TMS VCL WebGMaps

DEVELOPERS GUIDE

April 2017

Web: http://www.tmssoftware.com
Email: info@tmssoftware.com

Table of contents

Introduction

The TMS VCL WebGMaps is a component that allows integration of the Google Maps road map control. The TWebGMaps component renders maps of different types: default roadmap view, satellite view, hybrid view (a mix of satellite view with roadmap information), and terrain (topographic style map). TWebGMaps offers pan, zoom and scale control.

In this document you will find an overview of the TWebGMaps component and its features, code snippets to quickly start using the component and overviews of properties, methods and events.

Availability

TWebGMaps and its helper components are available as VCL component for Delphi and C++Builder.

TWebGMaps is available for Delphi 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin,10.2 Tokyo & C++Builder 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin,10.2 Tokyo and can be used for Win32 and Win64 application development *.

TWebGMaps has been designed for and tested with: Windows 2003, Windows Vista, Windows 2008, Windows 7, Windows 8, Windows 10.

*Win64 development requires RAD Studio XE2 or newer versions.

Terms of use

With the purchase of TWebGMaps, you are entitled to our consulting and support services to integrate the Google Maps service in Delphi or C++Builder applications and with this consulting and support comes the full source code needed to do this integration. As TWebGMaps uses the Google Maps service, you're bound to the terms of this Google service that can be found at:

http://code.google.com/apis/maps/terms.html

http://maps.google.com/help/terms_maps.html

TMS software is not responsible for the use of TWebGMaps. The purchase of TWebGMaps does not include any license fee that you might possibly be required to pay to Google. It will depend on your type of usage of the Google Maps service whether a license fee needs to be paid to Google.
It is the sole responsibility of the user or company providing the application that integrates the Google maps service to respect the Google terms and conditions . TMS software does not take any responsibility nor indemnifies any party violating the Google maps service terms & conditions.

Limited warranty

TMS software cannot guarantee the current or future operation & uptime of the Google maps service. TMS software offers the consulting and support for TWebGMaps in good faith that the Google maps service is a reliable and future-proof service. In no case, TMS software shall offer refunds or any other compensations in case the Google maps service terms/operation changes or stops.

List of included components

TWebGMaps is the core map component.
TWebGMapsGeoCoding is the component to convert an address to longitude & latitude coordinate.
TWebGMapsReverseGeoCoding is the component to convert a longitude & latitude coordinate to an address.
TWebGMapsLookupEdit is a helper component that allows entering addresses with auto completion based on information from Google Maps.
TWebGMapsDirectionList is a helper component that allows to display a list of steps of a route with HTML formatted text.
TWebGMapsTimeZone is the component that allows to lookup time zone information based on a longitude & latitude coordinate.

Online references

TMS software website:
http://www.tmssoftware.com

TMS VCL WebGMaps page:
http://www.tmssoftware.com/site/webgmaps.asp

TWebGMaps

TWebGMaps authentication

It is recommended to use an API Key to authenticate your application with the Google Maps JavaScript API service. Retrieving an API Key is free and can be obtained at the Google Developers Console.
Instructions can be found on this page:
https://developers.google.com/maps/documentation/javascript/get-api-key

Make sure to enable the following APIs in Google's Console:

- Google Maps JavaScript API
- Google Maps Directions API
- Google Maps Geocoding API
- Google Maps Geolocation API
- Google Maps Elevation API

The API Key should be assigned to the APIKey property, one time before the map is loaded, in the Form's OnCreate event of an application.

Example:
```
procedure TForm1.FormCreate(Sender: TObject);
begin
  WebGMaps1.APIKey := 'myAPIKey';
end;
```

When using the TWebGMapsGeocoding and/or TWebGMapsReverseGeocoding controls the API Key should be assigned the respective APIKey properties as well.

Example:
```
procedure TForm1.FormCreate(Sender: TObject);
begin
  WebGMapsGeocoding1.APIKey := 'myAPIKey';
  WebGMapsReverseGeocoding1.APIKey := 'myAPIKey';
end;
```

TWebGMaps description

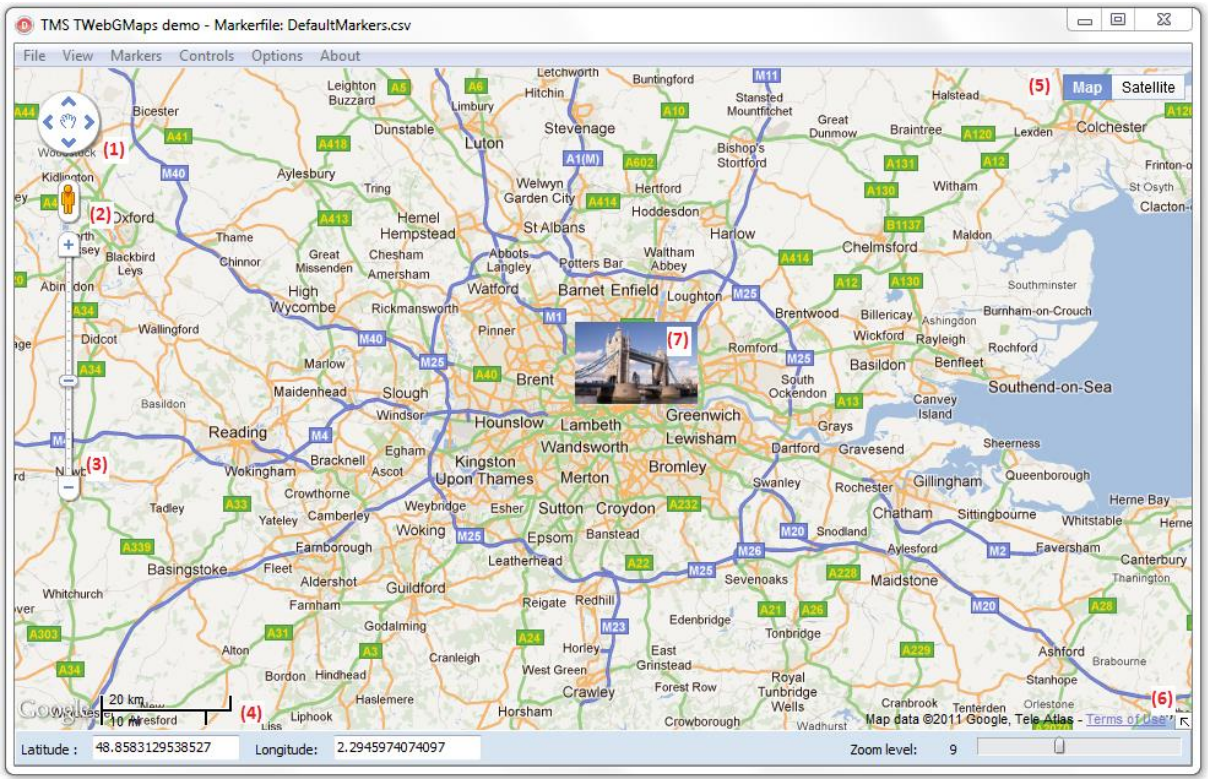
The TMS VCL WebGMaps is a mapping component to integrate for display & control Google Maps in a VCL Windows application. It supports the default roadmap view, satellite view, hybrid view, and terrain view. The TWebGMaps component offers pan, zoom and scale control. An overview-map is integrated for faster panning. Street view offers a life-like 3D experience (where available).
Markers can be added to the map via the longitude/latitude coordinates or via an address. Various marker types exist: default balloon marker, image marker, text marker, marker with hint. Markers can also be displayed with a custom HTML label.
Polylines can be added to the map via a Path, which is a collection of longitude/latitude coordinates.
Polygons can be added to the map via longitude/latitude coordinates. Various polygon types exist: custom polygon, circle or rectangle.
Directions can be retrieved via start and destination address. Directions can also be displayed on the map.
Screenshots of the displayed map can be saved to .BMP, .JPG and .PNG files.
A separate component TWebGMapsGeocoding is available to perform address to longitude/latitude conversions.
A separate component TWebGMapsLookupEdit is available that allows to enter addresses with auto completion based on information from Google Maps.
A separate component TWebGMapsDirectionList is available that allows to display a list of steps of a route with HTML formatted text.

TWebGMaps features

- Different map modes are available: default road map, satellite view, hybrid view (mixed satellite/roadmap view), terrain (topographic style map) or Google streetview (where available).
- Image files can be created of the maps displayed. These can be saved in different formats: .BMP, .JPG or .PNG.
- Extra map information can be displayed: Bicycle View, Panoramio (pictures-) information, Traffic information.
- Position markers can be added to the maps. Markers are displaying additional information on the marker position when clicked. Markers can be default balloons or custom images.
- Markers is a collection of positions that are indicated on the map. Markers are based on longitude and latitude coordinates.
- Moving over a marker can optionally display a hint with the marker title information.
- A custom HTML label can optionally be displayed on top of a Marker.

- Polylines is a collection of lines that are displayed on the map. Polylines are based on a list of longitude and latitude coordinates.
- Polygons is a collection of closed lines with a filled region that are displayed on the map. Polygons are based on a list of longitude and latitude coordinates (for Polygons of type ptPath), a center point and radius (for Polygons of type ptCircle) or two longitude and latitude coordinates (for Polygons of type ptRectangle).
- Directions is a collection of routes between a start location and a destination address.
- Different controls are available and can be turned on or off. MapType control, OverViewMap control, Pan control, Scale control, StreetView control and Zoom control. The position on the screen of the control as well as the visibility and in some cases the style can be defined.
- Different mouse and keyboard options are available: dragging of the map, enabling/disabling all controls, enabling/disabling zoom on double clicking the mouse, enabling/disabling the mouse scroll wheel and enabling/disabling the keyboard.
- TWebGMapsGeocoding is a separate component that takes an address as input and converts this to longitude and latitude.
- TWebGMapsReverseGeocoding is a separate component that takes a longitude and latitude coordinate as input and converts this to an address.
- TWebGMapsLookupEdit is a separate component that can perform auto completion of address entry via the Google Maps service. It is required to have a Google API key for this.
- TWebGMapsDirectionList is a separate component that can display the different steps of a route with HTML formatting.
- TWebGMapsTimeZone is a separate component that can lookup time zone information based on a longitude & latitude coordinate.

TWebGMaps architecture



The core part of the TMS WebGMaps is the TWebGMaps control, a VCL component integrating the Internet Explorer browser and exposing properties, methods and events to control Google Maps. Additional Google Maps controls can be optionally enabled on the map, i.e. a PanControl (1), a StreetViewControl (2), a ZoomControl (3), a ScaleControl (4), a MapTypeControl (5) and an OverviewmapControl (6).

Different markers (7) can be added to display preferred locations. The marker can display a default balloon or when a valid URL is provided, an image or icon is displayed.

Various events are triggered when the user interacts with keyboard or mouse with the map.

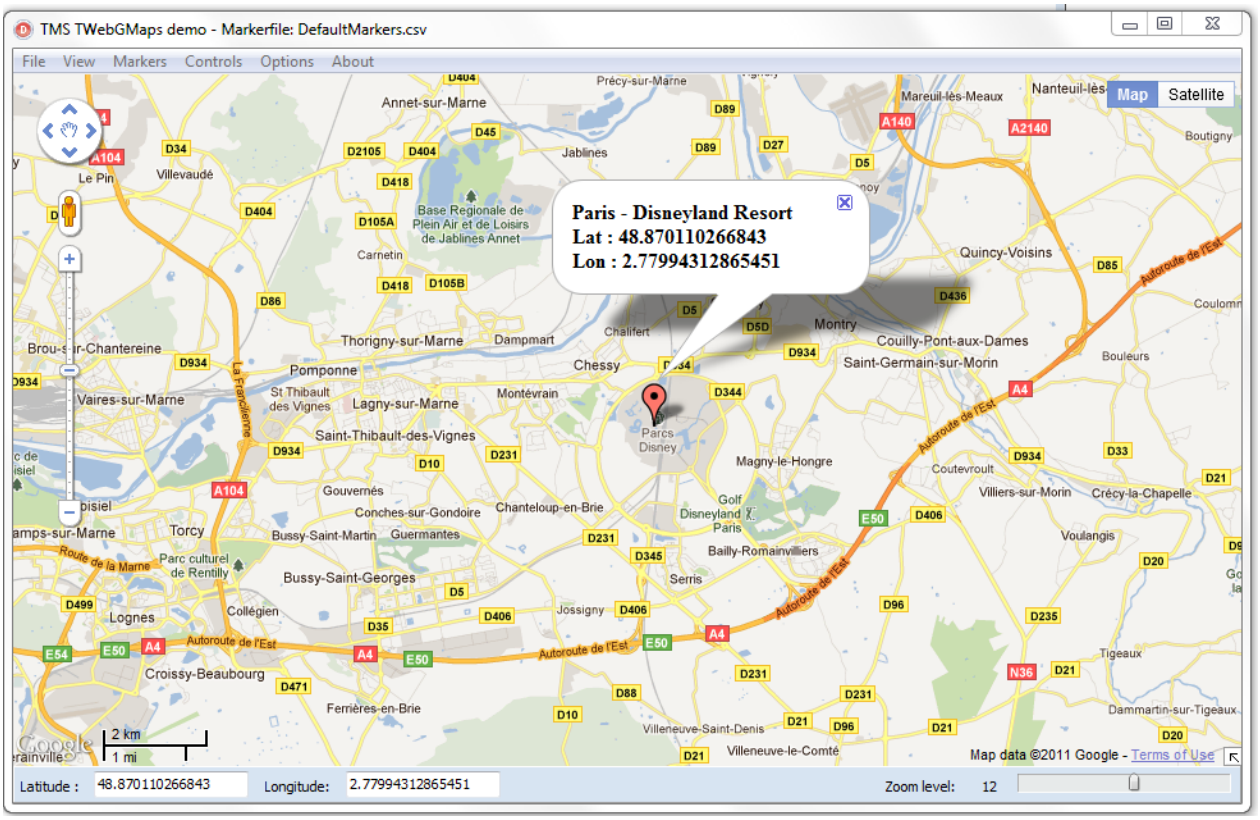
TWebGMaps use

Getting started

From the component palette, select TWebGMaps and drop it on a form. This shows an empty map. The map is only displayed when WebGMaps.Launch is called. The default center location displayed when WebGMaps.Launch is called is set by: WebGMaps.MapOptions.DefaultLongitude,

WebGMaps.MapOptions.DefaultLatitude.

