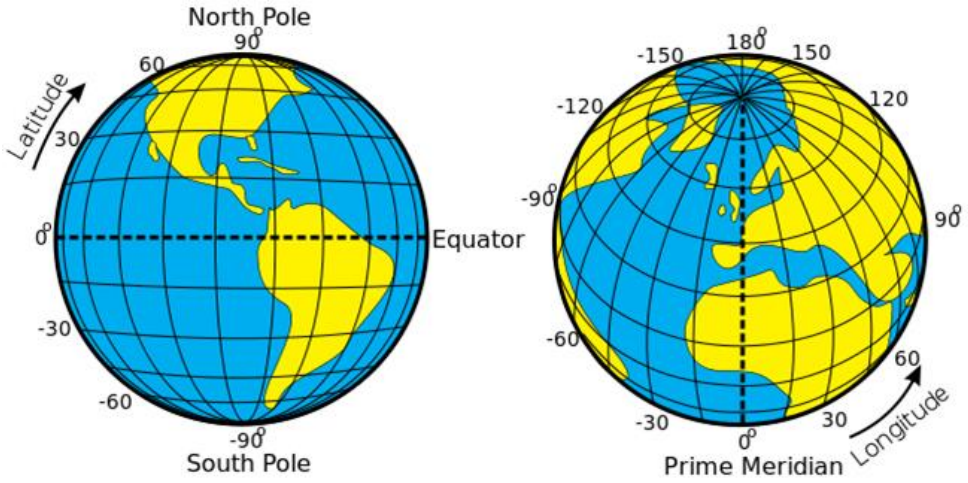
Using GEO services in Delphi applications with TMS components

Introduction

In the past few years, a vast aray of services related to a position on our planet earth became available. With a wide range of components, TMS software offers seamless access to these services from Delphi VCL desktop applications, IntraWeb web applications and FireMonkey mobile applications running on iOS or Android. As such, it's mostly your imagination that is the limitation of what you can do these days with geo services in your applications.

Basis of geo services

The fundament on which all geo services are built is the determination of a position on our planet. For this, in almost all cases, the system of longitude and latitude is used. The latitude is a value between -90 and +90 that defines the angle of the position between south pole and north pole. The longitude is a value between -180 and +180 that defines the position along the equator. These values are typically expressed in decimals or with degrees, minutes and seconds. For ease of calculation, most services use the decimal notation.



Types of geo services

If there is a classification to make for what geo services are being used, we'd divide these in following categories:

- Mapping: techniques for displaying & manipulating maps and visualize information on maps
- Geocoding / reverse geocoding: techniques for converting an address to a longitude and latitude and vice versa

- Geolocation : techniques to obtain the longitude and latitude of a computing device
- Geo POI services : services that provide information / data about points of interest at a specific position
- Routes : calculate the routes between two or more positions

In this article, we have examples for using each of these different services from Delphi applications.

Mapping

The 3 major services that provide digital maps are Google Maps, Microsoft Bing and Open Street Maps. Digital maps are provided via a HTTP service and can be displayed via a browser. These mapping services are targetted mainly at browser usage, so if we want to take advantage of these maps from a Delphi application, the main two challenges are calling a Javascript API from the Delphi application that is executed in the browser and handle Javascript events that are being triggered from the map and that ideally are exposed as class events to a Delphi application. For a Windows applpication, to call Javascript functions from a Delphi application for a map displayed in a browser, the HTMLWindow.execScript() function can be used. To handle Javascript events from a browser at Delphi application level, it is required to implement the IDocHostUIHandler interface. Fortunately, this is somewhat easier from an IntraWeb application as the map is displayed typically on the same page where the IntraWeb application page is rendered. For a FireMonkey mobile application, a technique similar to the Windows desktop application is necessary but in this case with the browser that runs on the mobile device. Fortunately, TMS software offers components for desktop, web and mobile application with the same interface, so this makes using the maps ease on any of these platforms.

