WEB MAPPING WITH GOOGLE MAPS MASHUPS: OVERLAYING GEODATA

I. O. Bildirici [a, *], N.N. Ulugtekin [b]

[a] Selcuk University, Faculty of Eng., Dept. of Geomatic Engineering, 42079 Selcuklu Konya, Turkey – bildirici@selcuk.edu.tr
[b] ITU, Civil Engineering Faculty, Dept. of Geomatic Engineering, 34469 Maslak Istanbul, Turkey - ulugtek@itu.edu.tr

Commission IV

KEY WORDS: Web cartography, Google Maps API, map mashups, JavaScript, XML, KML

ABSTRACT:

The term mashup is used for incorporating different web resources and information within a web site. Mashups are an integral part of Web2.0, which represents a variety of innovative resources, and ways of interacting with, or combining web content. Mashups are based on Application Programming Interfaces (APIs) that are online libraries of functions. Most of the APIs are available at no cost to web developers. Most common mashup applications involve web mapping or web cartography. There is a variety of API providers for map mashups, including Google, Yahoo and etc. The functionality of their APIs are similar, but the data content. Google provides a huge amount of geodata worldwide. With Google Maps API, web site developers can add dynamic maps to their pages, and can overlay their own point, line and polygon data on to the maps. Such data can be overlaid within the JavaScript code, from external XML or KML files. It is also possible to connect a database and overlay data from the database based on a query. In this study the basics for creating map mashups are introduced, then the ways of overlaying data are discussed and the usage of XML and KML files are focused on. KML overlays are not flexible and there are some limitations. XML overlays are more flexible, but there is no standard XML schema. Web developers can define their own elements and attributes, and develop their Javascript code accordingly. We propose an XML schema, which is full compatible with Google Maps API classes.

1. INTRODUCTION

The World Wide Web (WWW) is the most recent innovative medium to present and distribute spatial data. Here, the map plays a key role, and has multiple functions. Maps can play the traditional role of providing insight into geospatial patterns and relations. Under these circumstances maps are used as they would in e.g. an atlas or newspaper to present the structure of a city or the location of any disaster occurred lately. According to Kraak & Brown (2001); “Web cartography can be considered a trend in cartography”. There are also similar terms such as Internet Mapping, Online Mapping etc. This topic catches the attention of cartographic community and books and papers have been published. Some of the recent works are: Cartwright et al (2007), Peterson (2003).

Map mashups are one of the efficient tools for online mapping, or web cartography. They are, in general, an integral part of web2.0, which represents a variety of innovative resources, and ways of interacting with, or combining web content. This term is closely related to Tim O'Reilly because of the O'Reilly Media Web 2.0 conference in 2004 (URL1). Mashups are created with online libraries and functions called Application Programming Interfaces (API). Most of the APIs are available to web developers at no cost. There is a variety of mapping API providers including Google, Yahoo, etc. Google Maps API is the most common one of the APIs because of the rich data content for the whole world. Approximately same vector data and imagery are used for Google Maps and Google Earth.

With map mashups, dynamic maps can be added to third party web sites, and third part content can be overlaid on such maps. Such maps are becoming common in many sites involved in location, such as on-line auction and shopping websites, hotel booking sites etc. The most common usage is depicting a certain address on the map with a marker, mostly the upside-down rain drop. In this example, the map is brought from Google’s servers; the point (the marker) is overlaid by the developer of the web site.

This developments increase the spatial awareness of ordinary people. There is a huge amount of spatial data available on the net. The term “neogeography” has emerged to address a set of new geographic concerns with the rise of such enabling technologies as web mapping services and pervasive GPS-enabled devices. Liu and Palen (2010) discusses such issues and examines the usage of map mashups in the crises management.

In this paper, we fist introduce the Google Maps API Family, and then discuss the overlay possibilities in the third party

* Corresponding author.

websites. We examine KML and XML overlays, and propose an XML schema.

2. GOOGLE MAPS API FAMILY

Google Maps has a wide array of APIs that let web developers embed the robust functionality and everyday usefulness of Google Maps into their own websites and applications, and overlay their own data on top of them. The Google Maps API family consists of the following (URL 2):

Maps JavaScript API: This API was previously known as Google Maps API, with which a Google Map can be embedded in third party web pages using JavaScript. The map can be manipulated and appropriate content can be added through a variety of services. Two versions are available: V2 and V3. V2 is deprecated on May 19th, 2010; V3 is the current one. Since V3 is a rather new development, we use mostly V2 for our discussion in this paper. This member of the family will be mentioned as "Maps API" in the next sections of the paper.