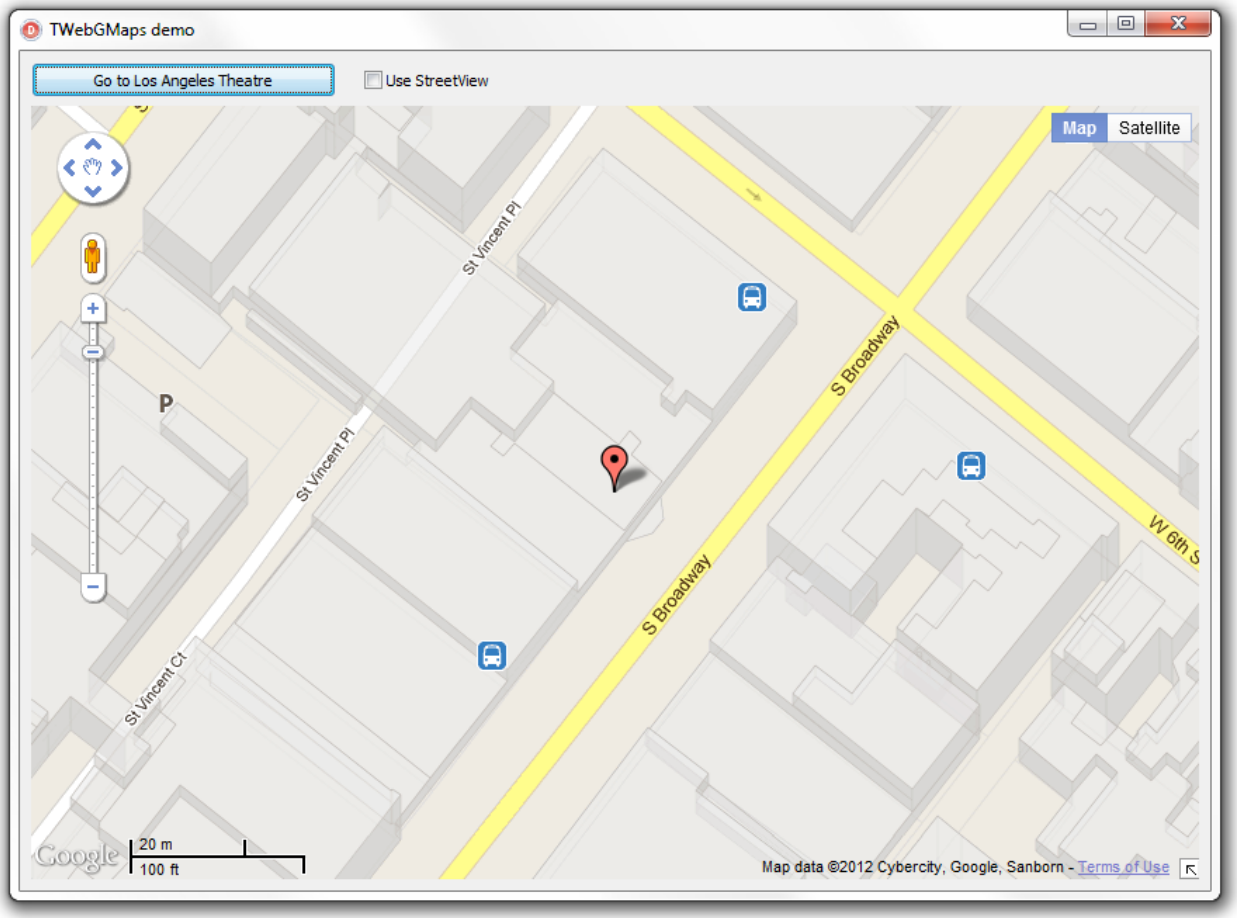
Markers can be added to the map by adding a new entry to the collection WebGMaps.Markers and setting the Marker's properties Longitude & Latitude.



This code snippet sets up the default view of the TWebGMaps to show the Los Angeles Theatre on Broadway at zoom level 19 with coordinates retrieved from the TWebGMapsGeocoding component:

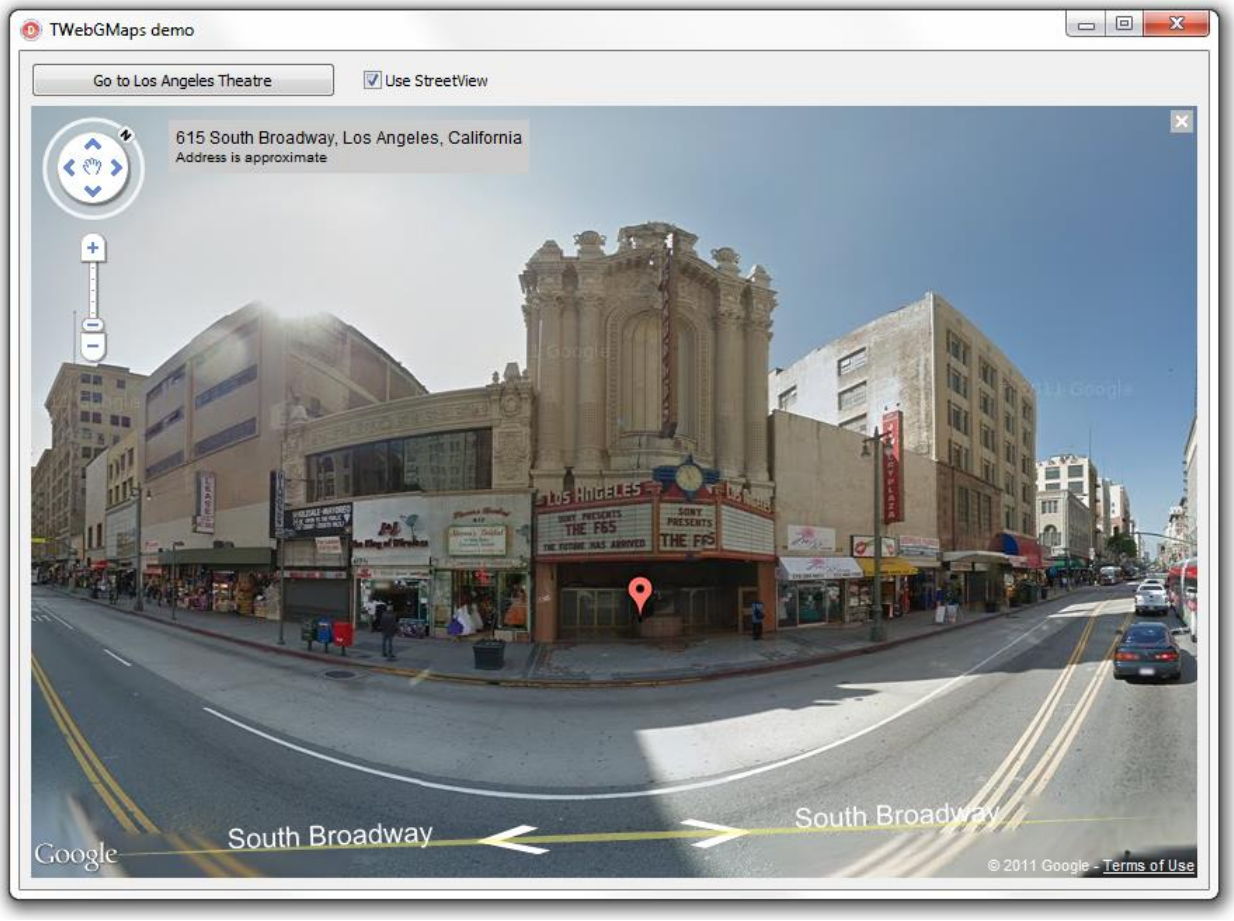
```
begin
  WebGMapsGeocoding1.Address := 'Broadway 615, LOS ANGELES, USA';

  if WebGMapsGeocoding1.LaunchGeocoding = erOk then
  begin
    // center the map at the coordinate
    WebGMaps1.MapOptions.DefaultLatitude :=

      WebGMapsGeocoding1.ResultLatitude;

    WebGMaps1.MapOptions.DefaultLongitude :=
```

```
      WebGMapsGeocoding1.ResultLongitude;

    // Add a marker for the Los Angeles theatre
    WebGmaps1.Markers.Add(WebGMapsGeocoding1.ResultLatitude,
WebGMapsGeocoding1.ResultLongitude,'Broadway theatre');

    // set zoom level
    WebGmaps1.MapOptions.ZoomMap := 19;

    // launch the display of the map
    WebGMaps1.Launch;
  end;
end;
```



Further to this, we can take a look at the Los Angeles theatre by switching the map to StreetView. Following code snippet makes this switch when a checkbox is clicked:

```
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  if checkbox1.Checked then
    WebGmaps1.SwitchToStreetView
  else
    WebGmaps1.SwitchToMap;
end;
```

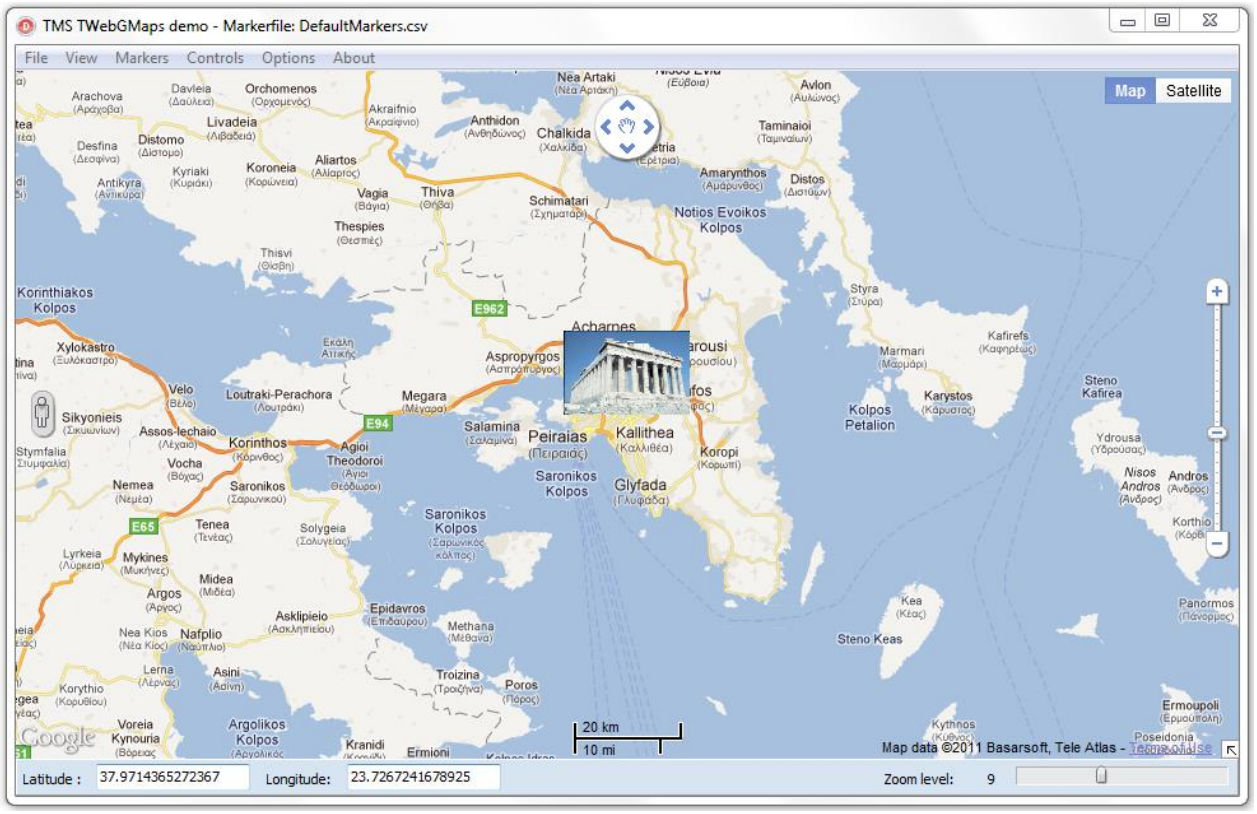


View Types

This gives an overview of the view types that can be set:

- **mtDefault:**

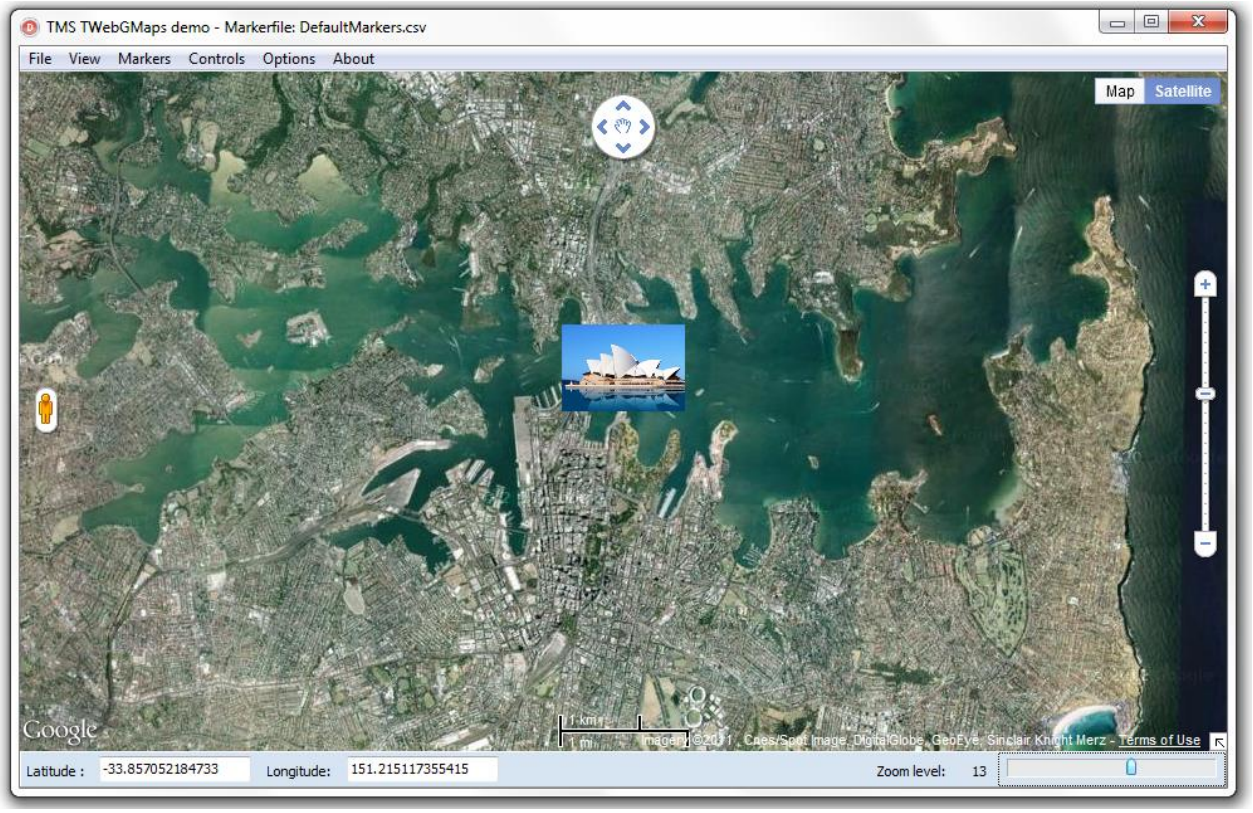
  The screenshot below shows an example of the default map type.



In this sample, the position of the Google Controls has been changed : PanControl to the TopCenter position, ZoomControl to RightCenter, ScaleContol to BottomCenter and StreetviewControl to LeftCenter.

- **mtSatellite:**

  Below is an example of a satellite map type.

