

# GIS – GPS – Geocoding – Maps and more



Presented by: **Thomas Maul, 4D Germany**

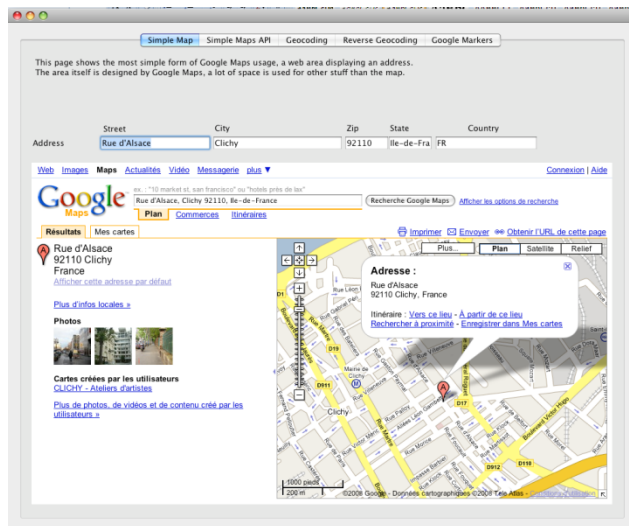
## INTRODUCTION

Since Google Maps people are used to maps or satellite picture on their computers. The most noticed example database of 4D Live Window was the Google Maps mash-up to display a map in a 4D form.

But there is much more possible: display directly customer(s) or shop locations in a map, such as Google Maps or OpenStreetMap. Display images in a map. Calculate the geographical coordinates based on an address – or find an address based on coordinates. Geotag images or find locations where a photo was made. Read data from a GPS device and display it in a map and much more.

## SIMPLE MAP

The most simplest way to include a map into a 4D form is to use Web Area in 4D v11 SQL or 4D Live Window in v2004. Construct an URL using the address – and the map appears.



The visible disadvantage is the "unwanted" area around the map, such as the Google bar, the results on the left side, or the popup to display the marker.

Constructing the URL is not difficult, but the Geo\_Component provides a method to make it even simpler. Call it as:

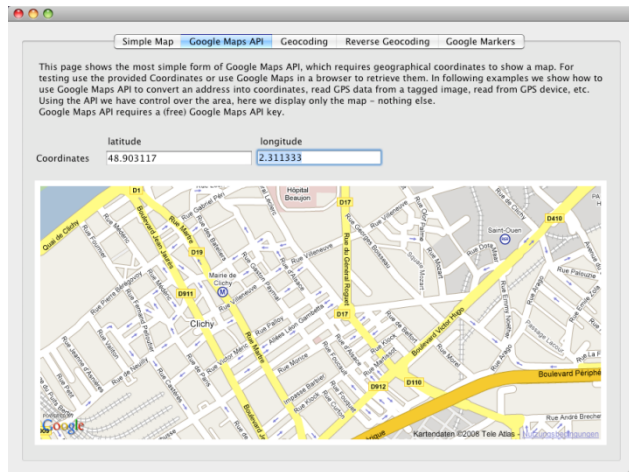
```
$url:=Geo_ConvertAddressToGoogleMapsU (tAddress;tcity;tzip;tState;tCountry;15;"en")
```

Pass the address as parameters, then the zoom level (15 in this example) and the language for the Google interface (English in this example). Done. With Web Area the result can be directly assigned to the Area URL variable.

## GOOGLE MAPS API

Using the Google Maps API we get full control of the area. We can decide to display or hide the zoom level, the layers (map/satellite/Terrain/etc) and using JavaScript we can add markers, overlays, polygons or react on user events such as mouse clicks. Because Google Maps API takes heavy usage of JavaScript, it is strongly recommended to use 4D v11 SQL for this usage. While it is possible with 4D Live Window on Windows, you may run into crashing problems on Mac OS with 4D Live Window and 4D v2004.

The following screen is a quite simple usage of Google Maps API, we simply hide everything:



As simple it is, we have maximum usage of our space and can display as much of the map as possible.

Using Google Maps API requires a little bit more work and preparation. First we need a Google Maps API key to identify our application against Google. The key can be required (for free) at:

<http://code.google.com/apis/maps/signup.html>

This requires of course the acceptance of the Google Maps API Terms of Use, which should be considered before embedding Google Maps into your application (see later chapter OpenStreetMap for alternative solution).

