Looking @ Google Maps

Joey van der Bie
Media Technology programme
Leiden University
joey@vanderbie.net

Maarten van der Mark
Media Technology programme
Leiden University
mvdmark@gmail.com

Vincent Vijn
Media Technology programme
Leiden University
vincent@vijn.me

ABSTRACT

In a shot time Google Maps has become the facto standard for online map services. We give an overview of how Google Maps works and is used.

1. PURPOSE, CONTEXT AND HISTORY

Google Maps is web-based mapping service and application. On the website http://maps.google.com users can view maps of almost any place on earth on different zoom levels. People can choose between maps and satellite/aerial images or a combination of both. The service also incorporates other information like business addresses and public transport information. Besides this maps application it is also possible to integrate maps on your webpage and write applications for other purposes using the extensive Google Maps Javascript API that is freely available.

Google Maps (until 2005 Google Local) was originally developed by the company Where2 which was acquired by google in 2004 [1]. In 2005 Google Maps was converted from plate Carrée to Mercator projection [1]. Traditionally mapping applications on the web would render GIS (Geographic information systems) information on an empty map as the user requested that specific area of the map. Google maps broke with that tradition and pre rendered all maps and serves them as small tiles when needed. This requires less processing in the server side and hence it can deliver the content quicker. Google's map technology has been highly influential as most other mapping services in the internet now uses a similar technique. The KML file format for storing geospatial information developed by Google has become a standard [3] and is used by others as well.

The maps application and accompanying API is constantly improved and enhanced. New features are added from time to time, like the support for bicycle paths last month and more and more features aimed at giving localized information about shops and events. When maps was first released users had to enter locations by there exact geographical coordinates, but the API now supports geocoding that allow street addresses to be translated to coordinates[1][4].

2. OPERATING PRINCIPLES

Although Google Maps is freely accessible, it is proprietary software. There is an user license [5], which dictates the terms of use, and the code and the architecture are kept a secret. However the client-side is written in Java-Script and can be analyzed, also Google explains the workings of the client on their documentation page [7]. The server side still is kept a secret but looking at other GIS [6] the architecture can be guessed.

Most likely Google Maps architecture looks like this:
1. a component which has all the maps
2. a component which has all the geodata
3. a component which has all the tiles
4. a caching component which caches the data fed to the client
5. a client which can be viewed in the browser

Note that components 1 till 4 can be combined and separated over multiple servers. The caching component is only for speed improvements and exists of caching components for each other component so for our explanations of the workings we will leave that out. This leaves us with 4 components: maps, geodata, tiles and a client.

The flow

1. When the client is started it requests tiles and geodata from Google Maps
2. The tile component feeds the tiles to the client
3. The geodata component feeds the geodata to the client
4. When the user starts working with the client new requests are done for new tiles and geodata

The client

The the client runs in the web browser of the user. It is powered by the JavaScript that Google provides, and takes care of displaying the map tiles and all extra information (popup's, routing instructions etc.). It also handles all user interaction (dragging the map, zooming) and communicates with the Google Maps API.

The geodata component

The geodata component contains Points of Interest (POI's are literally points of interest and could be anything from a house to a bus stop) which are created by users, bought and collected by Google through searching websites.

Next Google Maps has the ability to draw small vector shapes. These shapes are also stored in this component. Last street and area information is stored.

The maps and tile components

For the tile component to be able to send tiles it first has to generate these from the maps component. The maps component has all kind of map data stored as images, vectors, streets and Points of Interest. Some data is bought from cartographers like TeleAtlas or Navteq, other data is created by Google. All the info is combined into one image and cut in separate tiles. Since zooming on an image isn't practical Google Maps uses zoom levels. The amount of tiles in a level increases by 4. So level 1 is 4 tiles, level 2 is 16, level 3 is 64, etc.

For each zoom level an image needs to be generated and cut in tiles. Generating of tiles (especially the detailed zoom levels) take much time and processing power therefore the tile component stores the tiles. Regeneration of the tiles normally only happens when changes are made to the map.