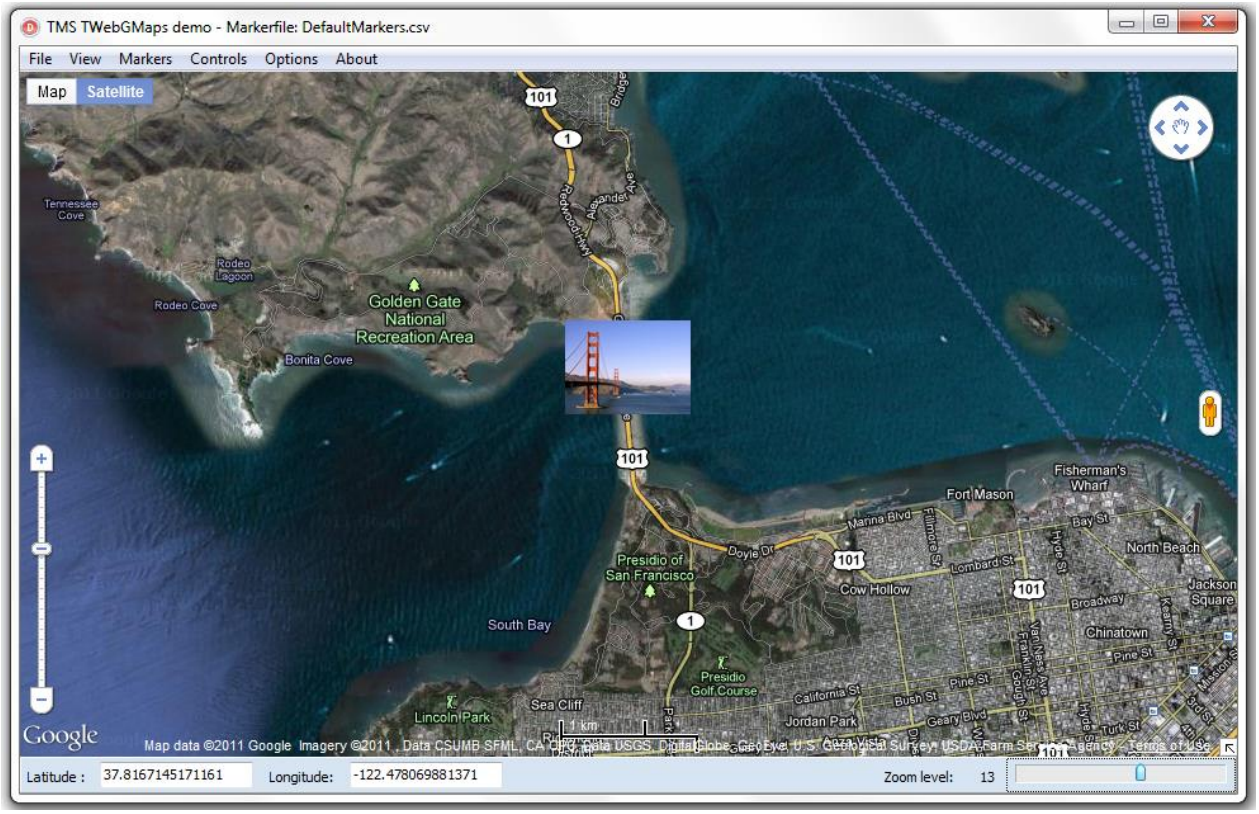


The local image used for the marker was defined with the URL:

file://filepath/Sidney opera house.png.

- **mtHybrid:**

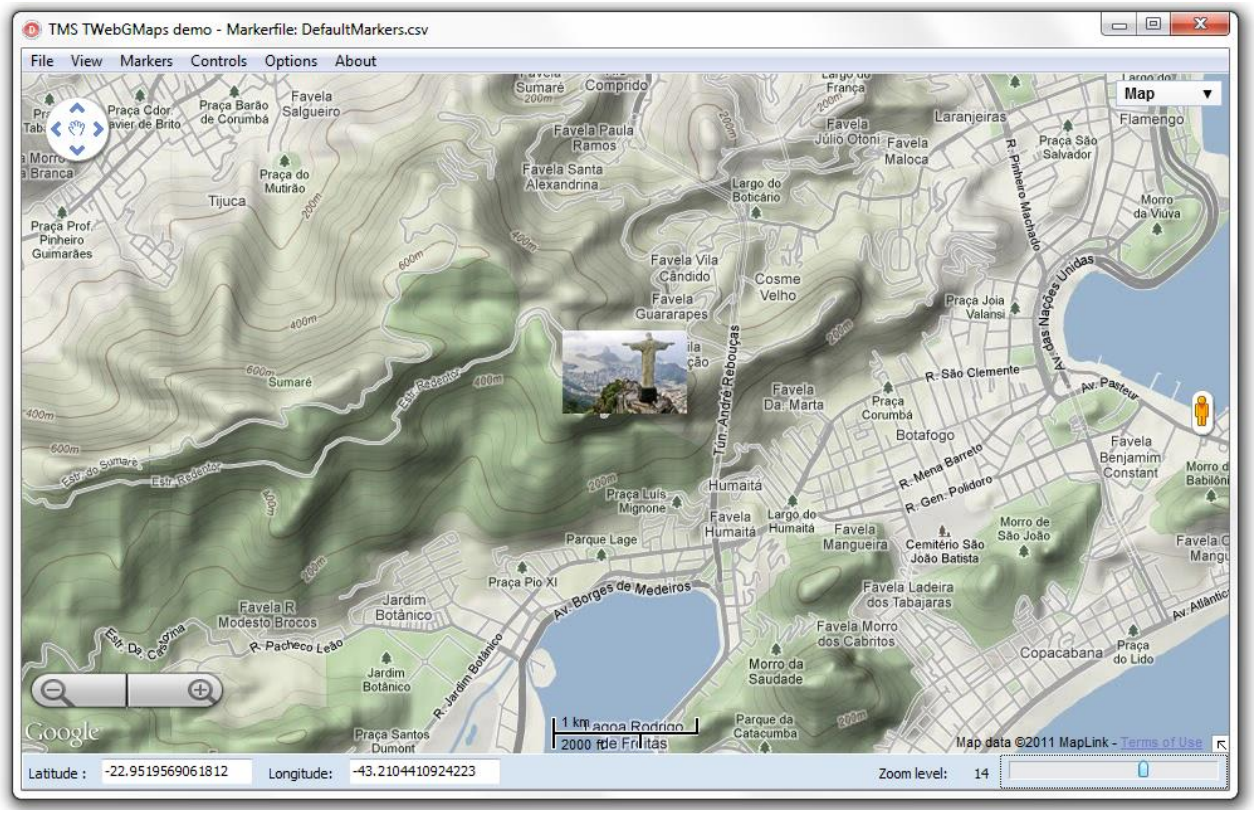
  This example shows the mix of a satellite view with added roadmap information.



The position of the Google Controls has been changed : PanControl to the TopRight position, ZoomControl to LeftBottom and StreetviewControl to RightCenter.

- **mtTerrain:**

  On the screenshot below a Terrain map is shown. This type of map displays a topographic type map, presenting terrain details.



The position of the Google Controls has been changed : PanControl to the TopLeft position, MapTypeControl to TopRight. The ControlsStyle is changed to Android style. Note the zoom control in the bottom left corner displayed in Android style.

General map settings

TWebGMaps properties

- **APIKey:** Optionally specify an API Key to identify the application with the Google Maps API
- **APIChannel**: Optionally specify a Channel ID to identify the application with the Google Maps Premium API. This value is ignored if an API Key value is specified
- **APIClientAuthURL**: Optionally specify the authenticated URL as specified on the Google Maps Premium console. The Default URL is: http://127.0.0.1. This value is ignored if an API Key value is specified
- **APIClientID**: Optionally specify a Client ID to identify the application with the Google Maps Premium API. This value is ignored if an API Key value is specified
- **APIClientSecret**: Optionally specify a Client Secret to identify the application with the Google Maps Premium API. If this value is specified the APISignature value is ignored and the signature is generated automatically based on the Client ID and Client Secret value (Only supported for the GetDirections and GetElevation calls). This value is ignored if a Signature or API Key is specified
- **APISignature**: Optionally specify an API Signature to identify the application with the Google Maps Premium API.
- **AutoLaunch:** Specify if the Map is displayed automatically or if a Launch call is required.
- **ShowDebugConsole:** Optionally display a debugging console at run-time.
- **PolygonLabel:** Configure the appearance of the Polygon/Polyline hints. (If a value is assigned to the TagString property of a Polygon or Polyline, this will be displayed as a hint when the Polygon or Polyline is hovered and PolygonLabel.Visible is set to True)
  - **Color:** Set the color of the label
  - **BorderColor:** Set the border color of the label
  - **Font:** Set the font of the label text (Only the Color, Name and Size are supported)
  - **OffsetTop:** Set the vertical offset of the label in pixels
  - **OffsetLeft:** Set the horizontal offset of the label in pixels
  - **Margin:** Set the margin of the between the label text and the label border
  - **Visible:** Indicates if the label is displayed or not (A non-empty Polygon/Polyline TagString property value is required to display the label)

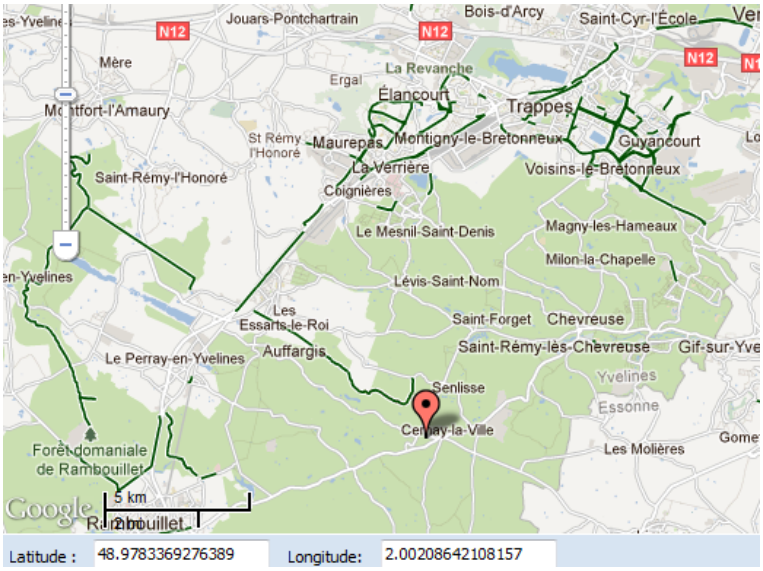
TWebGMaps.MapOptions properties

- **DefaultLatitude:** Sets the latitude value for the default position when Launch is called.
- **DefaultLongitude:** Sets the longitude value for the default position when Launch is called.

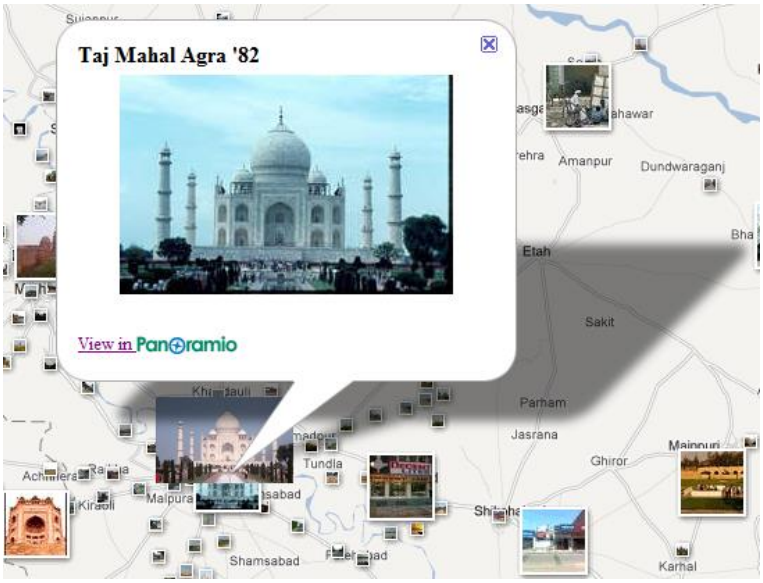
- **DefaultToCurrentLocation:** Sets the current location as the default position when Launch is called. DefaultLatitude and DefaultLongitude are ignored if set to true.
- **DisableControls:** Disables all map controls.
- **DisableDoubleClickZoom:** When set to true, disables zoom functions when double-clicking.
- **DisablePOI:** When set to true, disable display of the points of interest on the map.
- **DisableTilt:** Disable the auto-tilted view on satellite maptype.
  Note: tilted view is only available at specific locations for specific zoom levels, this is a limitation of the Google Maps API.
- **Draggable:** When set to true, the entire map can be moved around in the control.
- **EnableKeyboard:** When set to true, enables the use of the keyboard for controlling panning in the map (or in street view mode).
- **Language:** Defines the language of the Copyright message, and the TWebGMaps.ControlsOptions.MapTypeControl displayed text.
- **MapType:** Sets the type of map. Following values are defined:

  mtDefault: A roadmap is displayed.
  mtSatellite: A satellite view map is displayed.
  mtHybrid: A satellite view map is displayed, along with roadmap information.
  mtTerrain: A topographic map is displayed.

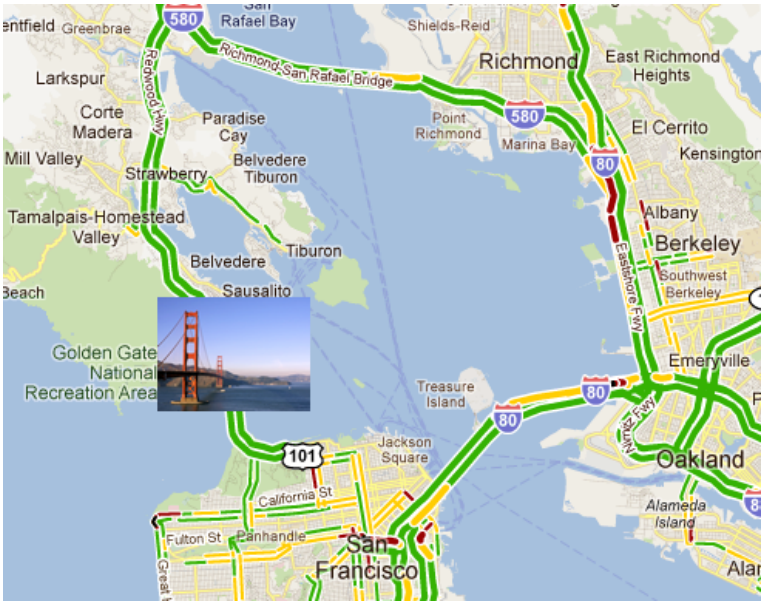
- **PreloaderVisible:** When set to true, an animation while loading the map is displayed.
- **ShowCloud:** **Please note that this service is unfortunately no longer available in the Google Maps API.** When true, shows a cloud layer on top of the map
- **ScrollWheel:** When set to true, enables the use of the scroll wheel. The scroll wheel can be used to zoom in and out on the map.
- **ShowBicycle:** When set to true, and if available in your country, bicycle trail information can be displayed on the map.



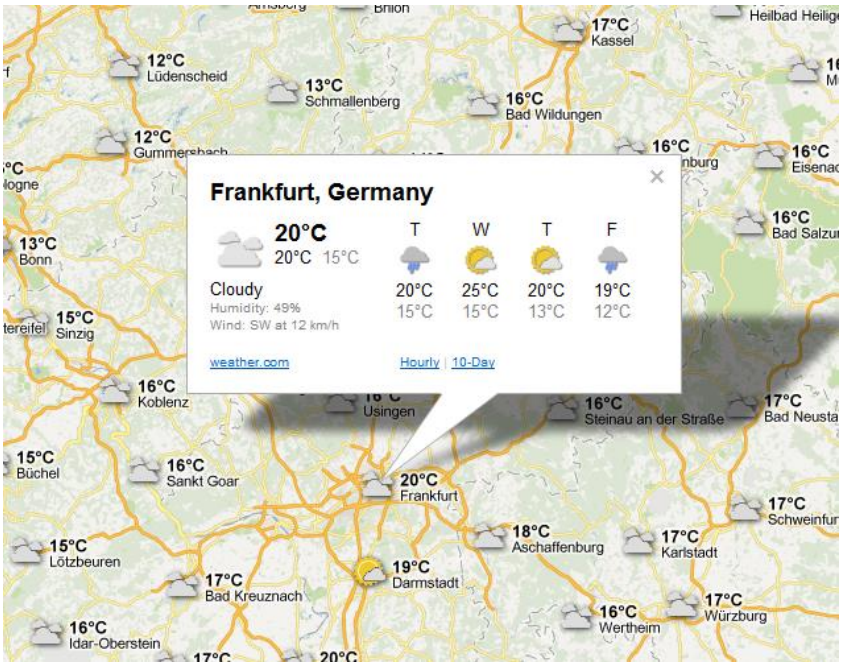
- **ShowPanoramio:** **Please note that this service is unfortunately no longer available in the Google Maps API.** When set to true, the Panoramio functionality is activated, showing thumbnails of posted pictures. The pictures are loaded when the thumbnail is clicked.



- **ShowTraffic:** When set to true, and if available in your country, traffic information can be displayed. Check for availability on: https://spreadsheets.google.com/spreadsheet/pub?key=0Ah0xU81penP1cDlwZHdzYWkyaERNc0xrWHNvTTA1S1E&gid=0 .