Technically the main difference to the first example is that we need to pass geographical coordinates and not simply an address. Obtaining this coordinates can be done with several approaches. There are tools to lookup an address (Geocoding) to get the coordinates, they can be read from a GPS device (Geotracking) or as example be embedded in a photo (GeoTaging, as with the new iPhone). We will see soon several examples how to do that.

Google Maps API allows us to embed a map into a HTML page, but it cannot "exists" on it's own, it must be included somewhere. This can be a local document, it does not need to be on a real web server. The component provides a command to create the document on the fly in the temp folder with the correct content and returns the URL. To allow to have several maps (in the same or different window or processes), we need to provide an unique ID for each map, which could be by example the web area object name + process number. The map is created on a specified size, by example the size of the web area object.

```
$UniqueObjectName := "WebArea1"+String(Current Process)
GET OBJECT RECT(*;"WebArea1";$left;$top;$right;$bottom)
$url:=Geo_CoordinatesToGoogleMapsURL (tlongitude; tlatitude;
    GoogleMapKey;$right-$left;$bottom-$top;15;$UniqueObjectName)
```

Before we go deeper into Mapping concepts, we discuss how to retrieve the coordinates.

### Geocoding using Google Maps API

Google Maps API includes some features to do geocoding. The Terms of Use allows to run up to 15000 geocoding requests per day, but deny to save/store this informations in the database or to use them for other purposes than to display on the map. We will also discuss using Open Source systems if you need this.

Similar to the normal Google Maps Web Interface we can pass an address and get the coordinates returned:

```
Geo_GoogleMapsGeoCoding (tAddress;tcity;tzip;tState;tCountry;->tlongitude;
    ->tlatitude;GoogleMapKey)
```

While finding the coordinates of an address can be done with a single call, the reverse is a little bit more complex. For given coordinates we can retrieve the nearest street. To do that we need to display a web area (which can be

invisible), in which a Google Maps object is used with some JavaScript code to retrieve the address. As soon the address is calculated, the JavaScript code calls 4D to pass the address.

The component provides two methods for this task:

```
$url:=Geo_GoogleMapsReverseGeoCoding (tlongitude;tlatitude;GoogleMapKey;
    $UniqueObjectName)
WA OPEN URL(*;"WebArea1";$url)
```

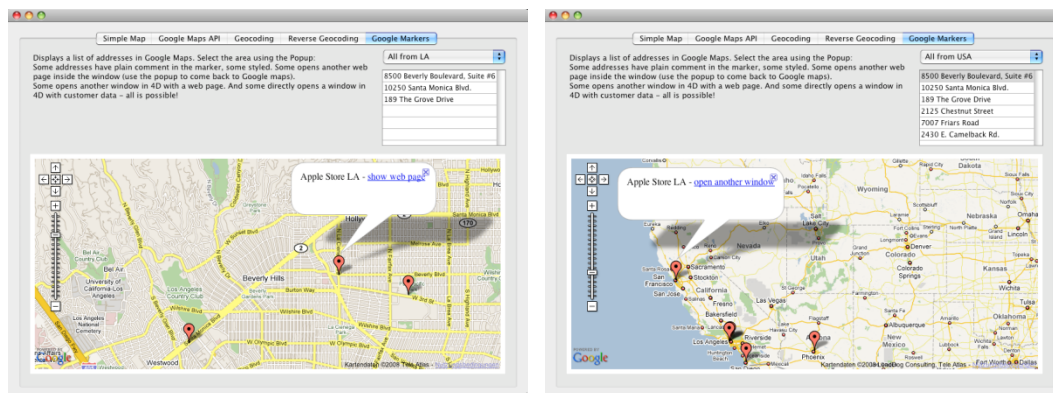
Similar to Geo\_CoordinatesToGoogleMapsURL this method returns an URL which will be displayed in a web area. The web area needs the "On URL Filtering" event enabled and an object method like this:

```
If (Form event=On URL Filtering )
    $url:=WA Get last filtered URL(*;"WebArea1")
    Geo_GoogleMapsReverseFilter ($url;->tAddress;->tCity;->tZip;->tState;
        ->tCountry)
End if
```

After the call the address variables contains the closed address to the passed coordinates.