To get started with using Google maps, drop the TMS TWebGMaps component on the form and add the code:

```
webgmaps1.MapOptions.DefaultLatitude := 51.2;
webgmaps1.MapOptions.DefaultLongitude := 4.37;
webgmaps1.Launch;
```

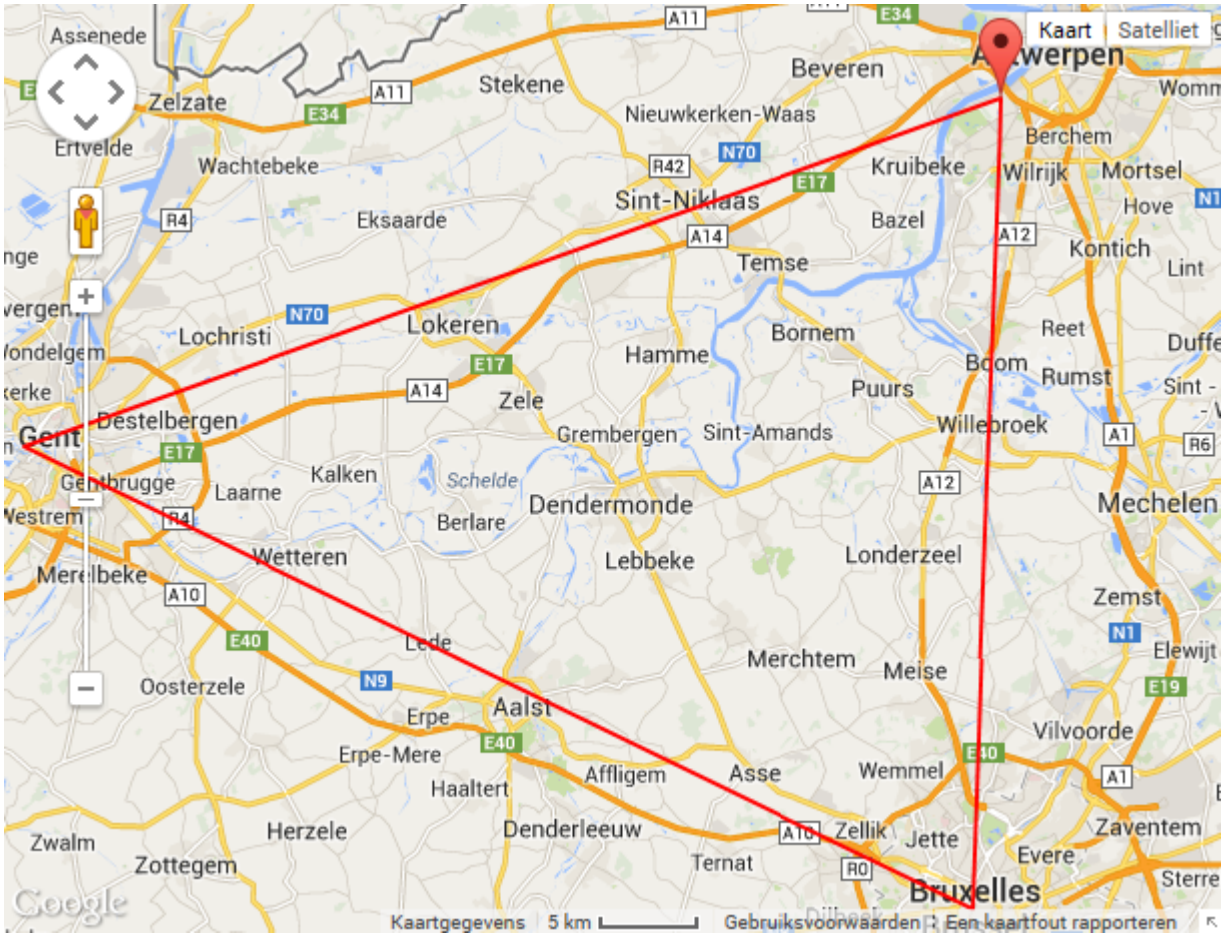
This code snippet initializes the map for position 51.2 / 4.37 that is Antwerp, Belgium. Next we can add a marker at this location. Markers are exposed via a markers collection. By default a marker with a hint is added with this code:

```
webgmaps1.Markers.Add(51.2,4.37,'Antwerpen')
```