- **ShowWeather:** **Please note that this service is unfortunately no longer available in the Google Maps API.** when true, shows the weather conditions & option to click to show weather forecast on the map.



- **ZoomMap:** Is to be used to set the default zoom at startup. The zoom value is a value between 1 and 21 with 21 being the highest zoom level.
- **ZoomMarker:** Sets the zoom behavior of a marker icon. When set to zmNone there is no zooming. When set to zmToggle the marker icon is zoomed in when clicked and zoomed out when clicked again or when a click occurs outside the marker icon. When set to zmClick the marker icon is zoomed in when clicked and zoomed out when clicked again. To be used in combination with the Marker's Width, Height, ZoomWidth, ZoomHeight properties.

TWebGMaps.StreetViewOptions properties

- **DefaultLatitude:** Sets the latitude value for the default street view position when StreetView is launched.
- **DefaultLongitude:** Sets the longitude value for the default street view position when StreetView is launched.
- **Heading:** Defines the heading at the street view position. Valid values are between 0 and 360 degrees.
- **Pitch:** Defines the pitch (view angle) for the street view. Valid values are between -90 and 90 degrees.
- **Visible:** When set to true, the street view is displayed.
- **Zoom:** Sets the zoom factor for the street view. Valid values are between 0 and 5.

TWebGMaps.WeatherViewOptions properties

- **LabelColor:** Sets the background color of the weather info balloons as either white or black.
- **ShowInfoWindows:** Enables to show balloons with weather info/forecast when clicked.
- **TemperatureUnit:** Sets the unit of temperature to Celcius or Fahrenheit.
- **WindspeedUnit:** Sets the unit of wind speed to kilomters per hour or miles per hour or metres per second.

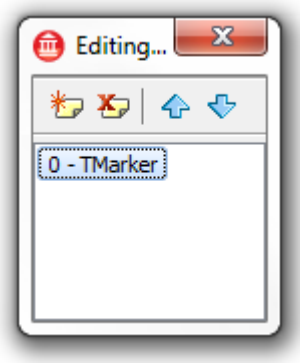
Map markers

TMarkers is a collection of marker items giving the possibility to highlight certain locations on the map. A marker is either a default balloon or can be set to a custom icon by defining the URL for it. The example below shows a mix of pictures and a standard Google balloon marker. A sample on how to create a marker info window can be found in the samples paragraph.

On the left hand side, a PNG image is used as marker. In the middle section, two JPEG images are displayed as marker. On the right hand side, a standard Google Maps balloon marker is used, as the marker was created with an empty icon property.

Adding markers

First open the markers collection editor by clicking the TWebGMaps.Markers property in the Object Inspector. From here, markers can be added or removed.



The equivalent in code is:

Adding a marker:

```
var

begin

  WebGMaps1.Markers.Add(aLatitude, aLongitude, aMarkerTitle, anIcon,
  aDraggable, aVisible, aClickable, aFlat, anInitialDropAnimation);

end;
```

TWebGMaps.Markers properties

- **Clickable:** When set to true, enables clicking on the marker. Clicking opens an extra info window on the Google Maps containing the text set by Marker.Title.
- **Cluster: TMapCluster:** Assign a TMapCluster object to this property to link the marker to a Cluster in the Clusters collection.
- **Draggable:** When set to true, the marker can be moved around the map when dragged.
- **Flat:** When set to true, the marker is drawn as a flat image on the map. Otherwise the marker is drawn as a 3D image with a shadow.
- **Icon:** Allows the use of an image as marker. This can also be a picture when the url to that image is defined. An example can be found in the samples paragraph.

- **IconColor:** Allows changing the color of the default marker icon to one of the available pre-defined colors: icDefault, icBlue, icGreen, icRed and icPurple (Only available if the Icon property is not assigned).
- **IconHeight:** Specify a custom height value in pixels for the marker icon when the IconState is msDefault. Can only be used when the Icon property is assigned. If this value is not specified the icon is displayed in its full size. Set the icon zoom behavior with the MapOptions.ZoomMarker property.
- **IconState:** Sets the state of the marker icon to zoomed in or zoomed out. Should be used in combination with IconWidth, IconHeight and IconZoomWidth, IconZoomHeight.
- **IconWidth:** Specify a custom width value in pixels for the marker icon when the IconState is msDefault. Can only be used when the Icon property is assigned. If this value is not specified the icon is displayed in its full size. Set the icon zoom behavior with the MapOptions.ZoomMarker property.
- **IconZoomHeight:** Specify a custom height value in pixels for the marker icon when the IconState is msZoomedIn. Can only be used when the Icon property is assigned. If this value is not specified the icon is displayed in its full size. Set the icon zoom behavior with the MapOptions.ZoomMarker property.
- **IconZoomWidth:** Specify a custom width value in pixels for the marker icon when the IconState is msZoomedIn. Can only be used when the Icon property is assigned. If this value is not specified the icon is displayed in its full size. Set the icon zoom behavior with the MapOptions.ZoomMarker property.
- **InitialDropAnimation:** When set to True, the marker is dropped with an animation effect on the map when displayed.
- **Latitude:** Sets the latitude value of the marker on the map.
- **Longitude:** Sets the longitude value of the marker on the map.
- **MapLabel:** Allows the use of a HTML label displayed on top of the marker. The label is automatically resized based on the Text value. A sample on how to create a custom label for a marker can be found in the samples paragraph.
    - **BorderColor:** The border color of the label.
    - **Color:** The color of the label.
    - **Font.Name:** The font name for the label text.
    - **Font.Size:** The font size for the label text.
    - **Font.Color:** The font color for the label text.
    - **Margin:** The margin in pixels between the label border and the label text.

  - **Text:** The text displayed in the label. If this value is empty, no label is displayed.
  - **OffsetLeft:** The left offset of the label relative to the marker coordinates. This is a percentage value. For example the value 0 will center align the label, the value 50 will right align the label.
  - **OffsetTop:** The top offset of the label relative to the marker coordinates. This is a pixel value. With a default Font.Size and the default Marker the label is displayed on top of the Marker.

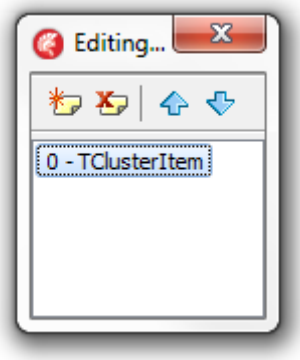
- **Title:** Sets the title for the marker. This marker title will be displayed in the hint, when hovering over the marker and in the info window when the marker is clicked.
- **Visible:** When set to true, the marker is shown on the map.
- **Text:** Sets the character that is displayed inside the marker instead of the default “dot”. Note: when using the default marker this is only supported when the IconColor property is set to icDefault.

Map clusters

TClusters is a collection of Cluster items to create and manage per-zoom-level clusters for large amounts of markers.

Adding clusters

First open the clusters collection editor by clicking the TWebGMaps.Clusters property in the Object Inspector. From here, clusters can be added or removed.



The equivalent in code is:

Adding a cluster:

```
var

begin

  WebGMaps1.Clusters.Add()

end;
```

Assigning markers to a cluster and display the cluster on the map:

```
var

  m: TMarker;

begin

    m := WebGMaps1.Markers.Add(Latitude, Longitude, 'Title');
    m.Cluster := WebGMaps1.Clusters[0].Cluster;
    WebGMaps1.CreateMapCluster(WebGMaps1.Clusters[0].Cluster);
```

Adding a marker to an existing cluster:

```
var

  m: TMarker;

begin

    m := WebGMaps1.Markers.Add(Latitude, Longitude, 'Title');
    m.Cluster := WebGMaps1.Clusters[0].Cluster;
    WebGMaps1.AddMarkerToCluster(WebGMaps1.Clusters[0].Cluster, m);
```

Removing a marker from an existing cluster:

```
  m := WebGMaps1.Markers[WebGMaps1.Markers.Count - 1];
  m.Cluster := nil;
  WebGMaps1.DeleteMarkerFromCluster(WebGMaps1.Clusters[1].Cluster,
m);
```

Updating an existing cluster:

```
var

  c: TMapCluster;

begin

  c := WebGMaps1.Clusters[0].Cluster;
  c.Title := 'New Title';
  c.ZoomOnClick := not c.ZoomOnClick;
  WebGMaps1.UpdateMapCluster(c);
```

Remove a single cluster or all clusters from the map:

```
  WebGMaps1.DeleteMapCluster(0);

  WebGMaps1.DeleteAllMapCluster;
```

Notes:

- Removing a cluster from the map will also remove all markers assigned to that cluster from the map.
- The clusters and markers are removed from the map, but not from the Clusters and Markers collection of the control.
- The calls to add/update/delete a cluster can only be used after the OnDownloadFinish event was triggered.

TWebGMaps.Clusters properties

- **AverageCenter: (Boolean)** Indicates if the position of a cluster marker should be the average position of all markers in the cluster. If set to false, the cluster marker is positioned at the location of the first marker added to the cluster.
- **BatchSize: (Integer)** The number of markers to be processed in a single batch.
- **Calculator: (TStringList)** The JavaScript function used to determine the text to be displayed on a cluster marker and the index indicating which style to use for the cluster marker. By default the number of markers contained in the cluster is displayed on the cluster marker. Cluster markers containing 2-9 markers use the first style, 10-99 markers use the second style and +100 markers use the third style. Custom styles can be configured in the Styles collection.

  The standard JavaScript function:

```
function(markers, numStyles) {
 var index = 0;
 var title = "";
```

```
  var count = markers.length;
  var dv = count;
  while (dv !== 0) {
   dv = parseInt(dv / 10, 10);
   index++;
  }

  index = Math.min(index, numStyles);
  return {
   text: count,
   index: index,
   title: title
  };
 };
```

- **ClusterClass: (string)** The name of the CSS class defining general styles for the cluster markers. Use this class to define CSS styles that are not available in the TClusterStyle properties. The CSS class code can be added using the OnInitHTML event.

  Example:

```
procedure TForm7.WebGMaps1InitHTML(Sender: TObject; var HTML: string);
begin
  HTML := HTML + '<style>.cluster{background-color:gray;}</style>'
end;
```

- **GridSize: (Integer)** The grid size of a cluster in pixels.
- **IgnoreHidden: (Boolean)** Indicates if hidden markers (Visible := False) are ignored.
- **MaxZoom: (Integer)** The maximum map zoom level at which clustering is enabled.
- **MinimumClusterSize: (Integer)** The minimum number of markers required in a cluster before the markers are hidden and a cluster marker appears.
- **Styles: (TClusterStyles)**

  Cluster markers containing 2-9 markers use the first style, 10-99 markers use the second style and 100 or more markers use the third style. Different criteria can be configured by adding a custom JavaScript function to the Calculator property.

  - **BackgroundPositionX (Integer)** The left position of the cluster icon image defined in the IconURL property. This property should be used if the image is a sprite that contains multiple images.
  - **BackgroundPositionY (Integer)** The top position of the cluster icon image defined in the IconURL property. This property should be used if the image is a sprite that contains multiple images.
  - **Font (TFont)** The font used for the cluster marker text.
  - **IconHeight (Integer)** The height of the cluster marker image.

- **IconOffsetX (Integer)** The left position offset of the cluster marker icon relative to the cluster location.
- **IconOffsetY (Integer)** The top position offset of the cluster marker icon relative to the cluster location.
- **IconURL (string)** The URL of the icon image.
- **IconWidth (Integer)** The width of the cluster marker image.
- **TextOffsetX (Integer)** The left position offset of the cluster marker text relative to the cluster location.
- **TextOffsetY (Integer)** The top position offset of the cluster marker text relative to the cluster location.

  Note: The IconURL, IconHeight and IconWidth properties are required when adding a style. Style items that have an empty value for IconURL or a 0 value for IconHeight/IconWidth will not function.

- **Title: (String)** The tooltip text to display for the cluster marker.
- **ZoomOnClick (Boolean)** Indicates if the map is zoomed to the bounds of the markers when a cluster marker is clicked.

TWebGMaps.Clusters methods

- **FitMapToMarkers(): Boolean;**
  Fits the map to the bounds of the markers managed by the cluster.

Map directions

TDirections is a collection of routes based on a start location and destination.

Retrieving directions

```
WebGMaps1.GetDirections(aOrigin, aDestination, aAlternatives, aTravelMode,
aUnits, aLanguage, aAvoidHighways, aAvoidTolls, aWaypoints,
aOptimizeWaypoints);
```

The Directions collection will automatically be filled with all available routes for the given parameters.

- **aOrigin:** (String) The start location.
- **aDestination:** (String) The destination location.

- **aAlternatives:** (Boolean) When set to true all available routes will be added to the Directions collection. When set to false only the default route will be added to the Directions collection.
- **aTravelMode:** (TTravelMode) Sets which travel mode should be used to calculate the directions.
    - **tmBicycling**
    - **tmDriving**
    - **tmWalking**
- **aUnits:** (TUnits) Sets which unit system to use for the DistanceText values.
    - **usMetric:** DistanceText values are returned using kilometers and meters.
    - **usImperial:** DistanceText values are returned using miles and feet.
- **aLanguage:** (TLanguageName) Sets the language to use for the Directions[].Legs[].Steps[].Instructions text values.
- **aAvoidHighways:** Restrict results to routes without highways
- **aAvoidTolls:** Restrict results to routes without tolls
- **aWayPoints:** List of additional locations. Waypoints allow you to calculate routes through additional locations, in which case the returned route passes through the given waypoints.
- **aOptimizeWaypoints:** Allow the Directions service to optimize the provided route by rearranging the waypoints in a more efficient order

TWebGMaps.Directions properties

- **Bounds:** Sets the bounds of a route.
    - **NorthEast:** Sets the latitude/longitude of the north east corner of the route.
        - **Latitude**
        - **Longitude**
    - **SouthWest:** Sets the latitude/longitude of the south west corner of the route.
        - **Latitude**

        - **Longitude**

- **Copyrights:** (readonly) Copyrights text to be displayed for this route.
- **Legs:** Contains information about this route and the steps of which it is composed.
    - **Distance:** The total distance of the leg in meters.
    - **DistanceText:** The text value of the total distance of the leg. If the usMetric parameter value is used in the GetDirections call then this value is specified in kilometers/meters. If the usImperial parameter value is used, the value is specified in miles/feet.
    - **Duration:** The typical required time for this leg in seconds.
    - **DurationText:** The typical required time for this leg specified in hours/minutes.
    - **EndAddress:** The address of the destination of this leg.
    - **EndLocation:** The geocoded destination of this leg.
        - **Latitude**
        - **Longitude**
    - **StartAddress:** The address of the origin of this leg.
    - **StartLocation:** The geocoded origin of this leg.
        - **Latitude**
        - **Longitude**
    - **Steps:**
        - **Distance:** The distance covered by this step in meters.
        - **DistanceText:** The text value of the distance covered by this step. If the usMetric parameter value is used in the GetDirections call then this value is specified in kilometers/meters. If the usImperial parameter value is used, the value is specified in miles/feet.
        - **Duration:** The typical required time to perform this step in seconds.
        - **DurationText:** The typical required time to perform this step specified in hours/minutes.
        - **EndLocation:** The ending location of this step.
            - **Latitude**

        - **Longitude**
    - **Instructions:** Instructions text for this step.
    - **Polyline:** A polyline describing the course for this step.
        - See Map Polylines for details.
    - **StartLocation:** The starting location of this step.
        - **Latitude**
        - **Longitude**
    - **TravelMode:** The mode of travel used in this step.
        - **tmBicycling**
        - **tmDriving**
        - **tmWalking**
    - **WayPointIndex:** Contains the index of the item in the list of WayPoints provided with the GetDirections call that corresponds with this Leg

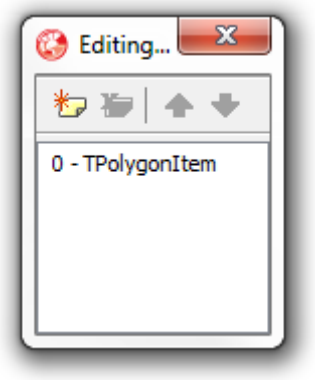
- **Polyline:** A polyline that represents the entire course of this route. The path is simplified in order to make it suitable in contexts where a small number of vertices is required.
    - See Map Polylines for details.
- **Summary:** Descriptive text for this route.