## Google Maps API and markers

We know already how to display a map in Google Maps using coordinates. The simple way is to center the map for the given coordinates. Let's make it more interesting, we want to display some addresses on a map with markers. Clicking on a marker – or on the record in 4D (!) – should select the marker and show a tip with some informations. The tip can include a HTML link, which either switch to this address or triggers 4D to show more data in a new 4D window.



Try it with the demo application. The popup menu on the right upper corner allows to select a customer group (here some Apple Stores). Clicking on a marker or on the address in the listbox activates the marker and shows a tip inside the web area. Two addresses don't show a tip – but directly opens a 4D Windows. Some tips contains plain text, some rich text, some URL's. A URL can change the currently displayed page – or triggers 4D which opens another window with a web area.

To build such an object we need to pass similar parameters as before, just now arrays as the map can contain several markers/coordinates. In addition to a longitude and latitude array we need an array with tip content and an array with direct click URL's.

```
$url:=Geo_MarkersGoogleMapsURL (->$lon;->$lat;->$label;->$markerurl;
    GoogleMapKey;$right-$left;$bottom-$top;15;$UniqueObjectName)
WA OPEN URL(*;"WebArea1";$url)
```

The label array can be plain text, such as "my Store" or "Straße" or "4 > 3", 4D handles automatically the character encoding. Starting the string with char(1) allows to disable the character encoding, which allows us to use HTML commands, such as char(1)+"<b>bold</b> text"

The array `markerurl` allows to install a click event trigger. As soon the user clicks on such a marker, the URL is launched. In some cases this may be the expected result, sometimes you may want to do something totally different. The Filter concept of web area (in 4D Live Window called navigation event) allows that:

```
ARRAY TEXT($filters;0)
ARRAY BOOLEAN($AllowDeny;0)
APPEND TO ARRAY($filters;"*") `Select all
APPEND TO ARRAY($AllowDeny;True) `Allow
APPEND TO ARRAY($filters;"*OpenURL*") `Run Object method for this URL..
APPEND TO ARRAY($AllowDeny;False)
APPEND TO ARRAY($filters;"*OpenCust*") `Run Object method for this URL..
APPEND TO ARRAY($AllowDeny;False)
WA SET URL FILTERS (*;"WebArea1";$filters;$AllowDeny)
```

URL's containing "OpenURL" or "OpenCust" are filtered and the object method is executed.

```
If (Form event=On URL Filtering )
    $url:=WA Get last filtered URL(*;"WebArea1")
    If ($url="@/OpenCust/@")
        $pos:=Position("/OpenCust/";$url)
        $rest:=Num(Substring($url;$pos+Length("/OpenCust/")))
        GOTO SELECTED RECORD([Test_Address];$rest)
        $p:=New process("Demo_DisplayCustomer";512000;"Demo
            Customer";Record number([Test_Address]))
    End if
End if
```

This is just an example, we test if the url contains "OpenCust", then expect the record number in the selection behind the URL and open a new process to display this record. We could also use customer ID, etc. This code is without error checking, see the example for full code.

But how do select a marker in the map by clicking in the listbox? We use Javascript:

```
If (Form event=On Clicked )
    $pos:=Find in array(listbox;True;1)
    If ($pos>0)
        $command:="doclick("+String($pos-1)+") "
        WA Execute JavaScript(*;"WebArea1";$command)
    End if
End if
```

Each marker can be accessed using the index of the array -1 (-1 because in JavaScript an array starts with element 0, while in 4D with element 1). The JavaScript method "doclick" (you need to write in lowercase because JavaScript is case sensitive) simulates the user click. This works fully transparent, if you select a customer which triggers back 4D, such as "707 Friars Road" in San Diego in the listbox, 4D will react and open another window. Check the code, the listbox does not call 4D, it just call JavaScript. The JavaScript code clicks the trigger, which executes the URL, which triggers the filter, which opens the window...

## OPEN SOURCE GEOCODING SERVICES

Commercial geocoding services, such as Google or Yahoo, limits the usage of the received information's, by example the coordinates are only to be used with the map itself, it is not allowed to store them offline. As long this is not needed, these services provide excellent information's.

As alternative there are open source/free services, which usually publish their data following the "creative commons" license. This license is very open (see [creativecommons.org](http://creativecommons.org)).