Next task is to draw a polyline on this map. To illustrate this, code will be added to draw a triangle between 3 cities: Antwerpen, Gent, Brussels. To draw a polygon, we basically create a path, i.e. a collection of coordinates between which the polygon is drawn. The path is of the type TPath and this collection

contains TPathItem that holds the longitude and latitude of each point. When this path collection is created, it is added to the PolyLines collection that is available at TWebGMaps level. This results in code:

```
var
  pt: TPath;
  pti : TPathItem;
begin
  pt := TPath.Create(webgmaps1);

  pti := pt.Add;
  pti.Latitude := 51.2;
  pti.Longitude := 4.37;

  pti := pt.Add;
  pti.Latitude := 51.05;
  pti.Longitude := 3.7;

  pti := pt.Add;
  pti.Latitude := 50.85;
  pti.Longitude := 4.35;

  pti := pt.Add;
  pti.Latitude := 51.2;
  pti.Longitude := 4.37;


webgmaps1.Polylines.Add(false,false,false,nil,pt,clred,255,2,true,
100);

  pt.Free;
end;
```



Geocoding / reverse geocoding

Geocoding is the process of converting an address to a longitude and latitude coordinate. Reverse geocoding means obtaining an address starting from a longitude and latitude coordinate. This is typically performed by a company that has all the required mapping data to perform this function. Some of the services that provide this are: Google Maps, Microsoft Bing, OpenAddresses, Yahoo PlaceFinder and several smaller services. When looking at the performance, reliability and quality of the service, Google easily comes out best. Therefore, we created two components that make using the Google geocoding and reverse geocoding service very easy. This is TWebGMapsGeocoding and TWebGMapsReverseGeocoding. To use these components is as simple as specifying the address and retrieving the result longitude and latitude and vice versa.

To demonstrate geocoding, we'll lookup the geolocation of Embarcadero have it displayed on a map and switch to streetview.

With a TWebGMapsGeoCoding component and TWebGMaps component on the form, following code obtains the longitude & latitude of the Embarcadero office in Scotts Valley, adds a marker on the map and pans the map to it:

```
begin
  // set the address
  WebGMapsGeocoding1.Address := '5617 Scotts Valley Dr #200,
Scotts Valley, CA';
  // launch geocoding
  WebGMapsGeocoding1.LaunchGeocoding;
  // pan map to location retrieved
  WebGMaps1.MapPanTo(WebGMapsGeocoding1.ResultLatitude,
WebGMapsGeocoding1.ResultLongitude);
  // add a marker
  WebGMaps1.Markers.Add(WebGMapsGeocoding1.ResultLatitude,
WebGMapsGeocoding1.ResultLongitude,'Embarcadero');
end;
```

To have a streetview on the location retrieved, following code can be used:

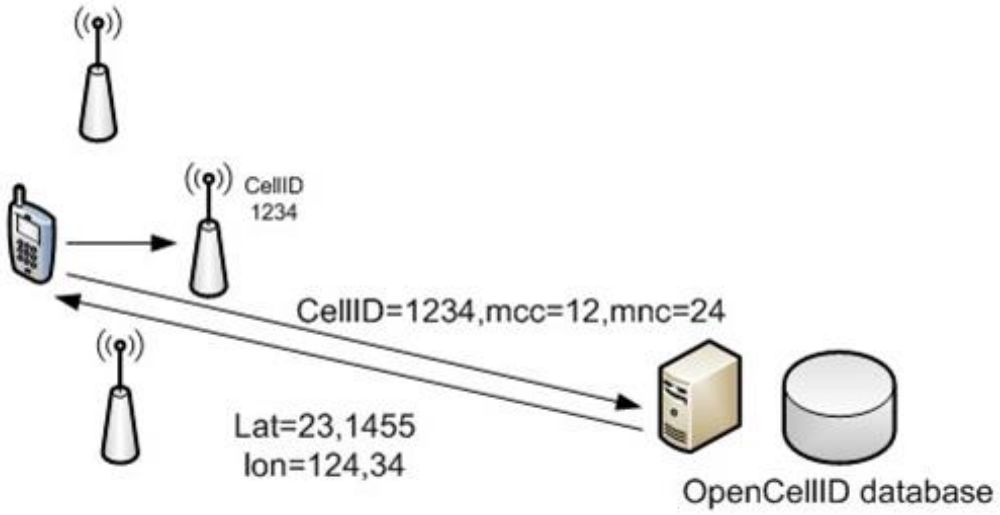
```
  // set coordinates of location to see with street view
  WebGMaps1.StreetViewOptions.DefaultLatitude :=
WebGMapsGeocoding1.ResultLatitude;
  WebGMaps1.StreetViewOptions.DefaultLongitude :=
WebGMapsGeocoding1.ResultLongitude
  // let the map switch to streetview
  webgmaps1.StreetViewOptions.Visible := true;
```

Geolocation

Geolocation is the name used for all kinds of techniques that provide information about the location of a device. These days, most mobile devices have a GPS built-in and this can return the longitude and latide of the device immediately. From Delphi XE4, the non-visual component TLocationSensor is provided that allows you to get this information. Using TLocationSensor is easy. Set LocationSensor.Active = true and via the event OnLocationChanged, the position is returned. The accuracy of determining this location is around 10 metres typically. When no GPS is available, we must resort to different techniques. These techniques can be:

- ISP IP address based: many ISPs have a database of what IP address range is being used in what area. Services exist that gather this information that can be used to retrieve location information based on an IP address.

- Cell phone based: when a mobile device is connected to a cell phone access point, the position of the cell phone access point is known and thus also the area the signal of this cell phone access point covers. There are also services that collect this information and make it accessible. OpenCellID is an example. See: http://www.opencellid.org/cell/map



- WiFi service based: Similar as with cell phone based geolocation, wifi based geolocation could be an option when a device is connected via a WiFi access point and the location of the WiFi access point is known. An example of a service that collects information on the position of WiFi access points is http://www.skyhookwireless.com

For mobile devices, typically a fallback mechanism is used. First, there is a check if a GPS exists. When not, it can try to see if it can find a position based on the IP address, the connected cell phone access point or WiFi access point. This is the mechanism that is built-in these days in any HTML5 compliant browser. This allows web applications to determine where the device is located that connects to it. This is known in the HTML5 standard as HTML5 Geolocation API.

To make it easy for desktop applications to determine as good as possible the location of a machine, TMS software offers a component TAdvIPLocation that uses the FreeGEOIP service. To make it easy for IntraWeb web applications, we have a component TTIWIPhoneGeolocation that uses the HTML5 Geolocation API to determine the location.

Sample code:

```
  if AdvIPLocation1.GetIPLocation then
  begin
    webgmaps1.MapOptions.DefaultLatitude :=
AdvIPLocation1.IPInfo.Latitude;
    webgmaps1.MapOptions.DefaultLongitude :=
AdvIPLocation1.IPInfo.Longitude;
    webgmaps1.Launch;
    memo1.Lines.Add(AdvIPLocation1.IPInfo.ZIPCode + ' ' +
AdvIPLocation1.IPInfo.City);
    memo1.Lines.Add(AdvIPLocation1.IPInfo.CountryName);
  end;
```

This code snippet uses the non-visual component TAdvIPLocation to obtain the location based on the IP address of the machine and shows this location on a map and adds the location city name, ZIP code and country in a memo.