Map polygons

WebGMaps.Polygons is a collection of closed lines giving the possibility to highlight certain regions on the map. The screenshot below shows a circle around Paris.

Adding polygons

First open the polygons collection editor by clicking the TWebGMaps.Polygons property in the Object Inspector. From here, polygons can be added or removed.



The equivalent in code is:

Adding a polygon:

```
var
  PolygonItem: TPolygonItem;

begin
  PolygonItem := WebGMaps1.Polygons.Add;
  PolygonItem.Polygon.BackgroundOpacity := 50;
  PolygonItem.Polygon.BorderWidth := 2;

  //Settings for a Polygon of type Circle
  PolygonItem.Polygon.PolygonType := ptCircle;
  PolygonItem.Polygon.Radius := 10000;
  PolygonItem.Polygon.Center.Latitude := 50;
  PolygonItem.Polygon.Center.Longitude := 2;

  //Settings for a Polygon of type Rectangle
  PolygonItem.Polygon.PolygonType := ptRectangle;
  PolygonItem.Polygon.Bounds.NorthEast.Latitude := 52;
  PolygonItem.Polygon.Bounds.NorthEast.Longitude := 4;
  PolygonItem.Polygon.Bounds.SouthWest.Latitude := 50;
  PolygonItem.Polygon.Bounds.SouthWest.Longitude := 3;

  //Settings for a Polygon of type Path
  PolygonItem.Polygon.PolygonType := ptPath;
  PolygonItem.Polygon.Path.Add(50, 2);
  PolygonItem.Polygon.Path.Add(52, 4);
  PolygonItem.Polygon.Path.Add(50, 4);

  WebGMaps1.CreateMapPolygon(PolygonItem.Polygon);

end;
```

Editing a polygon:

```
PolygonItem.Polygon.Visible := not PolygonItem.Polygon.Visible;

WebGMaps1.UpdateMapPolygon(PolygonItem.Polygon);
```

Removing a polygon:

```
//Removes the Polygon from the map
WebGMaps1.DeleteMapPolygon(PolygonItem.Index);

//Removes the Polygon from the collection
WebGMaps1.Polygons.Delete(PolygonItem.Index);
```

TWebGMaps.Polygons properties

- **BackgroundColor:** The color of the polygon.
- **BackgroundOpacity:** The opacity of the polygon.
- **BorderColor:** The border color of the polygon.
- **BorderOpacity:** The border opacity of the polygon.
- **BorderWidth:** The width of the polygon border in pixels.
- **Bounds:** Sets the bounds of a polygon when PolygonType is set to ptRectangle.
    - **NorthEast:** Sets the latitude/longitude of the north east corner of the rectangle
        - **Latitude**
        - **Longitude**
    - **SouthWest:** Sets the latitude/longitude of the south west corner of the rectangle
        - **Latitude**
        - **Longitude**
- **Center:** Sets the latitude/longitude of the center point of the circle when PolygonType is set to ptCircle.

  - **Latitude**
  - **Longitude**

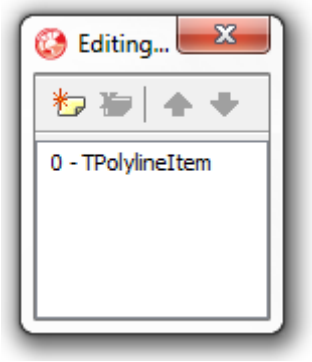
- **Clickable:** When set to true, enables clicking on the polygon.
- **Editable:** When set to true, the polygon can be edited.
- **Geodesic:** When set to true, each edge is rendered as a geodesic. When set to false, render each edge as a straight line.
- **HoverBackgroundColor:** The color of the polygon when hovered
- **HoverBorderColor:** The border color of the polygon when hovered
- **Path:** The ordered sequence of coordinates of the polygon that forms a closed loop (when PolygonType is set to ptPath). Paths are closed automatically.
  - **Latitude:** Sets the latitude value of the polygon path item on the map.
  - **Longitude:** Sets the longitude value of the polygon path item on the map.
- **PolygonType:** Sets the type of polygon to be rendered.
  - **ptCircle:** Renders a circle based on the Radius and Center property values.
  - **ptPath:** Renders a polygon based on the list of Path coordinates.
  - **ptRectangle:** Renders a rectangle based on the Bounds property values.
- **Radius:** The radius of the polygon in meters. (When PolygonType is set to ptCircle)
- **TagString:** The text associated with the polygon (optional). The appearance of the hint can be configured with the PolygonLabel properties. If PolygonLabel.Visible is set to true, this value will be displayed as a hint when hovering the polygon on the map.
- **TagObject:** The object associated with the polygon (optional)
- **Visible:** When set to true, the polygon is shown on the map.
- **Zindex:** The zIndex compared to other elements on the map.

Map polylines

WebGMaps.Polylines is a collection of lines giving the possibility to highlight certain routes on the map. The screenshot below shows a route between a start and end location.

Adding polylines

First open the polylines collection editor by clicking the TWebGMaps.Polylines property in the Object Inspector. From here, polylines can be added or removed.



The equivalent in code is:

Adding a polyline:

```
var
  PolylineItem: TPolylineItem;

begin
  PolylineItem := WebGMaps1.Polylines.Add;
  PolylineItem.Polyline.Width := 2;
  PolylineItem.Polyline.Path.Add(50, 2);
  PolylineItem.Polyline.Path.Add(52, 4);
  PolylineItem.Polyline.Path.Add(50, 4);

  WebGMaps1.CreateMapPolyline(PolylineItem.Polyline);

end;
```

Editing a polyline:

```
PolylineItem.Polyline.Visible := not PolylineItem.Polyline.Visible;

WebGMaps1.UpdateMapPolyline(PolylineItem.Polyline);
```

Removing a polyline:

```
//Removes the Polyline from the map
WebGMaps1.DeleteMapPolyline(PolylineItem.Index);

//Removes the Polyline from the collection
WebGMaps1.Polylines.Delete(PolylineItem.Index);
```

TWebGMaps.Polylines properties

- **Clickable:** When set to true, enables clicking on the polyline.
- **Color:** The color of the polyline.
- **Editable:** When set to true, the polyline can be edited.
- **Geodesic:** When set to true, each edge is rendered as a geodesic. When set to false, render each edge as a straight line.
- **HoverColor:** The color of the polyline when hovered.
- **Icons:** The list of icons to be rendered along the polyline.
    - **SymbolType:** The type of icon that is displayed on the polyline.
    - **Offset:** The distance from the start of the line at which an icon is to be rendered. Can be set as a percentage (OffsetType set to dtPixels) or in pixels (OffsetType set to dtPercentage).
    - **OffsetType:** Defines the way the Offset value is used.
        - **ctPercentate:** The Offset is handled as a percentage value.
        - **ctPixel:** The Offset is handled as a pixel value.
    - **RepeatValue:** The distance between consecutive icons on the line. Can be set as a percentage (RepeatType set to dtPixels) or in pixels (RepeatType set to dtPercentage).
    - **RepeatType:**
        - **ctPercentate:** The RepeatValue is handled as a percentage value.
        - **ctPixel:** The RepeatValue is handled as a pixel value.
    - **FixedRotation:** If set to true, each icon in the sequence has the same fixed rotation regardless of the angle of the edge on which it lies. If set to false, each icon in the sequence is rotated to align with its edge.
- **Opacity:** The opacity of the polyline.
- **Path:** The ordered sequence of coordinates of the polyline.
    - **Latitude:** Sets the latitude value of the polyline path item on the map.
    - **Longitude:** Sets the longitude value of the polyline path item on the map.

- **TagString:** The text associated with the polyline (optional). The appearance of the hint can be configured with the PolygonLabel properties. If PolygonLabel.Visible is set to true, this value will be displayed as hint when hovering the polyline on the map.
- **TagObject:** The object associated with the polyline (optional)
- **Visible:** When set to true, the polyline is shown on the map.
- **Width:** The width of the polyline in pixels.
- **Zindex:** The zIndex compared to other elements on the map.

Map ControlsOptions

The ControlsOptions class property bundles various settings for controlling the appearance and behaviour of various controls in the map.

TWebGMaps.ControlsOptions properties

- **ControlsType:** Sets the control to one of the defined types: ctAndroid, ctDefault, ctSmall or ctZomPan.
  - **ctAndroid**: Changes the zoom control to Android style.

  - **ctDefault:** Sets all controls to Google style controls. Please check ZoomControl zsLarge for an image.
  - **ctSmall:** Sets the zoom control to a small format control. Please check ZoomControl zsSmall for an image.
  - **ctZoomPan:** Sets the zoom control to the standard Google Maps zoom control.

- **MapTypeControl:** Defines the settings for the MapType control that allows choosing another map view from within the actual map.

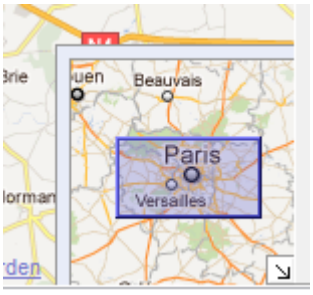
- **Position:** Sets the TWebGMaps.ControlsOptions.MapTypeControl.Position to one of these predefined choices: cpBottomLeft, cpBottomCenter, cpBottomRight, cpLeftBottom, cpLeftCenter, cpLeftTop, cpRightBottom, cpRightCenter, cpRightTop, cpTopCenter, CpTopLeft, cpTopRight.
- **Style:** Sets the TWebGMaps.ControlsOptions.MapTypeControl.Style to one of these predefined choices: mtsDefault, mtsDropDownMenu or mtsHorizontalBar.

  mtsDefault: For more info, check mtsHorizontalBar.

  mtsDropDownMenu: Displays the maptype control as a drop down menu. When choosing Map, the possibility is offered to draw the map in Terrain mode. When Satellite is chosen, the possibility is offered to show map details (Hybrid mode).

  mtsHorizontalBar: Displays the maptype control as a horizontal bar. When Map is clicked, the possibility is offered to draw the map in Terrain mode. When Satellite is chosen, the possibility is offered to show map details (Hybrid mode).
- **Visible:** When set to true, the MapType control is drawn on the map.

- **OverviewMapControl:** Defines the settings for the overview map control that shows a larger area, with the actual view displayed in transparent blue. When holding the mouse down in this blue area, and moving within the overviewmap control, the map can be panned to another location.



- **Open:** When set to true, the overview map is shown is displayed in the right bottom corner of the map. When set to false, an arrow control is drawn that opens the overview map control when clicked.
- **Visible:** When set to true, the OverviewMap control is drawn on the map.

- **PanControl:** Sets the pan control that allows panning the actual view of the map within the control, by clicking the arrow keys (up, down, left, right).

- **Position:** Sets the TWebGMaps.ControlsOptions.PanControl.Position to one of these predefined choices: cpBottomLeft, cpBottomCenter, cpBottomRight, cpLeftBottom, cpLeftCenter, cpLeftTop, cpRightBottom, cpRightCenter, cpRightTop, cpTopCenter, CpTopLeft, cpTopRight.
- **Visible:** When set to true, the Pan control is drawn on the map.

- **RotateControl:** Sets the rotate control that allows rotating the map and switching between tilted and default view on satellite map type.
Note: Only available on satellite map type for specific locations and specific zoom levels.

- **Position:** Sets the TWebGMaps.ControlsOptions.RotateControl.Position to one of these predefined choices: cpBottomLeft, cpBottomCenter, cpBottomRight, cpLeftBottom, cpLeftCenter, cpLeftTop, cpRightBottom, cpRightCenter, cpRightTop, cpTopCenter, CpTopLeft, cpTopRight.
- **Visible:** When set to true, the Rotate control is drawn on the map.
Note: the control can only be visible on satellite map type for specific locations and specific zoom levels.

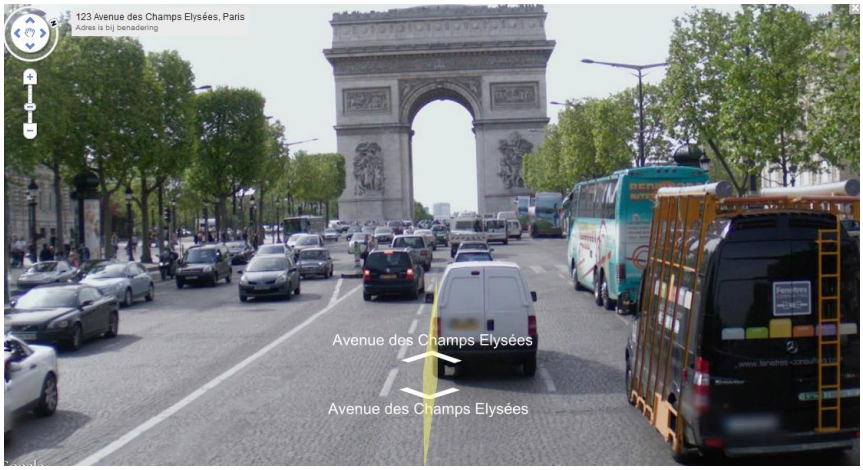
- **ScaleControl:** Defines the settings for the Scale control that shows the actual scale of the view in the control. When the Google logo is clicked, the actual view is opened in Google Maps in a web browser page.

- **Position:** Sets the TWebGMaps.ControlsOptions.ScaleControl.Position to one of these predefined choices: cpBottomLeft, cpBottomCenter, cpBottomRight, cpLeftBottom, cpLeftCenter, cpLeftTop, cpRightBottom, cpRightCenter, cpRightTop, cpTopCenter, CpTopLeft, cpTopRight.
- **Visible:** When set to true, the Scale control is drawn on the map.

- **StreetViewControl:** Defines the settings for the StreetView control that allows opening a 3D photo view of the area chosen by dragging the control to the wanted position. When the icon is greyed-out, the streetview mode is not available for that place.

- **Position:** Sets the TWebGMaps.ControlsOptions.StreetViewControl.Position to one of these predefined choices: cpBottomLeft, cpBottomCenter, cpBottomRight, cpLeftBottom, cpLeftCenter, cpLeftTop, cpRightBottom, cpRightCenter, cpRightTop, cpTopCenter, CpTopLeft, cpTopRight.
- **Visible:** When set to true, the StreetView control is drawn on the map.

  Example of a street view image**:**



- **ZoomControl:** Defines the settings for the Zoom control that allows to zoom in on the actual view of the map, or to zoom out to a larger area. The center position on the screen is used as zooming location.
    - **Position:** Sets the TWebGMaps.ControlsOptions.ZoomControl.Position to one of these predefined choices: cpBottomLeft, cpBottomCenter, cpBottomRight, cpLeftBottom, cpLeftCenter, cpLeftTop, cpRightBottom, cpRightCenter, cpRightTop, cpTopCenter, CpTopLeft, cpTopRight.
    - **Style:** Sets the TWebGMaps.ControlsOptions.ZoomControl.Style to one of these predefined choices: zsDefault, zsLarge or zsSmall.

      zsDefault: Same as type zsLarge.

      zsLarge: Displays the zoom control with divisions.

      zsSmall: Displays the zoom control as two buttons, plus and minus.

    - **Visible:** When set to true, the Zoom control is drawn on the map.

Map elevations

Elevations allow obtaining the elevation data of a latitude and longitude coordinate or along a path of coordinates.