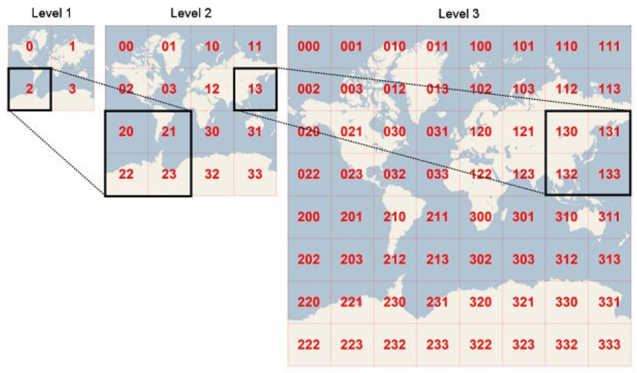


Figure 1: Map slices for different zoom levels
http://i.msdn.microsoft.com/dynimg/IC96238.jpg

Google Maps uses their own standard for structuring their tiles [7],though they differ only by index from the other big GIS [7][8][9]. Google Maps and most other GIS (like Bing Maps and Open Street Maps) use the Mercator projection for mapping their tiles [10]. Mercator projection treats the earth as a sphere, while normally you would threat the earth as an ellipsoid. This difference affects calculations done based on treating the map as a flat plane. Projections in GIS are referred to by their “EPSG” codes, identifiers managed by the International Association of Oil & Gas Producers (OGP) formerly known as the the European Petroleum Survey Group [11]. Spherical Mercator is described in EPSG:3785 [12] which replaces EPSG:900913. Taking that the information the GIS supply don't differ that much from eight other tiles from different GIS can be combined easily.

3. STRENGTHS AND WEAKNESSES

Google Maps is a free and flexible solution. Google’s infrastructure of servers and their connections is solid and fast so including a map doesn’t slow down the loading of your webpage too much. Via the API you can configure a whole range of options, including adding markers, area’s, info windows or even your own map overlay. The newest version of Google Maps is specially optimised to run on the PC but also on mobile devices. There’s also a Flash version available if you have a Flash based website. The interface works fluently for the users.

There are off course also some downsides to using Google Maps. The map is Google branded and the therms of service require you to only use their maps in public web applications. You can’t use the map images for other purposes such as print documents or your own programs, or use it for turn-by-turn navigation or controlling vehicles automatically. You also can’t use the routing or geocoding functions for uses other than displaying them on a map.

Google Maps is also a closed and proprietary platform. You have to use the map data per Google’s therms, but you also can’t adjust the maps or correct errors. Open projects such as OpenStreetMaps don’t have such limitations. Open-StreetMaps also allows you to send in corrections and since it’s open data, you can use the maps in every way you like. OpenStreetMaps only has a limited API (you can include a map and fixed markers, but it’s not as extensive as Google Maps), but other services built upon Open-StreetMaps do provide more options, such as CloudMade.

Some other map services include Yahoo! Maps, MapQuest and Bing Maps. A more extensive list of mapping services is available at [13]. These services offer more or less the same features in therms of functions and map quality, but Bing and Google offer also high quality areal images combining satellite images with photos taken with airplanes, so you can also zoom in.

4. INTENDED APPLICATIONS

Intended applications include showing one or more locations on a map or show a certain route. A lot of applications combine a certain dataset and a map, to show recent crime reports, police calls or Flickr photos that contain GPS coordinates. There’s also Google City Tours, that automatically suggests a route trough a city of choice using their database of POI’s and their navigation system [15].

One of the coolest and oldest application (dating back from 2006) used to be Treinvizier. It combined the train schedule and up-to-date train delays to calculate the current position of all trains in The Netherlands and showed them moving on a map. Unfortunately, the author didn’t have time to update the system to the new Google Maps API and the new train planner, so the site isn't online anymore.

5. UNINTENDED APPLICATIONS

Some more unexpected use is for example detecting earthquakes. A US governmental service combined Twitter messages about earthquakes that contain geo information or a placename with Google Maps to show recent earthquake reports [16].