While this services are easy to access and provide often very detailed information's, specially the Geocoding of street numbers is poor compared to Google Map API. On the other side, services such as Geonames.org have detailed data as population, alternative names (localized names for cities) or services as OpenStreetmap return

## Geonames.org

[illegible]

There are many options to limit the result, see: <http://www.geonames.org/export/geonames-search.html>

Geonames also provides a ZIP based Geocoding service:



Name

Reverse: Zip

Reverse: Place

Reverse: Address (US)

Search Anything

Using [ws.geonames.org](http://ws.geonames.org) we can also search directly for zip codes, like:  
<http://ws.geonames.org/postalCodeSearch?postalCode=85375&country=DE&maxRows=10>

Result contains coordinates and some more info. You must enter the ISO name of the country, like US or DE

Zip

Country

Address

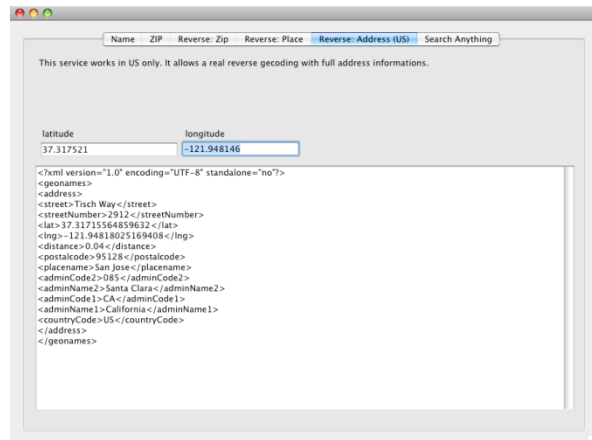
95128

US

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<geonames>
<totalResultsCount>1</totalResultsCount>
<code>
<postalCode>95128</postalCode>
<name>San Jose</name>
<countryCode>US</countryCode>
<lat>37.3163</lat>
<lng>-121.9336</lng>
<adminCode1>CA</adminCode1>
<adminName1>California</adminName1>
<adminCode2>085</adminCode2>
<adminName2>Santa Clara</adminName2>
<adminCode3>
<adminCode3>
</adminCode3>
</code>
</geonames>
```

The Service provides based on ZIP and country the geographical coordinates and some administrative informations. Again the Web service is HTTP based and simple to call, see method DemoOpenSource\_FormPage for an example. The service is using only the ZIP code and does not allow using the street name for more accurate results.

Other services from Genomes are Reverse Geocoding with a result as ZIP, place or address (US only):



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<geonames>
<address>
<street>Tisch Way</street>
<streetNumber>2912</streetNumber>
<lat>37.31715564859632</lat>
<lng>-121.94818025169408</lng>
<distance>0.04</distance>
<postalcode>95128</postalcode>
<placename>San Jose</placename>
<adminCode2>085</adminCode2>
<adminName2>Santa Clara</adminName2>
<adminCode1>CA</adminCode1>
<adminName1>California</adminName1>
<countryCode>US</countryCode>
</address>
</geonames>
```

## OPENSTREETMAP

From the web site: "OpenStreetMap is a free editable map of the whole world. It is made by people like you. OpenStreetMap allows you to view, edit and use geographical data in a collaborative way from anywhere on Earth."

Similar to Wikipedia, OpenStreetMap data is collected, entered and edited by "normal people" and extremely fast growing. People add streets, foot ways and cycle ways by using GPS devices to receive tracks and then vectorize them. In addition they add street names, restaurants, bus stops, postage boxes, tourist attractions, etc. OpenStreetMap is becoming very fast a very complex detailed system with information's difficult to find somewhere else – and totally free. All information's are covered by the creativecommons license! This allows to use maps or data in commercial applications...

Very active centers are in Europe, such as in UK and Germany, with details such as hiking trails or cycle roads. In the United States they imported the TIGER data set and got a full coverage of all cities/streets...

Maps can be downloaded in different graphical formats (including SVG) or as raw XML data (nodes with coordinates). Open Source applications allow converting this data into other formats, such as maps for GPS devices like Garmin eTrex.

## Using OpenStreetMap for Geocoding

Using the raw data behind the maps of OpenStreetMap provides a very powerful search engine, which finds almost everything – except street numbers. It allows to find for addresses, like "Hanns-Braun-Strasse, Neufahrn, Deutschland", for places like "Zoo, München" (or Zoo, Munich), for cities in the world (San Jose), or in an area (San Jose, CA) or even places in that city (Restaurants near San Jose, CA). It knows about places like "LAX" and can even do reverse geocoding by simply providing the coordinates.