GEO POI services

Geo POI services is the name of services that provide point of interest information at a specific location. This includes things as railway stations, musea, restaurants, sports infrastructure etc... Typically, a service can provide a list of points of interest that matches a requested category and a specific location. It can then offer information such as address, description, recommendations, opening hours of the points of interest. Many services exist that offer this kind of information but the main suppliers with the biggest amount of information are FourSquare, Google Place, Bing Spatial Data Services, Factual...
As FourSquare is one of the leading services, TMS software has a component TAdvFourSquare that makes using this service very easy. Typically, all we need to do is specify a location, i.e. longitude & latitude, specify category of points of interest we're interested in and possibly also a radius. The service then returns a list of points of interests of which we can query a description, photo, etc..

To illustrate this, we'll use the component TAdvFourSquare, available in the TMS Cloud Pack. To start using this component, it is necessary to first obtain a (free) FourSquare application key and secret. You can register for this at https://developer.foursquare.com

First we obtain the different categories and subcategories of points of interests that FourSquare has and fill a listbox with this:

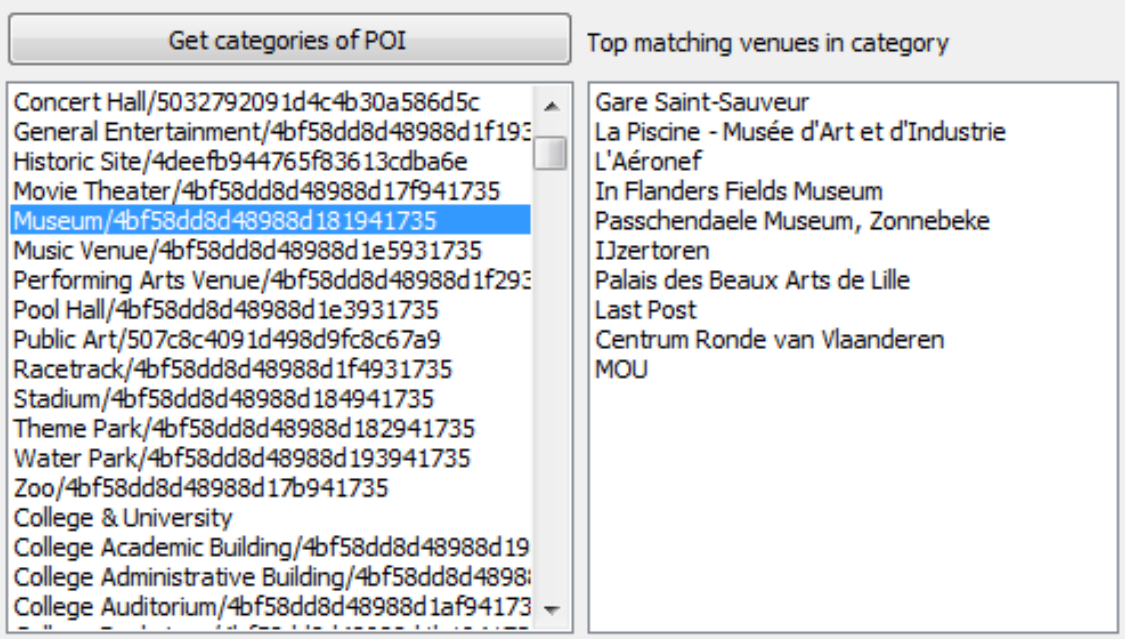
```
var
  i,j: integer;
  id: string;


begin
  AdvFourSquare1.App.Key := FourSquare_AppKey;
  AdvFourSquare1.App.Secret :=  FourSquare_AppSecret;

  AdvFourSquare1.GetCategories;

  for i := 0 to advfoursquare1.Categories.Count - 1 do
  begin
    listbox1.Items.Add(AdvFourSquare1.Categories[i].Summary);
    for j := 0 to AdvFourSquare1.Categories[i].SubCategories.Count
- 1 do
    begin

listbox1.Items.Add(AdvFourSquare1.Categories[i].SubCategories[j].S
ummary + '/'+AdvFourSquare1.Categories[i].SubCategories[j].ID);
    end;
  end;
end;
```