Maps API for Flash: This ActionScript API is used to embed a Google Map in a Flash-based web page or applications.

Google Earth API: A true 3D digital globe can be embedded into a third party web page. With this API a mashup with the functionality of Google Earth can be provided for the visitors.

Static Maps API: A fast and simple Google Maps image in a web page or mobile site without requiring JavaScript or any dynamic page loading can be embedded.

Web Services: URL requests to access geocoding, directions, elevation can be realized.

Maps Data API: Map data through Google Data API feeds, using a model of features (placemarks, lines & shapes) and collections of features, can be viewed, stored and updated.

All these APIs are free services, available for any web site that is free to consumers. For Version 2 a key from Google is necessary, which can be obtained easily. For the current version (V3) no key is needed. Businesses that charge fees for access, track assets or build internal applications must use Google Maps API Premier, which provides enhanced features, technical support and a service-level agreement. The latter is not a free service.

Google Maps API Family is programmed with JavaScript, the most popular scripting language for developing dynamic web content. Web developers should also use JavaScript, whose use is free, in their web sites.

The datum of the map data is WGS 84. The default map projection is the Mercator projection. Due to increasing distortion towards poles, Polar Regions are not visible, so the world is shown between ~85 N and ~85 S latitudes. In terms of scale there are 18 zoom levels ranging from a map scaled ~1:5000 to a world map scaled ~1:250 million. For each level certain objects are selected and generalized accordingly. It is noticeable that there is a good text optimization.

A comprehensive summary about Google Maps and Maps API can be found in URL 3.

In order to use the Maps API, the API library must be declared in the head section of the HTML document. For V2:

```
<script src="http://maps.google.com/maps?file=api&amp;
v=2&amp;key=ABQIAAAAEm9eg8M7..."
type="text/javascript"></script>
```

For V3 (without key):

```
<meta name="viewport" content="initial-scale=1.0, user-
scalable=yes" />
<link href="http://code.google.com/apis/maps/
documentation/javascript/examples/standard.css"
rel="stylesheet" type="text/css" />
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false">
</script>
```

In the code above, there are some points to be mentioned. In the viewport declaration the option "user-scalable" enables the user to change the scale of the map. Otherwise zooming will be disabled. An important development in the Maps API V3 is the sensor parameter in the API declaration, with which the built-in GPS receivers can be reached from the API code, if the web browser is compatible. Since the web pages created by using Maps API V3 can be displayed in certain types of the mobile devices, Iphone and devices running Android operating system, the internal GPS receivers can easily be used within the web browsers without installing any software.