- **function GetElevation(Latitude, Longitude: double): boolean;**
  Retrieves the elevation data for a single latitude and longitude coordinate.
  The data is added to the Elevations collection. Returns true if the request succeeded, false otherwise.

- **function GetElevation(Path: TPath; ResultCount: integer = 2): boolean;**
  Retrieves the elevation data for a Path that contains latitude and longitude coordinates. The start location (first coordinate in Path) and end location (last coordinate in path) are used to form a straight line. The elevation data is retrieved along the straight path at specified intervals. The number of intervals is indicated by the ResultCount parameter.
  The data is added to the Elevations collection. Returns true if the request succeeded, false otherwise.

- **Elevations: TElevations;**
  Collection containing the result(s) of the GetElevation call.
  - **Latitude:** The latitude coordinate for the elevation data
  - **Longitude:** The longitude coordinate for the elevation data
  - **Elevation:** The elevation of the location in meters
  - **Resolution:** Indicates the maximum distance between data points from which the elevation was interpolated.

Map routing

Routing allows manually constructing a route by adding waypoints to the map. Waypoints can be added by a click or a doubleclick on the map. Each time a waypoint is added to the route the “OnRoutingWaypointAdded” and “OnAfterRoutingWaypointAdded” events are triggered. If a click occurs on a location for which no route can be calculated, the “OnWebGMapsError” event is triggered with the “ErrorType” parameter set to “etInvalidWaypoint”.

- **procedure Clear;**
  Completely remove an existing route.

- **Procedure RemoveLastWayPoint;**
  Removes the last waypoint that was added to the route.

- **Distance: integer;**
  Contains the total distance of the currently displayed route in meters or feet (depending on the "Units" setting).

- **Enabled: Boolean;**
  Enable routing mode. When set to true the first click or doubleclick (depending on the "RoutingType" property value) on the map will add a starting point for a route to the map. Further clicks or double clicks will add a new waypoint to the route.

- **EndAddress: string**;
  Contains the end address of the currently displayed route.

- **MarkerColor: TMarkerIconColor;**
  Indicates the color the markers that will be used to indicate the starting point and each waypoint of a route.

- **MarkerIcon**: **String;**
  Set the path to the image file to use as a marker icon if Markers is set to rmCustom.

- **Markers: TRoutingMarkers;**
  Set the type of marker that will be used to indicate the starting point and each waypoint of a route.

    - **rmNone:** No markers are displayed
    - **rmColor:** Use a marker with a specific color. Set the color with the MarkerColor property.
    - **rmCustom:** Use a custom image file as a marker. Set the path to the file with the MarkerIcon property
    - **rmDefault:** The default marker appearance is used

- **PolylineOptions: TMapPolylineOptions;**
  Configure the appearance of the route polyline.

    - **Color: TColor;**
      Set the color of the polyline
    - **Icons:** Configure the symbols displayed on the polyline
        - **SymbolType:** The type of icon that is displayed on the polyline.

- **Offset:** The distance from the start of the line at which an icon is to be rendered. Can be set as a percentage (OffsetType set to dtPixels) or in pixels (OffsetType set to dtPercentage).
- **OffsetType:** Defines the way the Offset value is used.
  - **ctPercentate:** The Offset is handled as a percentage value.
  - **ctPixel:** The Offset is handled as a pixel value.
- **RepeatValue:** The distance between consecutive icons on the line. Can be set as a percentage (RepeatType set to dtPixels) or in pixels (RepeatType set to dtPercentage).
- **RepeatType:**
  - **ctPercentate:** The RepeatValue is handled as a percentage value.
  - **ctPixel:** The RepeatValue is handled as a pixel value.
- **FixedRotation:** If set to true, each icon in the sequence has the same fixed rotation regardless of the angle of the edge on which it lies. If set to false, each icon in the sequence is rotated to align with its edge.

- **Opacity: Integer;**
  Set the opacity of the polyline (0-255)
- **Width: Integer;**
  Set the width of the polyline

- **RoutingType: TRoutingType;**
  Set to rtClick to add a waypoint with each click on the map or to rtDoubleClick to add a waypoint with each double click on the map.
- **StartAddress: string**;
  Contains the start address of the currently displayed route.
- **Units: TUnits;**
  Sets which unit system to use for the Distance value. If set to usMetric the Distance value is returned in metres, if set to usImperial in feet.

Map methods

- **function ScreenShot(ImgType: TImgType): TGraphic;**

The function takes a screenshot of the actual map canvas. This screenshot is taken in the chosen imagetype: itJPeg, itBitmap or itPng. The graphic can easily be saved to file (see the samples paragraph).

- **function Launch: Boolean;**

  This function launches the Google Maps control.

- **function CreateMapCluster(Cluster: TMapCluster): Boolean;**

  Adds Cluster from the Clusters collection to the map.

- **function UpdateMapCluster(Cluster: TMapCluster): Boolean;**

  Applies updated property settings to a cluster on the map.

- **function DeleteMapCluster(Id: Integer): Boolean;**

  Removes a cluster from the map. All markers assigned to the cluster will also be removed from the map.

- **function DeleteAllMapCluster: Boolean;**

  Removes all clusters from the map. All markers assigned to the clusters will also be removed from the map.

- **function DeleteAllMapMarker: Boolean;**

  This function removes all previously created markers.

- **function CreateMapMarker(Marker: TMarker): Boolean;**

  The function adds a new marker in the markers collection.

- **function DeleteMapMarker(Id: Integer):Boolean;**

  The function removes a marker from the markers collection.

- **function GetCurrentLocation: Boolean;**

  Sets the CurrentLocation.Latitude and CurrentLocation.Longitude to the coordinates of the current location.

- **function OpenMarkerInfoWindowHtml(Id: Integer; HtmlText: String):Boolean;**

  The function opens the marker info window for the marker with selected marker-id (Marker.Index). Extra information can be passed via the HtmlText string. A sample can be found in the samples paragraph.

- **function CloseMarkerInfoWindowHtml(Id: Integer):Boolean;**

  The function closes the marker with the given marker-id (Marker.Index).

- **function GetMapBounds: Boolean;**

  This function retrieves the bounds coordinates of the currently displayed map. The bounds are returned via the OnBoundsRetrieved event.

- **function MapPanTo(Latitude, Longitude: Double):Boolean;**

  This function performs a pan to a location set by latitude and longitude coordinates. This is useful to set a certain position in the center of the control canvas.

- **function MapZoomTo(Bounds: TBounds): Boolean;**

  This function performs a zoom to fit the map inside the given bounds coordinates.

- **function MapPanBy(X, Y: Integer):Boolean;**

  The function moves the map horizontally (x) and vertical (y) pixels.

- **function RenderDirections(Origin, Destination: string; TravelMode: TTravelMode = tmDriving; AvoidHighways: Boolean = false; AvoidTolls: Boolean = false; WayPoints: TStringList = nil; OptimizeWayPoints: Boolean = false; RouteColor: TColor): Boolean;**

  The function renders the directions on the map based on the provided parameters.

- **function RemoveDirections(): Boolean;**
  function RemoveDirections(Index: Integer): Boolean;

  The function removes the last directions that where placed on the map using the RenderDirections function.
  If the Index parameter is assigned the function removes only the directions indicated by the index value.

- **function RemoveAllDirections(): Boolean;**

  The function removes all directions that where placed on the map using the RenderDirections function.

- **function DegreesToLonLat(StrLon, StrLat: string; var Lon, Lat: double): Boolean;**

  This function converts degrees to longitude / latitude coordinates.

- **Function XYToLonLat(X, Y: integer; var Lon, Lat: double): Boolean;**

  This function converts XY coordinates to longitude / latitude coordinates. Returns the latitude and longitude coordinates of the X and Y pixel coordinates in the control window.

- **Function LonLatToXY(Lon, Lat: double; var X, Y: integer): Boolean;**

  This function converts longitude / latitude coordinates to XY coordinates. Returns the X and Y pixel coordinates in the control window of the latitude and longitude coordinates.

- **function AddMapKMLLayer(Url: string; ZoomToBounds: boolean): Boolean;**

  This function displays a KML file on the map as defined by the Url parameter.
  If the ZoomToBounds is true the map is zoomed to the bounding box of the contents of the layer.

- **function DeleteMapKMLLayer(Id: Integer):Boolean;**

  The function removes a KML layer from the map.

- **function DeleteAllMapKMLLayer: Boolean;**

  This function removes all KML layers from the map.

- **function LoadGPSRoute(AFilename: string; AColor: TColor; AWidth: integer): string;**
  This function loads a GPS route from a GPX file and displays it on the map as a Polyline.

- **function GetModifiedMapPolyline(Polyline: TPolyline):Boolean;**
  This function retrieves modified coordinates from a Polyline on the map and updates the Polyline.Path values.

- **function GetModifiedMapPolygon(Polygon: TMapPolygon):Boolean;**
  This function retrieves modified coordinates from a Polygon on the map and updates the Polygon values. (This includes the Path values for a Polygon of type ptPath, the Center and Radius values for Polygon of type ptCircle and the Bounds values for a Polygon of type ptRectangle)

- **function AddGeoImage(FileName: string; Title: string = ''; Width: integer = -1; Height: integer = -1; ZoomWidth: integer = -1; ZoomHeight: integer = -1): TMarker;**
  This function retrieves the geo coordinates that are found in the exif data of the image file specified in the FileName parameter and adds a Marker to the map at that location with the image file as its icon. Use the optional Width, Height, ZoomWidth and ZoomHeight parameters to specify a custom size for the marker icon. Use in combination with MapOptions.ZoomMarker. See Marker properties IconWidth, IconHeight, IconZoomWidth, IconZoomHeight for further information.

- **Procedure SaveMarkersToPoi(PoiFile: string);**
  This functions saves the coordinates of all markers to the POI file specified in PoiFile.

- **Procedure LoadMarkersFromPoi(PoiFile: string; MarkerColor: TMarkerIconColor);**
  This functions loads a set of coordinates from the POI file specified in PoiFile and automatically adds them to the map as markers.
  Optionally the color of the markers can be specified with MarkerColor.

- **Procedure SaveMapBounds;**
  Save the current map bounds

- **Procedure LoadMapBounds;**
  Load the previously saved map bounds

- **Procedure ClearPolygons;**
  Remove all polygons from the map and clear the Polygons collection

- **Procedure ClearPolylines;**
  Remove all polylines from the map and clear the Polylines collection

- **Function LoadGeoJSONPolyline(AFilename: string; AColor: TColor = clBlue; Opacity: integer = 255; AWidth: integer = 2; Zoom: boolean = true; HoverColor: TColor = clBlue): string;**
  This function loads coordinates from a GEOJSON file and displays it on the map as a Polyline or Polylines.
  Optionally set the Color, Opacity, Width, HoverColor of the Polyline(s).
  Optionally set Zoom to true to automatically zoom the map to the bounds of the Polyline(s).

- **Function LoadGeoJSONPolygon(AFilename: string; BorderColor: TColor = clBlue; Opacity: integer = 255; BackgroundColor: TColor = clBlue; BackgroundOpacity: integer = 100; AWidth: integer = 2; Zoom: boolean = true; HoverBorderColor: TColor = clBlue; HoverBackgroundColor: TColor = clBlue): string;**
  This function loads coordinates from a GEOJSON file and displays it on the map as a Polygon or Polygons.
  Optionally set the BorderColor, Opacity, BackgroundColor, BackgroundOpacity, Width, HoverBorderColor, HoverBackgroundColor of the Polygon(s).
  Optionally set Zoom to true to automatically zoom the map to the bounds of the Polygon(s).

Map events

- **OnBoundsRetrieved(Sender: TObject; Bounds: TBounds);**

  Event triggered after the GetBounds function has been called. This event returns the bounds coordinates of the currently displayed map.

- **OnClusterClick(Sender: TObject; IdCluser, MarkerCount: integer; Latitude, Longitude: Double);**

Event triggered when a cluster is clicked. Returns the number of markers contained in the cluster and the latitude and longitude coordinates of the cluster position.

- **OnClusterMouseEnter(Sender: TObject; IdCluster, MarkerCount: integer; Latitude, Longitude: Double);**

  Event triggered when the mouse cursors enters a cluster. Returns the number of markers contained in the cluster and the latitude and longitude coordinates of the cluster position.

- **OnClusterMouseExit(Sender: TObject; IdCluster, MarkerCount: integer; Latitude, Longitude: Double);**

  Event triggered when the mouse cursor exits a cluster. Returns the number of markers contained in the cluster and the latitude and longitude coordinates of the cluster position.

- **OnDownloadFinish(Sender: TObject):**

  Event triggered when the map download is finished.

- **OnDownloadProgress(Sender: TObject; Progress, ProgressMax: Integer):**

  Event triggered while the map is downloading. The event returns the current progress position and the maximal progress value.

- **OnDownloadStart(Sender: TObject):**

  Event triggered when the map download is started.

- **OnInitHTML(Sender: TObject; var HTML: string);**

  Even triggered after the HTML data to display the map has been generated. Allows modifying the HTML data through the HTML parameter.

- **OnMapTilesLoad(Sender: TObject);**
  Event triggered when all map tiles have finished loading after the map position and/or zoom level has changed.

- **OnMapClick(Sender: TObject; Latitude, Longitude: Double; X, Y: Integer; Button: TMouseButton):**

  Event triggered when the map is clicked. Returns the latitude and longitude coordinates of that position, the X and Y values indicate the pixel coordinates in the control window, button parameter returns what button was clicked on the mouse.

- **OnMapDblClick(Sender: TObject; Latitude, Longitude: Double; X, Y: Integer):**

  Event triggered when the map is double-clicked. Returns the latitude and longitude coordinates of that position, the X and Y values indicate the pixel position in the control window.

- **OnMapIdle(Sender: TObject):**

  Event triggered when the map is idle.

- **OnMapMouseEnter(Sender: TObject; Latitude, Longitude: Double; X, Y: Integer);**

  Event triggered when the mouse cursor enters the control. Returns the latitude and longitude coordinates of that position, the X and Y values indicate the pixel position in the control window.

- **OnMapMouseExit(Sender: TObject; Latitude, Longitude: Double; X, Y: Integer):**

  Event triggered when the mouse cursor exits the control. Returns the latitude and longitude coordinates of that position, the X and Y values indicate the pixel position in the control window.

- **OnMapMouseMove(Sender: TObject; Latitude, Longitude: Double; X, Y: Integer):**

  Event triggered when the mouse is moved within the control. Returns the latitude and longitude coordinates of that position, the X and Y values indicate the pixel position in the control window.

- **OnMapMove(Sender: TObject; Latitude, Longitude: Double; X, Y: Integer):**

  Event triggered when the entire map is moved (left mouse and drag) within the control. Returns the latitude and longitude coordinates of the mouse cursor position, the X and Y values indicate the pixel position of the mouse cursor in the control window.

- **OnMapMoveEnd(Sender: TObject; Latitude, Longitude: Double; X, Y: Integer):**

  Event triggered at the end of an entire map move (left mouse and drag) within the control. Returns the latitude and longitude coordinates of that position, the X and Y values indicate the pixel position in the control window.

- **OnMapMoveStart(Sender: TObject; Latitude, Longitude: Double; X, Y: Integer):**

  Event triggered at the start of an entire map move (left mouse and drag) within the control. Returns the latitude and longitude coordinates of that position, the X and Y values indicate the pixel position in the control window.

- **OnMapTypeChange(Sender: TObject; NewMapType: TMapType):**

  Event triggered when the map type is changed. This event returns the selected map type.

- **OnMapZoomChange(Sender: TObject; NewLevel: Integer):**

  Event triggered when the zoom level is changed via any type of the zoom control. The event returns the selected zoom level.

- **OnMarkerClick(Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double; Button: TMouseButton):**

Event triggered when a marker is clicked. Returns the marker title, the marker id, latitude and longitude coordinates defined for the marker, and what button has been clicked on the mouse.