Next, when we click on a category in the listbox, we perform a query of the top 10 points of interests nearby the computer location (obtained with TAdvIPLocation) and fill the listbox with this info:

```
procedure TForm1.ListBox1Click(Sender: TObject);
var
  id: string;
  i: integer;
  la,lo:double;
begin
  id := listbox1.Items[listbox1.ItemIndex];

  id := copy(id,pos('/',id)+1, 255);

  listbox2.Items.Clear;

  AdvIPLocation1.GetIPLocation;
  orgla :=  AdvIPLocation1.IPInfo.Latitude;
  orglo :=  AdvIPLocation1.IPInfo.Longitude;

  // This fills the AdvFourSquare Venues collection with points of
interest:
```

```
  AdvFourSquare1.GetNearbyVenues(orgla,orglo,'','',id);

 // add the summary line to a listbox
  for i := 0 to advfoursquare1.Venues.Count - 1 do
  begin
    listbox2.Items.Add(AdvFourSquare1.Venues[i].Summary);
  end;
end;
```

Other than the summary information, FourSquare makes a lot more information available. This includes the longitude & latitude of the point of interest, the address, phone number, website URL when available, opening hours when available etc... All this information is made easily accessible via the TFourSquareVenue class in the Venues collection.



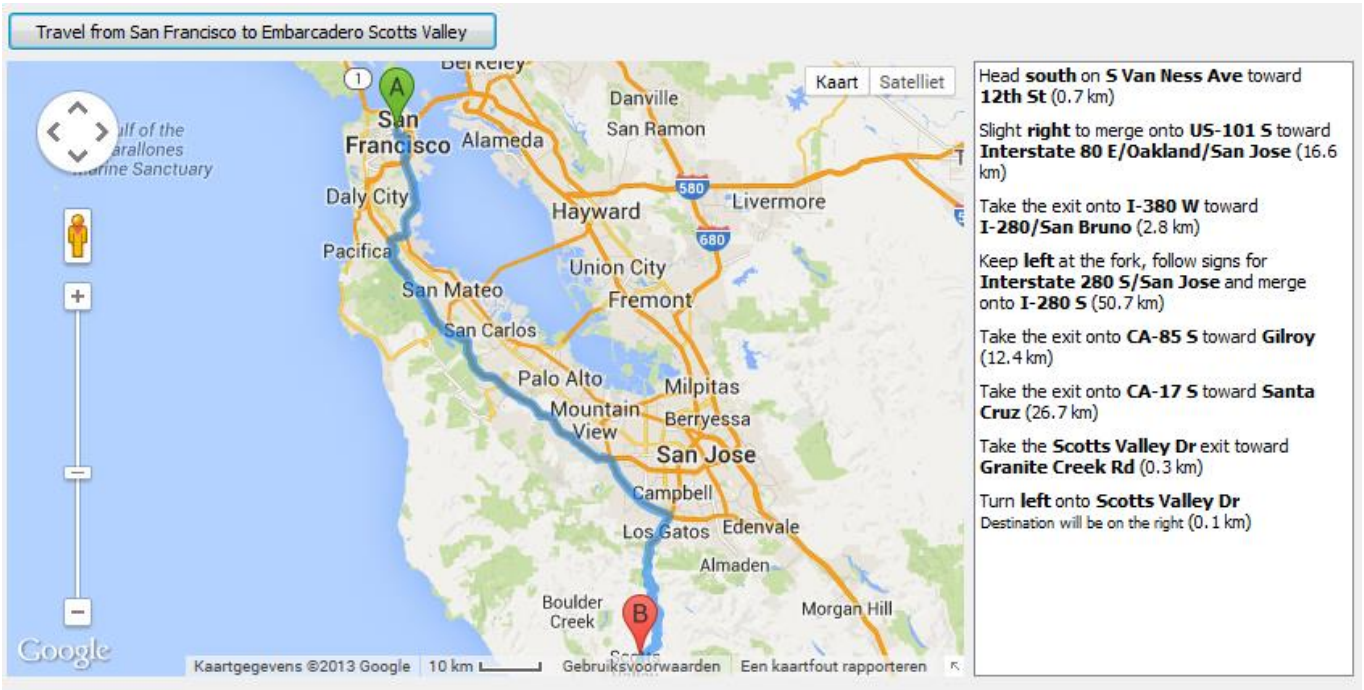
Routes

A final important part in useful geo information based services we can consume from Delphi applications, is getting routing or directions information to travel from a location A to a location B. Again, the three major suppliers of these services are Google with the Google Directions API, Microsoft with Bing Routes API and the Openroute service (http://www.openrouteservice.org/). Such service typically works in following way: we make a request based on two locations, either specified as two sets of longitude/latitude of two sets of addresses, the start address and end address. The service than returns one route or a set of routes that can be used to travel from start point to end point. Note that some services also support waypoints, i.e. points between the start point and end point the route must go along. A route is typically returned as a series of textual descriptions of the route to follow. Each part of the route that is described is called a leg. A set of routes can be returned when alternative

routes exist. Along the textual description, typically also polygon data is returned and this polygon data can be used to visualize the route on a map.