The functions calling Maps API functions are to be coded also in the head section of the HTML document. The map is displayed within a "div" tag in the body section. V2 code below displays a simple map with a marker on the centre:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html;
charset=utf-8"/>
 <title>Google Maps JavaScript API Example</title>
<script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;
key=..." type="text/javascript"></script>
<script type="text/javascript">
   function load() {
   if (GBrowserIsCompatible()) {
   var map = new GMap2(document.getElementById("map"));
   map.setCenter(new GLatLng(41.05, 29.05), 11);
   map.addControl(new GScaleControl());
   var customUI = map.getDefaultUI();
   map.setUI(customUI);
   var point=new GLatLng(41.05, 29.05);
   var marker=new GMarker(point);
   map.addOverlay(marker);
   GEvent.addListener(marker,"click", function() {
   var myHtml = "Center of the map"+point;
   map.openInfoWindowHtml(point, myHtml);
      });}
    }
</script>
</head>
<body onload="load()" onunload="GUnload()">
<div id="map" style="width: 400px; height: 400px">
</div>
</body>
</html>
```

In the head section the API library is declared within the first script tag. In the second script tag there is a function named load that will be invoked when page is loading. This is declared in the body tag. In this function: a map object is created from GMap2 instance; the centre and the scale (zoom level) of the map is declared; a bar scale is added; the user interface of the map is declared (the map type buttons and navigation tool); a point is created from the geographical coordinates; at that point a marker is created; and a “click” event for the marker is created causing an info window is displayed.

In the body section a div tag is declared, in which the map is displayed. Within the div tag dimensions of the map are determined.

The map created with this code is shown in Figure 1. The local language of the client has been recognized automatically, so the buttons up right are in Turkish.

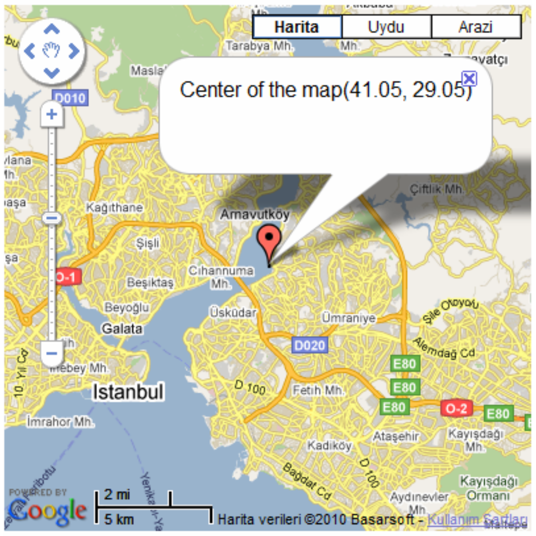


Figure 1. A simple map with a marker on the centre

3. OVERLAYS

Three types of spatial data can be overlaid on Google Maps, point, line, polygon, which are called markers, polylines and polygons in Google Map’s terminology. Image overlays are also possible. To determine the locations geographical coordinates (longitude and latitude) are used. The datum is WGS84. The map on the background consists of frames each downloaded from a different server. The overlays are combined with the background image (map) without downloading any image again. Because of this technique overlay process occurred in the client computer. So overlays are easy to be performed, and costs no extra time and download.

3.1 Overlays within the JavaScript Code

The simplest way of overlaying data is to do this within JavaScript code. In the example in Figure 1, a marker (point) is overlaid on the center point of the map. The most common situation is the following. In a website of a company (or a shop) there is a marker showing the address of the company on the map. Here a marker representing the address is overlaid on the map.

After creating a map the code below creates a marker and overlays it.

```
var marker=new GMarker(GLatLng(41.05, 29.05));
map.addOverlay(marker);
```

The function GLatLng creates a point. The parameters are latitude and longitude in decimal degrees. GMarker delivers a marker attached to this point. addOverlay method performs the overlay. Here a standard Google marker is shown (upside down rain drop). In order to use a symbol an icon is needed. The image format for icons is PNG with transparent background. Google provides a set of basic icon images for mapping purposes. There are also other web sites that publish free icon images.

Defining an icon is complex because of the number of different images that make up a single icon in the Maps API. At a minimum, an icon must define the foreground image, the size of type GSize, and an icon offset to position the icon.

The simplest icons are based on the G_DEFAULT_ICON type. Creating an icon based on this type enables to change the default icon by modifying only a few properties.

In the code below, an icon using the G_DEFAULT_ICON type is defined, and then modified to use a different image.

```
var point=new GLatLng(41.11, 29.02);
var myIcon=new GIcon(G_DEFAULT_ICON);
myIcon.image="http://atlas.selcuk.edu.tr/maps/icons/uni
versity.png";
myIcon.iconSize=new GSize(24,24);;
markerOptions = { icon:myIcon };
var marker=new GMarker(point,markerOptions);
map.addOverlay(marker);
```

Similarly polylines (Gpolyline) and polygons (GPolygon) are also created, and overlaid. The code below creates a simple polyline passing through 5 random points around the centre of the map. The result is shown in the figure 2. The second and the third parameters of the GPolyline are the color and the thickness of the polyline. Geodesic polylines are also possible.

