, Denmark, CD, 8.

Hu, S. 2012. Online Map Services Using Google Maps API and Open Source Methods. In: *Online Maps with APIs and WebServices*, pp. 265-278. M. M. Peterson, ed. Springer.

Johnston, L. R., and K. L. Jensen. 2009. MapHappy: A user-centered interface to library map collections via a Google maps "Mashup". *Journal of Map and Geography Libraries* 5(2): 114-130.

Jhnidk, X. 2010. Use PHP, MySql and Google Map API v3 for displaying data on map. http://tips4php.net/2010/10/use-php-mysql-and-google-map-api-v3-for-displaying-data-on-map/. Last accessed on May 10, 2011.

jQuery. 2011. The write less, do more, JavaScript library. http://jquery.com/. Last accessed on May 9, 2011.

Liu, S.B., and L. Palen. 2010. The new cartographers: crisis map mashups and the emergence of neogeographic practice, *Cartography and Geographic Information System* 37(1): 69-90.

Luccio, M. 2010. Mobile GIS helps firefighters respond more effectively. http://www.esri.com/news/arcwatch/0610/marietta-fire-dept.html. Last accessed on August 15, 2012.

Niccolai, J. 2008. So what is an enterprise mashup, anyway? http://www.pcworld.com/businesscenter/article/145039/so_what_is_an_enterprise_mashup_anyway.html. Last accessed on December 12, 2010.

Pan, B., J.C. Crottsa, and B. Mullerb. 2010. Developing web-based tourist information tools using Google Map. http://www.ota.cofc.edu/pan/PanCrottsMullerDevelopingGoogleMap.pdf. Last accessed on May 7, 2011.

Pejic, A., S. Pletl, and B. Pejic. 2009. An expert system for tourists using Google Maps API, *7th International Symposium on Intelligent Systems and Informatics*, SISY '09.

Peng, X., and X. Wu. 2010. Digital campus map publishing based on Google Map API, *Journal of Geomatics* 35(1): 25-27,

Peterson, M. P. 2008. International Perspectives on Maps and the Internet: An Introduction, In M. P. Peterson (Ed.), *International Perspectives on Maps and the Internet (pp. 3-10),* Springer.

Peterson, M. P. 2012. Online Mapping with APIs, *Online Maps with APIs and Mapservices* (M. P. Peterson, ed.), Springer, pp. 3-12.

Roth, R. E., and K. S. Ross. 2009. Extending the Google maps API for event animation mashups. *Cartographic Perspectives* 64: 21-31.

Scholefield, K. 2008. *Web based map services for scientific tourism: a case study of eighteenth and nineteenth century Edinburgh*. Master of Science Thesis, http://hdl.handle.net/1842/2475. Last accessed on May 9, 2012.

Udell, S. 2009. *Beginning Google Maps Mashups with Mapplets, KML, and GeoRSS*. New York, NY: Apress.

W3Schools. 2012. XML Basics and XML JavaScript http://www.w3schools.com/xml/xml_whatis.asp. Last accessed on May 9, 2012.