For much more examples see: [http://wiki.openstreetmap.org/index.php/Name\\_finder](http://wiki.openstreetmap.org/index.php/Name_finder)

The result is returned in XML format, allowing easy access, see the following example:

NameZIPReverse: ZipReverse: PlaceReverse: Address (US)Search Anything

Using [www.openstreetmaps.org](http://www.openstreetmaps.org) we can also search for nearly anything, street names, cities, places, airports, gas stations...  
Examples: "Hanns-Braun-Strasse, Neufahrn, Deutschland"  
"San Jose" or "San Jose, CA" or even "Restaurants near San Jose, CA"  
"LAX"  
"48.902547,2.311335"  
many more examples, see: [http://wiki.openstreetmap.org/index.php/Name\\_finder](http://wiki.openstreetmap.org/index.php/Name_finder)

Find

Zoo, München

```
<?xml version="1.0" encoding="UTF-8"?>
<searchresults find="Zoo, München" sourcedate="2008-07-14" date="2008-07-14 15:57:45" distancesearch="no"
findname="Zoo" findplace="München" foundnearplace="yes">
<named type="node" id="117948626" lat="48.099445" lon="11.557992" name="Flamingo-Eingang" category="tourism"
rank="0" region="48008" info="gate: zoo" zoom="16">
<description>gate: zoo &lt;strong>Flamingo-Eingang&lt;/strong> found about 1km south-east of middle of suburb
&lt;strong>Thalkirchen&lt;/strong> in München (which is about 4km south-west of city &lt;strong>München
[en.Munich]&lt;/strong> in München, Oberbayern, Bayern, Bundesrepublik Deutschland, Europe) and about 4km south of
city &lt;strong>München [en.Munich]&lt;/strong>, ditto (which is about 70km south of city &lt;strong>Ingolstadt&lt;/
strong> in Ingolstadt, Oberbayern, Bayern, Bundesrepublik Deutschland, Europe)</description>
</place>
<named type="node" id="240059422" lat="48.137254" lon="11.575513" name="München [en.Munich]" category="place"
rank="60" region="48008" is_in="in München, Oberbayern, Bayern, Bundesrepublik Deutschland, Europe" info="city"
distance="4.391209" approxdistance="4" direction="73" zoom="10">
<nearestplaces>
<named type="node" id="240118839" lat="48.766700" lon="11.433300" name="Ingolstadt" category="place" rank="60"
region="48008" is_in="in Ingolstadt, Oberbayern, Bayern, Bundesrepublik Deutschland, Europe" info="city"
distance="70.658207" approxdistance="70" direction="99" zoom="10">
</nearestplaces>
</named>
</nearestplaces>
</place>
</nearestplaces>
```



Searching for Zoo, Munich, finds 4 entries. The Zoo itself (which is named for Munich "Tierpark Hellabrunn", the two main entries (first one in the screen shot) and a small zoo nearby. Each with the coordinates, a name (Flamingo-Eingang = Flamingo Entry), category (tourism) and a description, which is in HTML format:

gate; zoo **Flamingo-Eingang** found about 1km south-east of middle of suburb **Thalkirchen** in München (which is about 4km south-west of city **München [en:Munich]** in München, Oberbayern, Bayern, Bundesrepublik Deutschland, Europe)

## Using Geocoding/Coordinate's for find customers nearby

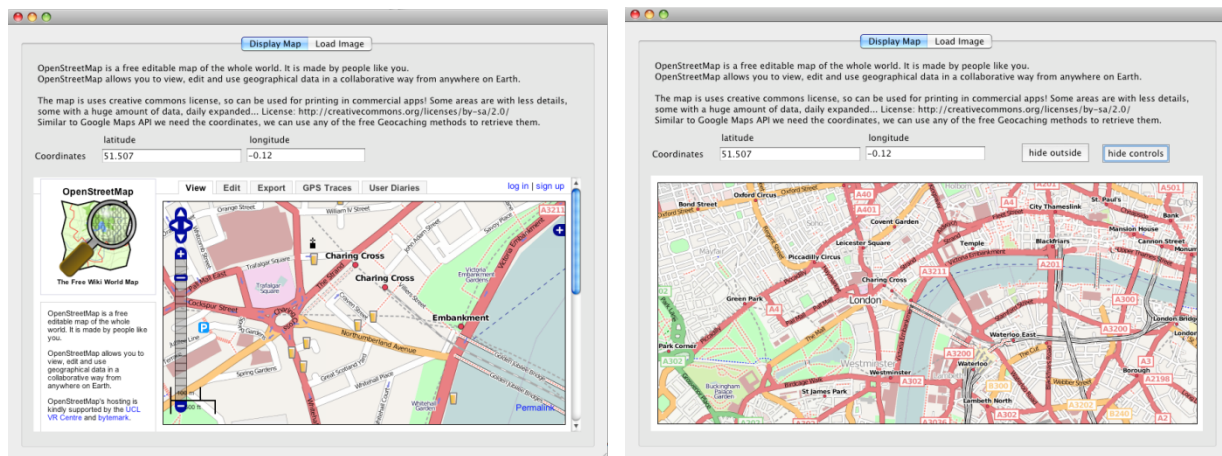
Do you want to find all customers in an area, like 20 miles distance to a specific point? You'll find the formula how to calculate that – and some interesting benchmarks about speed with 4D v11 SQL on our web site:

[http://www.4d.com/products/benchmarks/qbf\\_mono20080624.html](http://www.4d.com/products/benchmarks/qbf_mono20080624.html)

## Displaying Maps from OpenStreetMap

This is as simple as with Google Maps, we simply create an URL and display it with Web Area:

```
$url="http://www.openstreetmap.org/?lat="+tlatitude+"&lon="+tlongitude  
      + "&zoom=14"  
WA OPEN URL (*; "WebArea1"; $url)
```



Depending of the zoom level, the map shows more or less details. In close views it displays tourist's information's, even pubs, which are hidden in higher scales.

Using the JavaScript features of Web Area we can control the display of the map, try clicking the "Hide outside" or "Hide controls" button. Even when the controls are hidden, you can still drag the map using the mouse – or zoom using the mouse scroll wheel.

(Tip: to discover the correct <div> ID's to hide/show, etc, a good approach is to use Firefox with installed Firebug extension, select the HTML tab and click on the <div> entries. Firefox will highlight the area. Also the DOM list helps.)

OpenStreetMap is using openlayers, so it can be deeply customized (see [www.openlayers.org](http://www.openlayers.org) or [dev.openlayers.org](http://dev.openlayers.org))

## Retrieving Image from OpenStreetMap

As the OpenStreetMap license allows us to embed maps into our application, printings, etc, it is interesting just to receive an image of a map. This is supported in various formats (JPEG, PNG, SVG, PDF or Postscript). You can try the export with the previous example, simply click on Export tab, browse to your home city, zoom in/out and click the export button.

Of course we can do this also automatically:

```
$err:=Geo_LoadOSMImage (tLongitude;tlatitude;$meterX;$meterY;tScale;"jpeg";->pImage)
```

Pass longitude/latitude and the requested size of the image. The size has to be given in Meter (1000 Meter = 1 Kilometer = 0,62 Miles). SizeX and SizeY define the ratio of the image. The size of the returned image is limited



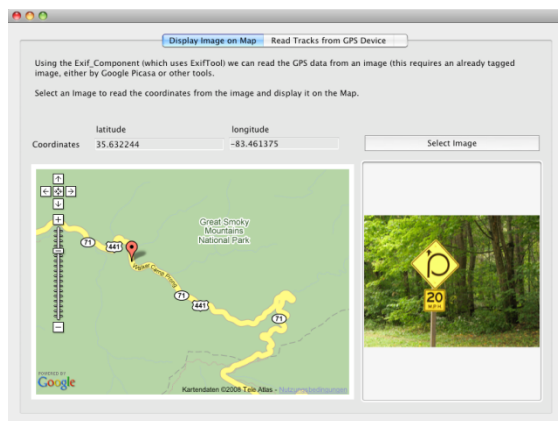
( $X+Y < 4000$  pixel), with Scale parameter defining the resolution and so the size of the image. If the requested scale would return a too large image, the command returns as error -1.



## GEOTAGGING – COORDINATES IN PHOTOS

Devices with both GPS receiver and camera (like the new iPhone) often include the geographical coordinates in an image. Services such as Google Picasa or Flickr displays this pictures directly on a map. Applications as Adobe Photoshop, Apple iPhoto or even the Finder are displaying this information's.