```
var points=new Array();
for(var i=0;i<4;i++)
{
    points[i]=new GLatLng (0.1*Math.random()
+41.05,0.1*Math.random()+29.05);
}
var poly=new GPolyline(points,"#ff0000", 2);
map.addOverlay(poly);
```

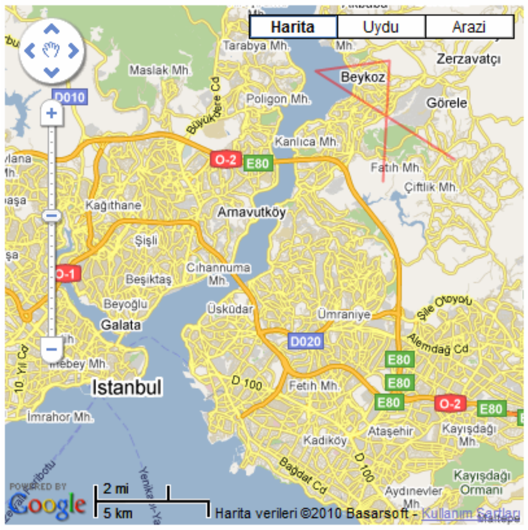


Figure 2. A simple polyline

The class GPolygon is very similar to GPolyline class.

This approach is useful if few objects are to be overlaid. Otherwise KML or XML files should be used.

3.2 KML Overlays

The content of the KML files can be overlaid on the Google Maps. The whole file is overlaid once. This can be thought as an individual layer. The file must be located on a valid URL address.

Since the KML objects and Maps API objects (or classes) are different, all attributes of the KML objects can not supported. So the appearance of KML contents may not be the same in the Google Earth and the Maps API. In Maps API V2, some attributes such as icon sizes of markers, line thicknesses and colors of polylines and polygons are ignored when overlaid. Such problems are mostly solved in the Maps API V3.

There are still some problems:

- All content is displayed as a separate layer at once.
- Only two attributes, name and description, are available. Therefore creation of a legend is not possible, and making queries based on attributes or thematic representations are limited.

The advantage of this approach is that a variety of GIS software can export to KML file format.

The V2 code below displays a KML content.

```
gx1 = new GGeoXml("http://www.example.com/test.kml");
map.addOverlay(gx1);
```

Above, for each object in the KML file an info window, triggered by clicking, is created. In the info window the name and description attributes of the object that is clicked are displayed.

In Maps API V3, a KML layer is defined, which is attached to a map.

```
var map = new google.maps.Map(
    document.getElementById("map_canvas"),
    myOptions);

var nyLayer = new google.maps.KmlLayer(
   'http://www.example.com/example.kml',
      {
       preserveViewport:true,
       suppressInfoWindows:true,
       map: map});
```

3.3 XML Overlays

When many objects are to be overlaid, using XML files is another alternative. There is not any certain XML schema suggested by Google. Instead, Maps API provide efficient tools to parse XML files. So the developers define their own XML schema and create JavaScript code accordingly.

Since this approach is very flexible, automatic legend creation and thematic representations on certain attributes are possible. Another advantage is the simple structure of XML files. By using a text or XML editor, data can be updated without changing JavaScript code. People without programming skills can easily do this.

The authors suggest following schema for XML data. The main element is markers. Under markers three elements can be defined for point, line and area objects (or geometries):

- marker
- polyline (includes point elements)
- polygon (includes point elements)

For each of these elements certain attributes are needed, which match properties and options of GMarker, GPolyline and GPolygon classes (for more information see URL 4). In table 1, 2, 3 and 4 the attributes for marker, point (child element of polyline and polygon), polyline and polygon elements are given.

According to specifications mentioned above an XML file looks like as follows:

Attribute	Meaning
Id*	Identifier
lat	Latitude in decimal degrees
long	Longitude in decimal degrees
name	Name of the point
description	The HTML content to be displayed in the info window
cat	Category or class required for legend
ico_wsize	Icon width in pixels
ico_hsize	Icon height
ico_wssize*	Shadow width
ico_hssize*	Shadow height
ico_icon	Icon image file name
ico_shadow*	Shadow image file
a01,a02,…*	Additional attributes for thematic

	representations

Table 1. Attributes of marker element, optional ones are marked with *

Attribute	Meaning
Lat	Latitude in decimal degrees
Long	Longitude in decimal degrees

Table 2. Attributes of point element (child element of the elements polyline and polygon)

Attribute	Meaning
id	Identifier
name	Name
description	The HTML content to be displayed in the info window
cat*	Category or class required for legend
cat_img*	Icon file to be used for legend
color	Line color
weight	Line weight
opacity	Opacity or transparency
options	Options for GPolyline class
a01,a02,…*	Additional attributes for thematic representations

Table 3. Attributes of polyline element, optional ones are marked with *

Attribute	Meaning
id*	Identifier
name	Name of the point
description	The HTML content to be displayed in the info window
cat*	The legend category or class
cat_img*	Icon file to be used for legend
scolor	Stroke color
sweight	Stroke weight
sopacity	Stroke opacity
fcolor	Fill color
fopacity	Fill opacity
Options*	Options for GPolygon class
a01,a02,…*	Additional attributes for thematic representations

Table 4. Attributes of polygon element, optional ones are marked with *