There are also some artistic applications. Some people just look for nice aerial photos [17], but others try to involve into the maps themselves. Aram Bartholl made a real life version of the default placemarker in Google Maps [18] and Helmut Smits burned down some grass to eventually create a death pixel in Google's aerial photos [19].

The fact that Google supplies pretty high quality aerial imaging for free also gives interesting opportunities, such as finding craters [20] or missing airplanes [21].

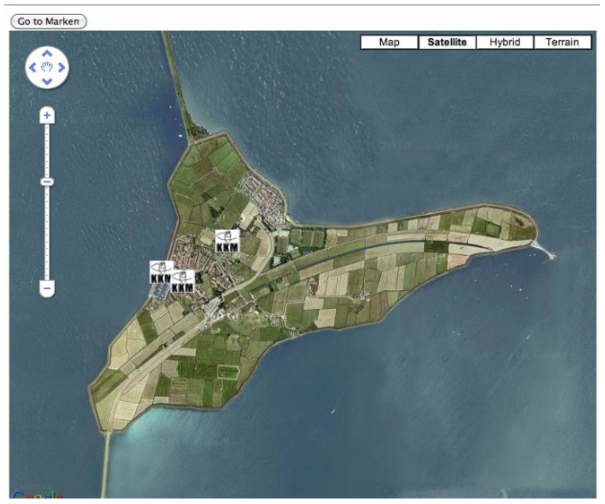
6. GETTING STARTED

The google maps v3 API is very well documented on googles website. The API reference and beginners tutorial can be found here:

http://code.google.com/apis/maps/documentation/javascript

For the report we did a simple implementation of a Google Maps application for an art festival. The festival called "Kunst in Zicht" is situated on the Marken island near Amsterdam and art installations can be seen on various geographic locations on the isle.

The web application we made implements a Google map that by default centers on the Marken Island. On the map we placed several markers indicating certain installations and events. The markers are indicated by the logo of the festival. If you click on one of the makers a popup-balloon gives you more information about the marker. Furthermore we implemented a feature that lets your view move to where you click on the map. A button on the top of the page will reset the view and bring you back to Marken.



See attachment 1 for the How-to.

The complete source code can be viewed at http://pastebin.com/V36HFhZU or in the attachment.

7. FINAL THOUGHTS

Google Maps is a powerful tool, and surely will be among us for a while. Though Google Maps is easy to use on multiple platforms, its license is limiting at best. Therefore we hope the dominant position of tiled maps servers will be overtaken by OpenStreetMaps, an Open Source platform which has a very open license, is also accessible on almost all platforms and is easy to edit.

REFERENCES

[1] Google Maps on Wikipedia, taken at 2010-06-03, http://en.wikipedia.org/wiki/Google_Maps
[2] Review of Google Maps http://all-things-spatial.blogspot.com/2009/06/ingenuity-of-google-map-architecture.html
[3] New layers for Google Maps, http://googlegeodevelopers.blogspot.com/2010/05/kml-traffic-and-bicycling-layers-come.html
[4] Geocoding on Wikipedia taken at 2010-06-03, http://en.wikipedia.org/wiki/Geocoding
[5] Google Maps User Terms, http://www.google.com/intl/en_en/help/terms_maps.html
[6] http://opengeo.org/publications/opengeo-architecture/
[7] Google Maps API version 3documentation http://code.google.com/intl/nl/apis/maps/documentation/javascript/overlays.html#CustomMapTypes
[8] OSGEO Tile Map Service Specification, http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification
[9]Bing Maps API specification, http://msdn.microsoft.com/en-us/library/bb259689.aspx
[10]Spherical Mercator projection, http://docs.openlayers.org/library/spherical_mercator.html
[11] Internatial Association of Oil & Gas Producers, http://www.epsg.org/
[12] EPGS 3857 standard, http://www.epsg-registry.org/report.htm?type=selection&entity=urn:ogc:def:crs:EPSG::3857&reportDetail=short&style=urn:uuid:report-style:default-with-code&style_name=OGP%20Default%20With%20Code&title=EPSG:3857
[13] http://www.programmableweb.com/apis/directory/1?apicat=Mapping
[14] Tile information about different tile servers, http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection/
[15] Google Maps City tours, http://citytours.googlelabs.com/search
[16] Earthquake detector using Tiwtter and Google Maps http://gislounge.com/usgs-twitter-earthquake-detector/