In this example, we use the TWebGMaps component as well as the TWebGMapsDirectionList. TWebGMapsDirectionList is a component especially created to visualize HTML formatted directions information returned by Google. Getting directions is as simple as calling WebGMaps.GetDirections with start and end address. Here we obtain directions information from San Francisco center to the Embarcadero offices in Scotts Valley:

```
var
  from_address, to_address: string;
begin
  from_address := 'San Francisco';
  to_address := '5617 Scotts Valley Dr #200, Scotts Valley, CA';
  webgmaps1.GetDirections(from_address,to_address);

  // fill the list component WebGMapsDirectionList with route
description
  webgmaps1.FillDirectionList(WebGMapsDirectionList1.Items);
  // render the route on the map
  webgmaps1.RenderDirections(from_address,to_address);
end;
```



Many more options are available, such as specifying waypoints, route types, language, travel mode (walking/bike/car) etc... that can be explored with the TMS TWebGMaps component.

Summary

It is amazing what amount of rich (and in many cases free) geo information and geo services are available to us Delphi programmers these days. With this information, we can add useful functionality to Delphi Windows based applications, IntraWeb web based applications and FireMonkey mobile applications for iOS and Android devices. At TMS software, we offer a wide range of components that allow you to consume these services right-away. This allows you to avoid studying the various APIs yourself and be productive immediately to integrate these in your applications. The several products covered in this article to make use of these services are:

TMS WebGMaps: http://www.tmssoftware.com/site/webgmaps.asp
TMS WebOSMaps: http://www.tmssoftware.com/site/webosmaps.asp
TMS Cloud Pack: http://www.tmssoftware.com/site/cloudpack.asp
TMS Cloud Pack for FireMonkey:
http://www.tmssoftware.com/site/tmsfmxpack.asp
TMS WebGMaps for FireMonkey:
http://www.tmssoftware.com/site/tmsfmxwebgmaps.asp
TMS WebOSMaps for FireMonkey:
http://www.tmssoftware.com/site/tmsfmxwebosmaps.asp
TMS IntraWeb WebGMaps: http://www.tmssoftware.com/site/webgmaps.asp
TMS IntraWeb WebOSMaps:
http://www.tmssoftware.com/site/webosmaps.asp
TMS IntraWeb iPhone controls pack:
http://www.tmssoftware.com/site/tmsiwiphone.asp