```
  <?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<markers>
  <marker lat="37.949615" long="32.496411" name="M1
Shopping Mall" cat="Mall" desc="…" ico_wsize="20"
ico_hsize="20" ico_wssize="20" ico_hssize="20"
ico_icon="icons/supermarket.png"/>

<Polyline name="tram" desc="…" color="#ff0000"
weight="5" opacity="0.5" options="">
    <point lat="37.88913" long="32.49349" />
    <point lat="37.92008" long="32.4961"  />
    <point lat="38.02723" long="32.510684"/>
  </polyline>
</markers>
```

In order to parse an XML file GXml class (V2) is used. Following code creates markers from an XML file. A click event for each marker is also added to invoke an info window.

```
GDownloadUrl("konya1.xml", function(data, responseCode)
{
var xml = GXml.parse(data);
var markers =
xml.documentElement.getElementsByTagName("marker");
for (var i = 0; i < markers.length; i++) {
   var pnt = new
GLatLng(parseFloat(markers[i].getAttribute("lat")),
         parseFloat(markers[i].getAttribute("long")));
var icon = new GIcon();
icon.iconSize = new
GSize(markers[i].getAttribute("ico_wsize"),
markers[i].getAttribute("ico_hsize"));
icon.iconAnchor = new GPoint(12, 16);
icon.image = markers[i].getAttribute("ico_icon");
var msg=markers[i].getAttribute("desc");
mrk[i]=new GMarker(pnt,icon);
GEvent.addListener(mrk[i], "click", function() { var
myHtml = msg;
map.openInfoWindowHtml(point, myHtml);  });
}});
```

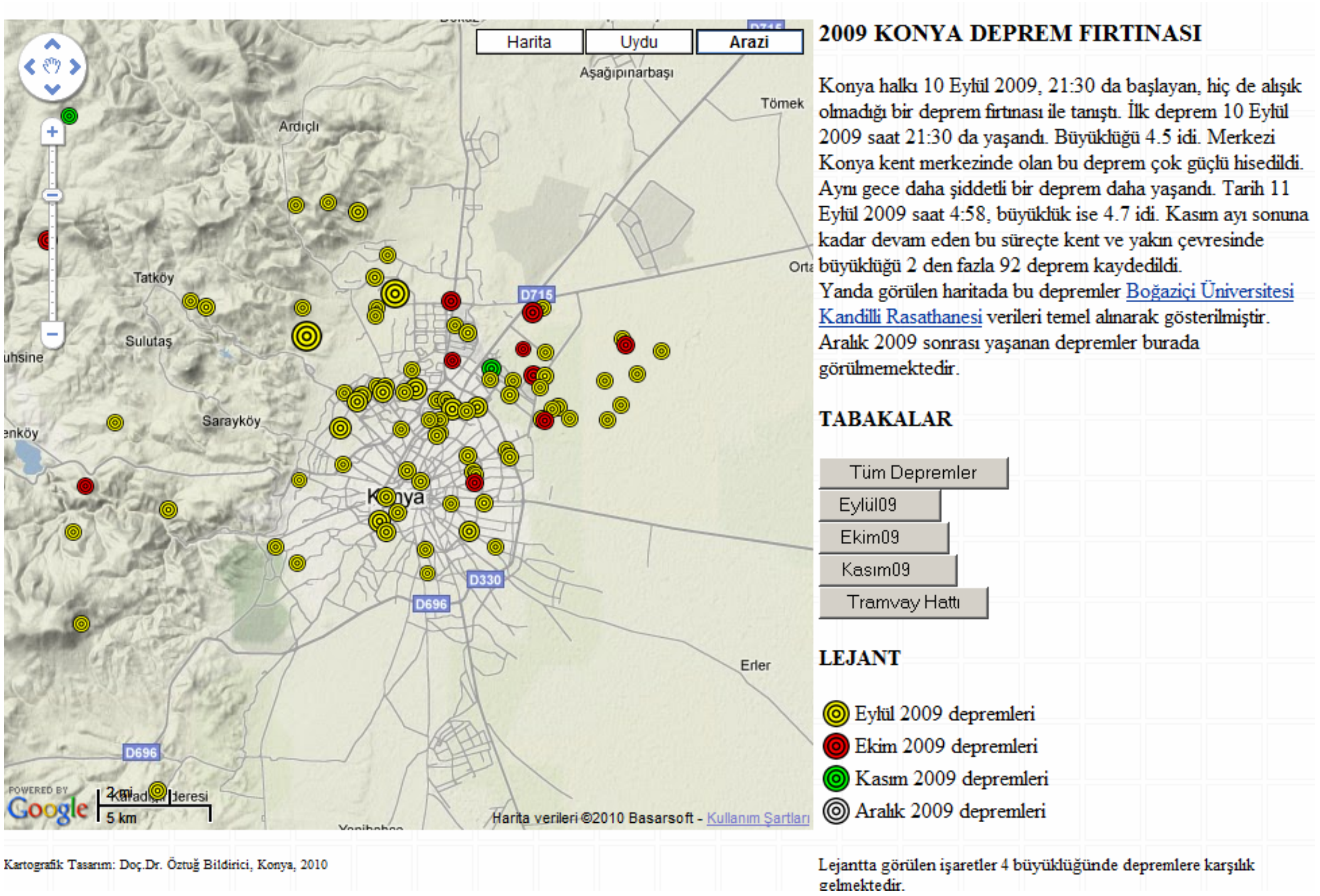
The code is somehow complicated, but flexible in terms of visualizing the XML content. Additionally, the XML content can be updated without touching the code. Furthermore a number of attributes can be used, while there are only two attributes in KML files (name and description).

GXml class does not exist in the Maps API V3, unfortunately. Such a class may be added in the future. XML parsing like V2 still possible by using a JavaScript code available at URL 5. A sample map using this code can be seen at URL 6.

3.3.1 Applications

By using XML overlay methodology explained here two maps were created: Earthquake map of Konya and city map of Konya (Turkey).