[17] Art in the world from Google Maps, http://pongsocket.com/earthart
[18] Placing a real POI in the world, http://www.datenform.de/map.html
[19] Creating a dead pixel in Google Maps/Earth, http://helmutsmits.nl/public-spaces/dead-pixel-in-google-earth
[20] Finding craters using Google Maps, http://www.gearthblog.com/blog/archives/2005/10/meteor_craters.html
[21] Finding Steve Fosset using Google Maps from Wikipedia, taken at 2010-06-03, http://en.wikipedia.org/wiki/Steve_Fossett#Disappearance_and_search

Attachment 1

How to implement:

In short this is what you need to do to create this application.

1. First create a empty html document.
2. In the *<head>* section add the following line to include the maps javascript API.

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

3. In the main body of your HTML page you should create a *<div>* to indicate the position of the map on your webpage. The "id" attribute of your div should be set so we can refer to this *<div>* later.
4. Create a javascript function to initialise the map. You can call this in the *onload()* of your main body.
5. In this function you should initialise a new map object with the following line

   new google.maps.Map(document.getElementById("yourDivId"), myOptions);

6. Where *myOptions* should be an object containing some options like center coordinates, zoom factor and map type. These can be found in the reference.
7. You should now have a working map. You can also use this initialize function to insert markers into the map.
8. An important part of working with maps are event listeners. You can listen for example for mouse clicks on many object. For example for a mouse click on a marker to open an info-bubble (infowindow).
9. A listener for a mouse click on a map location will return the geographic coordinates. See example below on how this can be used.

   ```
   google.maps.event.addListener(map, 'click', function(event){
           map.panTo(event.latLng);
   });
   ```

```
<html>
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=true"></script>
<script type="text/javascript">
 var map;
 function initialize() {
        //start position
     var latlng = new google.maps.LatLng(52.456846,5.110703);

    //options of the map
     var myOptions = {
       zoom: 14,
       center: latlng,
       mapTypeId: google.maps.MapTypeId.SATELLITE
     };

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

     //marker icons
     var iconImage =  new google.maps.MarkerImage("http://i50.tinypic.com/nf3ba.png");

     //markers hardcoded positions
     var marker1 = new google.maps.Marker({position: new google.maps.LatLng
(52.458088,5.100035),icon: iconImage, title: "Harbor performance", map: map});
     var marker2 = new google.maps.Marker({position: new google.maps.LatLng(52.460028,5.10651),
icon: iconImage, title: "Church performance", map: map});
     var marker3 = new google.maps.Marker({position: new google.maps.LatLng(52.45748,5.102055),
icon: iconImage, title: "Hof van Marken", map: map});
     addBubble(marker1, "In the harbor of marken there"+"<br/>"+"will be an hourly performance.");
     addBubble(marker2, "Various artist will present their"+"<br/>"+" work in and around the
village church.");
     addBubble(marker3, "The tiny streets of the hof van marken"+"<br/>"+" are an excelent location
to spot"+"<br/>"+" artists and installations.");

     //click to pan to mouse position
     google.maps.event.addListener(map, 'click', function(event) {
               map.panTo(event.latLng);
      });

  }

  function addBubble(marker, text){

       //infowindow (bubble)
       var infowindow = new google.maps.InfoWindow({
          content: text
       });
      //click to open popup bubble for a marker
       google.maps.event.addListener(marker, 'click', function() {
           infowindow.open(map,marker);
      });
  }
</script>
</head>
<body onload="initialize()">
<!-- //buttons -->
<button onClick="initialize()">Go to Marken</button>
  <div id="map_canvas" style="width:100%; height:100%"></div>
</body>
</html>
```