By example a real estate agency could make images of the offered houses using an iPhone (or similar device) and your software automatically creates the exposé including maps showing the location of the objects:



The example database uses the "Exif\_Component" for this task, which itself uses the Open Source Project ExifTool (GPL License). ExifTool is a command line tool, on Mac available via Terminal, on Windows using DOS Window. It can be called from 4D using Launch External Process. The component includes the binary as well as the GPL license. While the component is compiled it also includes the 4D source, so it can be modified if required.

The component provides two functions, Exif\_ReadAllTags and Exif\_WriteTags. The first one returns two array, one with all tag names, the other with the corresponding values. These are not just the GPS Tags, all Exif, IPTC, GPS and other meta information's are returned. Exif\_WriteTags allow modifying, adding or deleting one or several tags. For full documentation see the manual (open the component package to see documentation, ExifTool license and source).

Open the Exif\_Component as 4D database allows to run a test form. Select an image (jpeg or raw) or other medias (pdf, flash, mpeg, etc). The listbox shows all readable meta data. Many of them are modifiable, simply long click in the values field to modify an item, such as keywords. Try "Add Tag" or "Remove Tag". The "Shift Time" buttons include examples how to modify the embedded time stamp, often necessary if a folder with pictures was done with wrong camera settings (like a wrong time zone).

The following screen shot shows an image with embedded GPS data. The two most important tags (and the only ones used in the Geo\_Component database are GPS Latitude and GPS Longitude. Usually there is also a GPS Altitude tag. Some Geotagging applications include reverse lookup and directly include fields just as City or Country.



## GPS DEVICES – GPX DOCUMENTS

So far we got coordinates by geocoding. Another typical resource is GPS devices. While this could be a navigation system, often it is a very small device in the size of an USB Stick which creates a so called "tracklog". This is a list of time + coordinates saved in regular intervals, as every x seconds or every x meter distance. Some devices (like the Garmin family) display the tracklog on their screens.

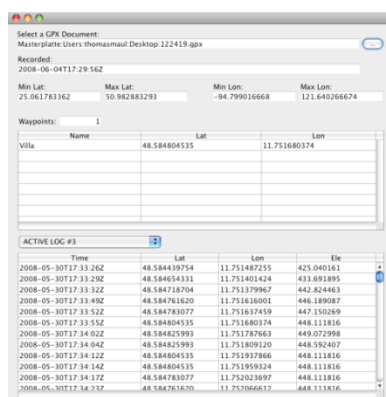
Often the device allows creating "waypoints", a kind of marker of the current position, often with a name or comments.

Many modern devices, such as the Garmin eTrex Legend, could be connected via USB and can be accessed similar to an USB Stick. The tracklog can be simply opened as text/xml document via Open Document. The log itself is a XML document in the GPX format. Older devices or some navigation systems can be accessed with open source software such as "GPSBabel", which also can convert many formats into GPX ([www.Gpsbabel.org](http://www.Gpsbabel.org), useable as console application for Mac and Windows, accessible from 4D with LAUNCH EXTERNAL PROCESS).

A 4D application can read the tracklist and by example use the waypoints and assign them to records and/or display them on a map. It could also parse an image folder and assign images to coordinates, by comparing the time of the image with the position in the tracklog.

While GPX is a simple format, we provide a component to make the usage even simpler: GPX\_Reader

The component provides 3 methods, Geo\_GPXGetInfo, Geo\_GPXGetTrackpoints and Geo\_GPXGetWaypoints. They pass a document and return the requested content in arrays. See the method comments for documentation. The method Geo\_GPXTest opens a test dialog, helping to analyze the content of documents:



## TERMS OF USE AND LICENSES

All maps produced with Google Maps in this document are included as example/demonstration only. You are not allowed to reuse/reprint them for other purpose (see Google Maps Terms of Use).

JavaScript Code used in the Google Reverse Geocoding example is licensed under the Apache license, see [example/Resources/OpenSourceFiles](#) for details.

All maps produced with OpenStreetMap ([www.openstreetmap.org](http://www.openstreetmap.org)) are under the Creative Commons Attribution-Share Alike license. You are freely allowed to reuse them following the conditions of the license:

<http://wiki.openstreetmap.org/index.php/Attribution>

Exif\_Component is using ExifTool, an open source console application (using Terminal on Mac, DOS on Windows). If you use/distribute this component/tool with your application don't forget to add the license agreement and inform the customer that he can use/distribute this tool following the license. More information's:

<http://owl.phy.queensu.ca/~phil/exiftool/index.html>