In Konya, approximately 90 earthquakes greater than two in magnitude occurred in two months, September and October 2009. Since earthquakes are rare in this region, there was a great panic in the society. The earthquake data are taken from the website of Kandilli Observatory and Earthquake Research Institute (URL 7). Earthquake data includes date and time, geographical coordinates in WGS84 datum, magnitude, depth, and place name. The authors developed a program that converts earthquake data into XML format. The attributes such as date and time, magnitude are taken as additional attributes, with which thematic representations are created. The sizes of icons depend on the magnitude, the colors on the month, in which the earthquake occurred. By using buttons users can see all the earthquakes, or filter them according to the month (September, October, and November). The default view is a physical map. Users can switch to the road map or satellite image. Some information about the topic is also added at the right side. The appearance of the website is shown in figure 3.



2009 KONYA DEPREM FIRTINASI

Konya halkı 10 Eylül 2009, 21:30 da başlayan, hiç de alışık olmadığı bir deprem fırtınası ile tanıştı. İlk deprem 10 Eylül 2009 saat 21:30 da yaşandı. Büyüklüğü 4.5 idi. Merkezi Konya kent merkezinde olan bu deprem çok güçlü hisedildi. Aynı gece daha şiddetli bir deprem daha yaşandı. Tarih 11 Eylül 2009 saat 4:58, büyüklük ise 4.7 idi. Kasım ayı sonuna kadar devam eden bu süreçte kent ve yakın çevresinde büyüklüğü 2 den fazla 92 deprem kaydedildi.
Yanda görülen haritada bu depremler Boğaziçi Üniversitesi Kandilli Rasathanesi verileri temel alınarak gösterilmiştir. Aralık 2009 sonrası yaşanan depremler burada görülmemektedir.

TABAKALAR

Tüm Depremler
Eylül09
Ekim09
Kasım09
Tramvay Hattı

LEJANT

◎ Eylül 2009 depremleri
◎ Ekim 2009 depremleri
◎ Kasım 2009 depremleri
◎ Aralık 2009 depremleri

Kartografik Tasarım: Doç.Dr. Öztuğ Bildirici, Konya, 2010

Lejantta görülen işaretler 4 büyüklüğünde depremlere karşılık gelmektedir.

Figure 3: Konya earthquake map (http://atlas.selcuk.edu.tr/maps)

Another application of XML overlays is the city map of Konya, which is crated for visitors of the Department of Geomatics Engineering of Selcuk University. For this purpose a number of points of interests (POI) were determined and an XML file was created. The POIs are divided into 10 classes for the legend, which is created automatically based on these classes. For each class custom icons are defined, and for each POI an info window triggered by clicking the item is defined, in which the name and description attributes are displayed. The description attribute contains HTML code with links and images about the POI. This HTML code in the info window is displayed, in which users can see images about the POI and clickable web site links. E.g., in the info window of a hotel, the image of the hotel and a link to the hotel's website are displayed.

All POIs are listed in a drop-down list box at the right side. When an item in the list is selected, the item is centered on the map with a circle around. The content of the drop-down list is filled automatically from the XML file.

Since the tram line connecting Selcuk University Campus and the downtown is not shown on the Google Map, an XML file is prepared and shown on the map when clicking the "tram" button. After second click the line is removed. Similarly the POIs (places) can be on and off. On the top of the page, and below the legend there is a link to Campus map where the location of the department and more details of the campus area are displayed. In this map, a detailed layer of buildings, the tram line and stops in the campus area, the entrance to the Faculty, in which the Geomatics Engineering Department is located, and the entrance to the convention center are shown as overlays.

The appearance of this website is shown in figure 4. These maps mentioned above and similar maps can be found in URL 8.

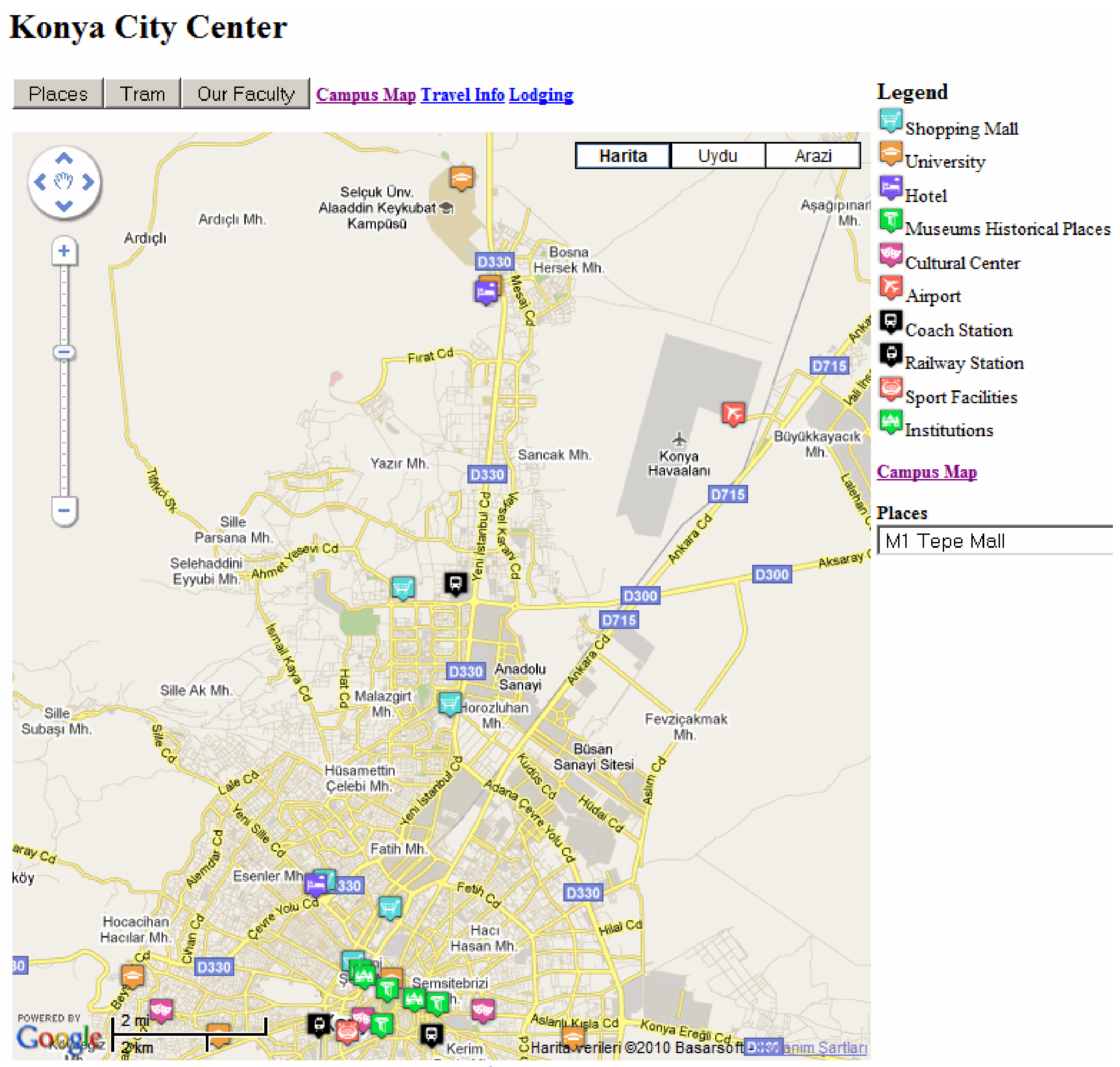


Figure 4: The city map of Konya (http://atlas.selcuk.edu.tr/maps/)

3.4 Overlays from a Database

Spatial data can be retrieved from a Geodatabase, and overlaid on Google Maps. Doing so, large volumes of data can be overlaid. After retrieving data, appropriate JavaScript code is created on the fly, and sent to client computers. The code run on the server can be written with any programming language, e.g. ASP, PHP etc. In this way commercial GIS software and Maps API can be combined.

4. CONCLUSION

In the map mashups created by using any mapping API, at least one point (a marker) is overlaid. The most common purpose of map mashups is to show the location of any business, say the location of a company, a hotel, a convention center, etc. If few items are be overlaid, it is realized within the Javascript code. In other cases data should be taken from a file or a database. Maps API supports KML overlays. In KML files only two attributes are available (name and description). Therefore thematic representations are limited to these attributes. XML overlays are more flexible, and more attributes can be used. The advantage of KML files is that these files can be created by Google Earth. Additionally some of the GIS software can export to KML format. XML files should be created manually, and edited by using text editors such as Notepad. Specific computer programs can be developed to create XML files from a variety of GIS data files. The authors also developed such programs for certain applications. Another alternative is of course getting data from a geodatabase and cerate Javascript code on the fly.

For XML overlays, there is no standard schema. We propose XML schemas for point, line and area objects. These schemas include attributes required for optimal representation of markers, polylines and polygons. The methods and properties of GMarker, Gpolyline and GPolygon classes are taken into account. Similar schemas and applications can be found on the net. A more comprehensible and standardized XML schema can be formed in the future. It will be feasible that such a schema will be compatible with Google's and other map APIs.

5. REFERENCES

Cartwright, W., Peterson, M.P., Gartner, G., 2007, Multimedia Cartography, Second Edition, Springer, Berlin, Heidelberg, New York.

Kraak, M.J., Brown, A., 2001, Web Cartography, Taylor & Francis, London.

Liu, S. B., Palen, L., 2010. The New Cartographers: Crisis Map Mashups and the Emergence of Neogeographic Practice, *Cartography and Geographic Information Science*, Vol. 37, No. 1, 2010, pp. 69-90.

Peterson, M.P. (ed), 2003, Maps and the Internet, Elsevier Science.

URL1: http://en.wikipedia.org/wiki/Web_2.0, accessed 30 August 2010.

URL 2: http://code.google.com/intl/tr-TR/apis/maps/, accessed 30 August 2010.

URL 3: http://en.wikipedia.org/wiki/Google_Maps, accessed 30 August 2010.

URL 4: http://code.google.com/intl/tr/apis/maps/documentation/javascript/v2/reference.html, accessed 30 August 2010.

URL 5: http://photomunchers.appspot.com/js/util.js, accessed 30 August 2010.

URL 6: http://gmaps-samples-v3.googlecode.com/svn/trunk/xmlparsing/downloadurl.html, accessed 30 August 2010.

URL 7: http://www.koeri.boun.edu.tr/scripts/lasteq.asp, accessed 30 August 2010.

URL 8: http://atlas.selcuk.edu.tr/maps