- **OnMarkerDblClick(Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double):**

  Event triggered when a marker is double-clicked. The event returns the marker title, the marker id, latitude and longitude coordinates of the selected marker.

- **OnMarkerDrag(Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double):**

  Event triggered when a marker is dragged around the control. The event returns the marker title, the marker id, latitude and longitude coordinates of the selected marker.

- **OnMarkerDragEnd(Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double):**

  Event triggered at the end of when a marker is dragged in the control. The event returns the marker title, the marker id, latitude and longitude coordinates of the selected marker.

- **OnMarkerDragStart(Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double):**

  Event triggered at the start of when a marker is dragged in the control. The event returns the marker title, marker id, latitude and longitude coordinates of the selected marker.

- **OnMarkerInfoWindowCloseClick(Sender: TObject; IdMarker: Integer):**

  Event triggered when the info window is closed. The event returns the marker id.

- **OnMarkerMouseDown(Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double):**

  Event triggered when the mouse cursor is over a marker and a mouse button is pressed. The event returns the marker title, marker id, latitude and longitude coordinates of the selected marker.

- **OnMarkerMouseEnter(Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double):**

  Event triggered when the mouse cursor enters a marker. The event returns the marker title, marker id, latitude and longitude coordinates for that position.

- **OnMarkerMouseExit(Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double):**

  Event triggered when the mouse cursor leaves the marker. The event returns the latitude and longitude coordinates of that position the X and Y values indicate the pixel position in the control window.

- **OnMarkerMouseUp(Sender: TObject; MarkerTitle: string; IdMarker: Integer; Latitude, Longitude: Double):**

  Event triggered when the mouse cursor is over a marker and a mouse button is released. The event returns the marker title, marker id, latitude and longitude coordinates of the selected marker.

- **OnMarkerZoomIn(Sender: TObject; IdMarker: Integer; var Allow: boolean):**

  Event triggered when the marker icon state changes from msDefault to msZoomedIn. Set Allow to false to prevent the state change. Use in combination with MapOptions.ZoomMarker and Markers[].Width, Height, ZoomWidth, ZoomHeight.

- **OnMarkerZoomIn(Sender: TObject; IdMarker: Integer; var Allow: boolean):**

  Event triggered when the marker icon state changes from msZoomedIn to msDefault. Set Allow to false to prevent the state change. Use in combination with MapOptions.ZoomMarker and Markers[].Width, Height, ZoomWidth, ZoomHeight.

- **OnPolylineClick(Sender: TObject; IdPolyline: Integer; Button: TMouseButton):**

  Event triggered when a polyline is clicked. Returns the polyline id and what button has been clicked on the mouse.

- **OnPolylineDblClick(Sender: TObject; IdPolyline: Integer):**

  Event triggered when a polyline is double-clicked. The event returns the polyline id.

- **OnPolygonMouseDown(Sender: TObject; IdPolyline: Integer):**

  Event triggered when the mouse cursor is over a polyline and a mouse button is pressed. The event returns the polyline id.

- **OnPolylineMouseEnter(Sender: TObject; IdPolyline: Integer):**

  Event triggered when the mouse cursor enters a polyline. The event returns the polyline id.

- **OnPolylineMouseExit(Sender: TObject; IdPolyline: Integer):**

  Event triggered when the mouse cursor leaves a polyline. The event returns polyline id.

- **OnPolylineMouseUp(Sender: TObject; IdPolyline: Integer):**

  Event triggered when the mouse cursor is over a polyline and a mouse button is released. The event returns the polyline id.

- **OnPolylineChanged(Sender: TObject; IdPolyline: Integer):**

  Event triggered when a polyline has been modified on the map. The event returns the polyline id. The modified coordinates can be retrieved with the GetModifiedPoyline method.

- **OnPolygonClick(Sender: TObject; IdPolygon: Integer; Button: TMouseButton):**

  Event triggered when a polygon is clicked. Returns the polygon id and what button has been clicked on the mouse.

- **OnPolygonDblClick(Sender: TObject; IdPolygon: Integer):**

  Event triggered when a polygon is double-clicked. The event returns the polygon id.

- **OnPolygonMouseDown(Sender: TObject; IdPolygon: Integer):**

  Event triggered when the mouse cursor is over a polygon and a mouse button is pressed. The event returns the polygon id.

- **OnPolygonMouseEnter(Sender: TObject; IdPolygon: Integer):**

  Event triggered when the mouse cursor enters a polygon. The event returns the polygon id.

- **OnPolygonMouseExit(Sender: TObject; IdPolygon: Integer):**

  Event triggered when the mouse cursor leaves a polygon. The event returns polygon id.

- **OnPolygonMouseUp(Sender: TObject; IdPolygon: Integer):**

  Event triggered when the mouse cursor is over a polygon and a mouse button is released. The event returns the polygon id.

- **OnPolygonChanged(Sender: TObject; IdPolygon: Integer):**

  Event triggered when a polygon has been modified on the map. The event returns the polygon id. The modified coordinates can be retrieved with the GetModifiedPoygon method.

- **OnWebGMapsError(Sender: TObject; ErrorType: TErrorType):**

  Event triggered when an error is received. This event returns the error type.

- **OnKMLLayerClick(Sender: TObject; ObjectName: string; IdLayer: Integer; Latitude, Longitude: Double):**

  Event triggered when an object inside a KML layer is clicked. Returns the object name, the layer id and latitude and longitude coordinates defined for the object.

- **OnStreetViewChange(Sender: TObject; Heading, Pitch, Zoom: integer):**

  Event triggered when the Point Of View is changed while StreetView mode is active. Returns the Heading, Pitch and Zoom values.

- **OnStreetViewMove(Sender: TObject; Latitude, Longitude: Double; X, Y: integer):**

  Event triggered when the geographic position is changed while StreetView mode is active. Returns the latitude and longitude coordinates of that position, the X and Y values indicate the pixel position in the control window.

- **OnAfterRoutingWaypointAdded(Sender: TObject; Latitude, Longitude: Double; Route: TPath);**

  Event triggered after a waypoint was added in routing mode (with Routing.Enabled set to True). Returns the latitude and longitude coordinates of that position and the path of the route that was added. The full path for all waypoints can be retrieved from the Routing.Path collection.

- **OnRoutingWaypointAdded(Sender: TObject; Latitude, Longitude: Double; Route: TPath; var Allow: Boolean);**

  Event triggered when a starting location or waypoint is added in routing mode (with Routing.Enabled set to True). Returns the latitude and longitude coordinates of that position and the path of the route that was added. Set Allow to false to remove this waypoint form the route. To retrieve the full path for all waypoints, use the “OnAfterWaypointAdded” event.
  If an invalid location for a waypoint was clicked the “OnWebGMapsError” event is triggered instead, with the ErrorType set to etInvalidWaypoint.

TWebGMaps Keyboard

When TWebGMaps.MapOptions.EnableKeyboard is set to true, keyboard support is enabled.

Below is a list of keys that will allow you to navigate through the map without using the mouse.

Panning the map

Arrow key up, Arrow key down, Arrow key left, Arrow key right, move the map in the corresponding directions a pixel at a time.

Page up key, Page down key, Home key and End key move the map up, down, left and right within the control a page at a time.

Zooming the map

The Plus (+) key functions as a click on the plus button of the zoom control, and performs a zoom in the map with one step.

The Minus (-) key performs the same as a click of the minus button in the zoom control, and performs a zoom out in the map with one step.

TWebGMaps Sample code

Sample 1

This sample shows how to load the info Window with text in when the marker is clicked.

```
procedure TFrmMain.WebGMaps1MarkerClick(MarkerTitle: string; IdMarker:
Integer; Latitude, Longitude: Double; Button:TMouseButton);

begin
  if Button = mbLeft then
    begin
      WebGMaps1.OpenMarkerInfoWindowHtml(IdMarker,'<b>'+ MarkerTitle + '<br
/>' + 'Lat : ' + floattostr(latitude)+ '<br />' + 'Lon : ' +
floattostr(longitude) + '</b><br />')
    End;
end;
```

Sample 2

This sample demonstrates how to save a screenshot to a Windows bitmap file.

```
procedure TFrmMain.Button2Click(Sender: TObject);
var
  Graphic: TGraphic;
begin
  Graphic := WebGMaps1.ScreenShot(itBitmap);
  Graphic.SaveToFile(GlobalPath + 'files\imgbmp.bmp');
  Graphic.Free;
End;
```

Sample 3

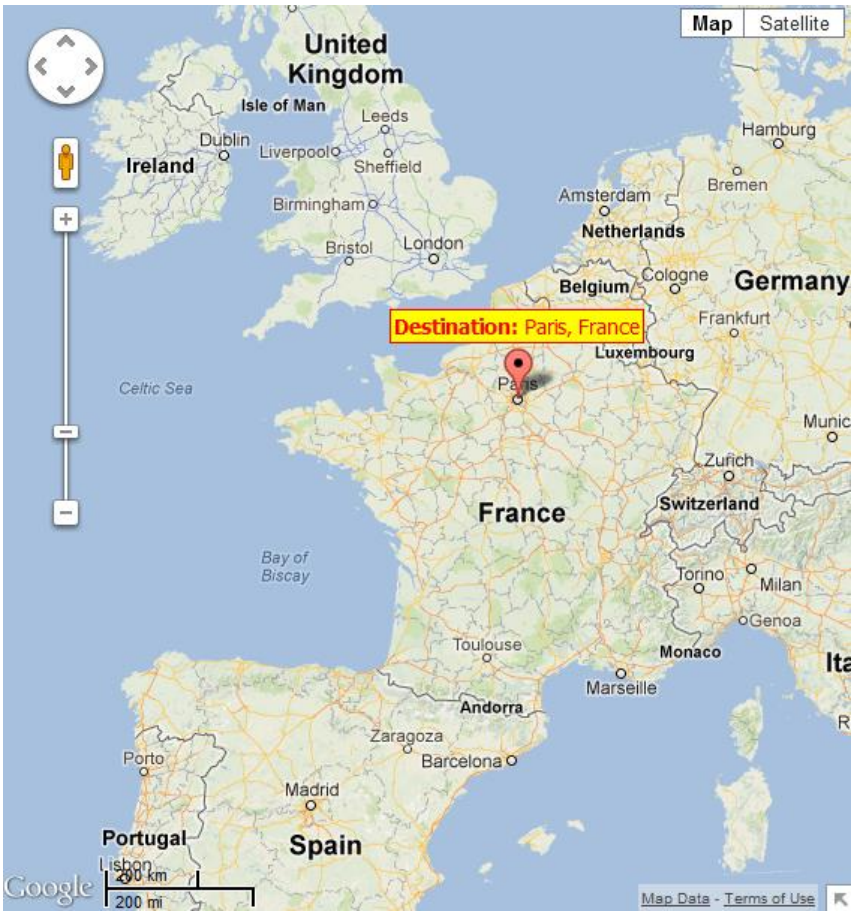
This example shows how to format an icon URL string for proper image loading.

```
Var
  MarkerIcon: string;

begin
```

```
  MarkerIcon := StringReplace('File://' + GlobalPath +
'Files\Thumbnails\EiffelTower.png', '\', '/',[rfReplaceAll, rfIgnoreCase]);

end;
```

Sample 4

This example shows how to format an icon URL string for proper image loading.




```
var
  Marker: TMarker;

begin

WebGMapsGeocoding1.Address := 'Paris, France';

if WebGMapsGeocoding1.LaunchGeocoding = erOk then
begin
  Marker := WebGMaps1.Markers.Add;
  Marker.Latitude := WebGMapsGeocoding1.ResultLatitude;
```

```
  Marker.Longitude := WebGMapsGeocoding1.ResultLongitude;
  Marker.Title := 'Destination: ' + WebGMapsGeocoding1.Address;
  Marker.MapLabel.Text := '<b>Destination:</b> ' +
WebGMapsGeocoding1.Address;
  Marker.MapLabel.Color := clYellow;
  Marker.MapLabel.BorderColor := clRed;
  Marker.MapLabel.Font.Color := clRed;
  Marker.MapLabel.Font.Size := 14;
  Marker.MapLabel.Font.Name := 'Tahoma';
  WebGMaps1.CreateMapMarker(Marker);
end;
```

TWebGMapsGeocoding component

The TWebGMapsGeocoding component is a helper component to enable using the Google geocoding service to convert an address to a longitude, latitude coordinate. The TWebGMapsGeocoding component is simple to use. Just set the address for which a lookup to longitude & latitude is needed and call the function TWebGMapsGeocoding.LaunchGeocoding. When the result of this call is erOK, the geocoding was successful and the longitude & latitude for the address can be read from:

TWebGMapsGeocoding.ResultLatitude
TWebGMapsGeocoding.ResultLongitude

Note that the recommended formatting for the address is:

STREET (NUMBER), (ZIPCODE) CITY, COUNTRY

Example:

```
WebGMapsGeocoding1.Address := 'Broadway 615, LOS ANGELES, USA';
if WebGMapsGeocoding1.LaunchGeocoding = erOk then
begin
     ShowMessage('Result:'+
FloatToStr(WebGMapsGeocoding1.ResultLongitude)+':'+
FloatToStr(WebGMapsGeocoding1.ResultLatitude));
end;
```

Possible error codes are:

erZeroResults : address not found
erOverQueryLimit : number of allowed geocoding queries per day exceeded
erRequestDenied: Google blocked request from your IP address
erInvalidRequest: address not correctly formatted
erOtherProblem: unknown problem

TWebGMapsReverseGeocoding component

The TWebGMapsReverseGeocoding component is a helper component to enable using the Google geocoding service to convert a longitude, latitude coordinate into an address. Using TWebGMapsReverseGeocoding is straightforward. Set the longitude and latitude values for which to lookup the address via the properties:

TWebGMapsReverseGeocoding.Latitude
TWebGMapsReverseGeocoding.Longitude

and call:

TWebGMapsReverseGeocoding.LaunchReverseGeocoding: TGeocodingResult;

For a successful reverse lookup, the result address is returned via:

TWebGMapsReverseGeocoding.ResultAddress: TWebGMapsAddress;

In this class property, the address is returned in a formatted way (TWebGMapsReverseGeocoding.ResultAddress.FormattedAddress) and in the specific parts of the address:

```
property Street: string;
property StreetNumber: string;
property City: string;
property State: string;
property Region: string;
property Country: string;
property CountryCode: string;
property PostalCode: string;
```

Example:

In this example, we get the address of the Big Ben in London via Geocoding and use the resulting geocoordinates to obtain the address via reverse geocoding:

```
// first retrieve geocoordinates from Big Ben
  WebGMapsGeocoding1.Address := 'Big Ben, London';
  WebGMapsGeocoding1.LaunchGeocoding;
```

```
// show Big Ben with marker on the WebGMaps control
  WebGMaps1.MapOptions.DefaultLatitude :=
WebGMapsGeocoding1.ResultLatitude;
  WebGMaps1.MapOptions.DefaultLongitude :=
WebGMapsGeocoding1.ResultLongitude;
  WebGMaps1.Markers.Add(WebGMapsGeocoding1.ResultLatitude,
WebGMapsGeocoding1.ResultLongitude, 'Big Ben');
  WebGMaps1.Launch;

// retrieve the address via reverse geocoding
  WebGMapsReverseGeocoding1.Latitude :=
WebGMapsGeocoding1.ResultLatitude;
  WebGMapsReverseGeocoding1.Longitude :=
WebGMapsGeocoding1.ResultLongitude;
  WebGMapsReverseGeocoding1.LaunchReverseGeocoding;

Memo1.Lines.Add(WebGMapsReverseGeocoding1.ResultAddress.FormattedAdd
ress);
```

Note that WebGMapsReverseGeocoding.LaunchReverseGeocoding function can return following results:

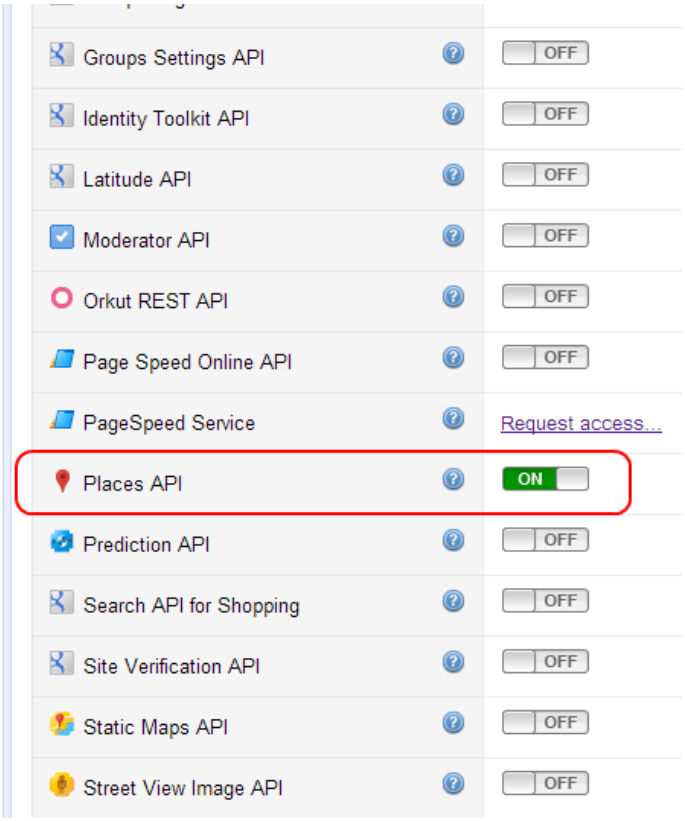
Possible error codes are:

erOK: reverse geocoding successful
erZeroResults : address not found
erOverQueryLimit : number of allowed geocoding queries per day exceeded
erRequestDenied: Google blocked request from your IP address
erInvalidRequest: address not correctly formatted
erOtherProblem: unknown problem

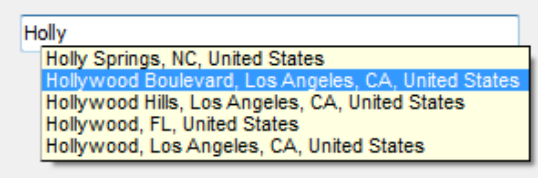
TWebGMapsLookupEdit component

The TWebGMapsLookupEdit component is a helper component that allows to enter addresses with auto completion based on information from Google Maps.
To use the Google Maps auto completion service, it is required to obtain a key from https://code.google.com/apis/console/

Depending on the type of use of your application, this key is free or needs to be paid. To get access with your app key to the Google places API for auto completion, make sure to turn this on under Services:



Use of TWebGMapsLookupEdit is straightforward. Make sure to set your key via the property : WebGMapsLookupEdit.APIKey := 'xxxxxxxx' and enable the lookup via: WebGMapsLookupEdit.Lookup.Enabled := true. As the user starts typing, the list of matching addresses retrieved via the Google Maps Places autocompletionn service is displayed:



Following properties allow to further fine-tune the lookup:

Lookup.AcceptOnTab: boolean : when true, pressing tab on the selected item will select it from the lookup list.

Lookup.Color: TColor: sets the background color of the lookup list

Lookup.DisplayCount : integer : sets the maximum number of lookup items to display in the dropdown. If more items are availble, a scrollbar is used.

Lookup.Enabled: boolean : enables or disables the lookup

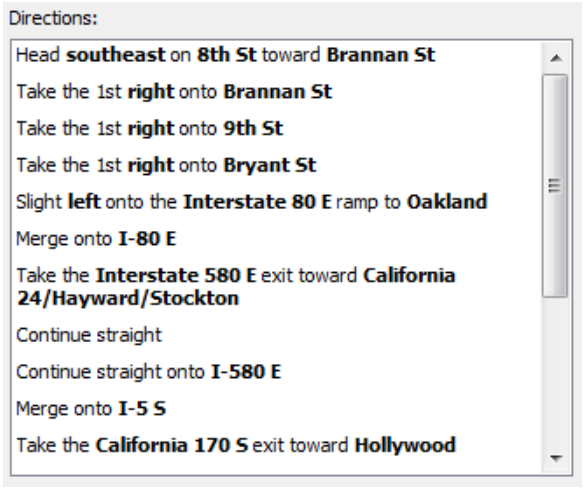
Lookup.Language: specifies what language to use for the lookup. This is the ISO 639 language code. See: http://www.loc.gov/standards/iso639-2/php/code_list.php for a full list. Some frequently used language names are:

English: en
Dutch: nl
French: fr
German: de
Spanish: es
Portuguese: pt
Russian: ru
Italian: it

Lookup.NumChars: integer : this sets the minimum number of characters the user must type before the lookup starts.

TWebGMapsDirectionList component

The TWebGMapsDirectionList component is a helper component that allows to display a list of steps of a route with HTML formatted text.



The TWebGMapsDirectionList behaves likes a regular VCL TListBox, except that it can display the HTML formatted text returned by the Google Maps Direction API. The TWebGMaps component can automatically add all steps of a selected route to a TWebGMapsDirectionList. To do this, call:

WebGMaps.FillDirectionList(WebGMapsDirectionList.Items);

TWebGMapsTimeZone component

The TWebGMapsTimeZone component is a helper component to enable using the Google TimeZone service to lookup time zone information based on a longitude, latitude coordinate.
Using TWebGMapsTimeZone is straightforward. Set the longitude and latitude values for which to lookup the timezone information via the properties:

TWebGMapsTimeZone.Latitude
TWebGMapsTimeZone.Longitude

Language and TimeStamp are two optional settings that can be also included in the request:

WebGMapsTimeZone1.Language;
WebGMapsTimeZone1.TimeStamp;

and call:

TWebGMapsTimeZone.GetTimeZone: TGeocodingResult;

For a successful time zone lookup, the result is returned via:

TWebGMapsTimeZone.ResultTimeZone: TWebGTimeZone;

In this class property, the time zone information is returned as follows:

property TimeOffset: double;
The offset from UTC (in seconds) for the given location.
property DaylightSavingTimeOffset: double;
The offset for daylight-savings time in seconds.
property TimeZoneId: string;
The “tz” ID of the time zone.
property TimeZoneName: string;
The long form name of the time zone.

Example:

In this example, we get the geocoordinate from an OnMapClick event and then use the GetTimeZone call to retrieve the time zone information for the clicked location:

```
procedure TForm1.WebGMaps1MapClick(Sender: TObject; Latitude,
Longitude: Double;
  X, Y: Integer; Button: TMouseButton);
begin
  WebGMapsTimeZone1.Latitude := Latitude;
  WebGMapsTimeZone1.Longitude := Longitude;

  Memo1.Clear;
  if WebGMapsTimeZone1.GetTimeZone = erOk then
  begin
Memo1.Lines.Add(FloatToStr(WebGMapsTimeZone1.ResultTimeZone.TimeOffs
et));
    Memo1.Lines.Add(WebGMapsTimeZone1.ResultTimeZone.TimeZoneId);
    Memo1.Lines.Add(WebGMapsTimeZone1.ResultTimeZone.TimeZoneName);
```

```
  end;
end;
```

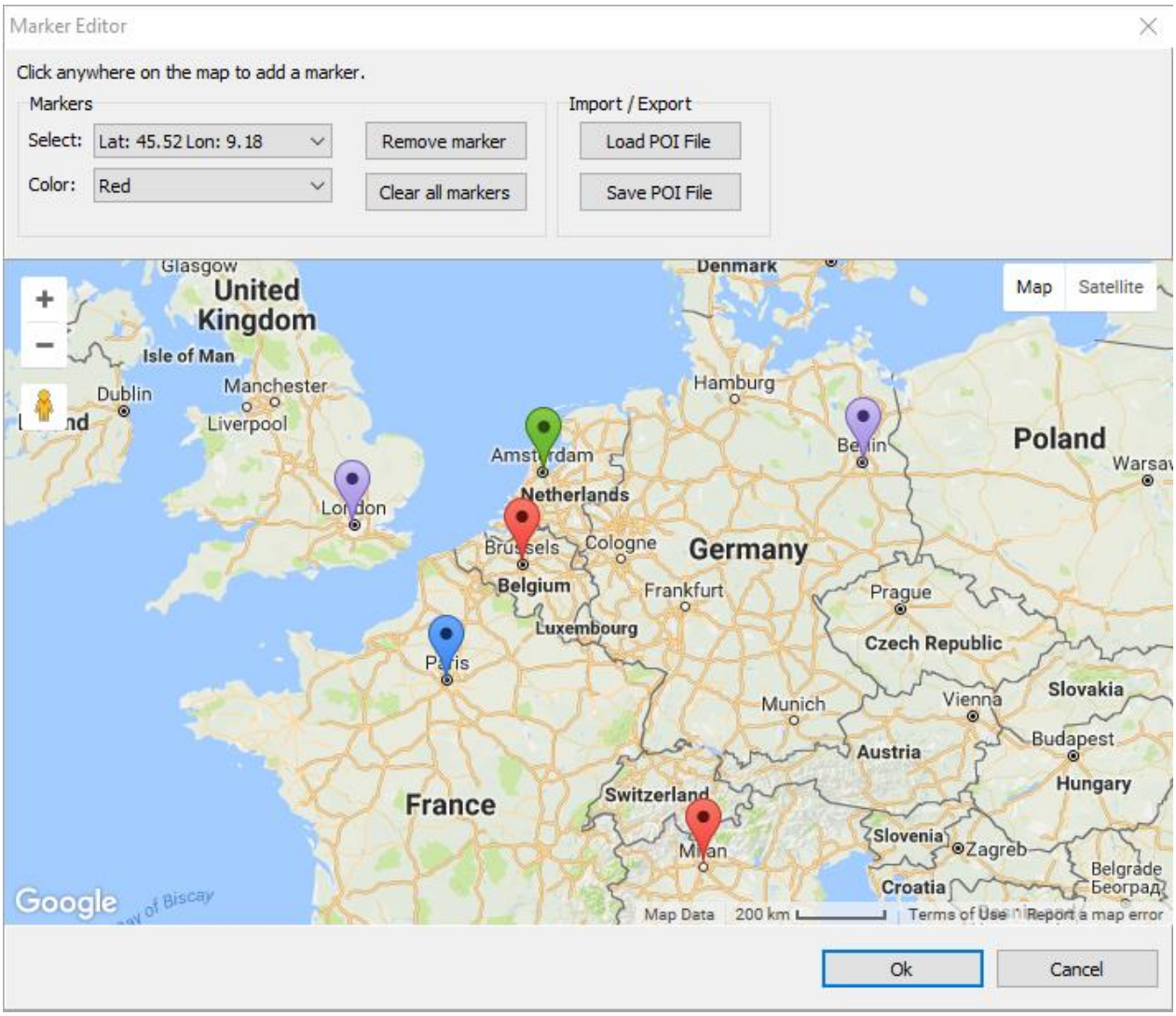
Note that WebGMapsTimeZone.GetTimeZone function can return following results:

Possible error codes are:

erOK: time zone lookup successful
erZeroResults : time zone not found
erOverQueryLimit : number of allowed time zone queries per day exceeded
erRequestDenied: Google blocked request from your IP address
erInvalidRequest: request not correctly formatted
erOtherProblem: unknown problem

TWebGMapsDialog component

The TWebGMapsDialog component is a helper component to enable marker editing at run-time. The TWebGMapsDialog offers functionality to add and remove markers, set or change the color of a marker and import/export POI files.



Properties:

ShowImportExport: Boolean;
Indicates if the import/export functionality is displayed on the dialog.
WebGMaps: TWebGMaps;
Can be used to assign an external TWebGMaps component. Existing markers and map position will be assigned to the dialog. Changes made to markers and map position will be assigned to the control assigned to WebGMaps when the OK button is clicked. If the dialog is canceled all changes are discarded.

Methods:

Execute: Boolean;
Shows the dialog. Returns true if the dialog's OK button is clicked or false if the dialog is canceled.

Events:

OnCancelClick(Sender: TObject):
Event triggered when the dialog is canceled.
OnClose(Sender: TObject):
Event triggered when the dialog is closed.

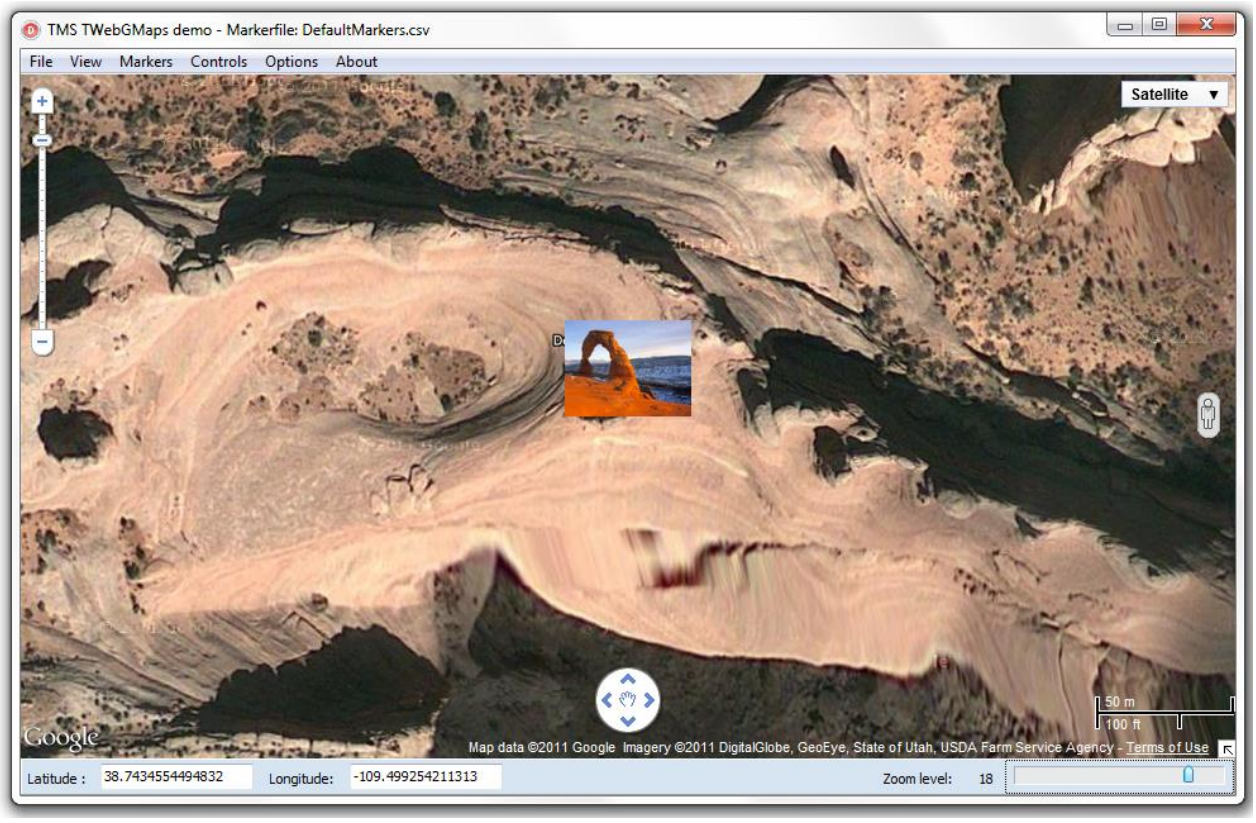
OnOKClick(Sender: TObject):
Event triggered when the OK button on the dialog is clicked.

OnShow(Sender: TObject):
Event triggered when the dialog is displayed.

TWebGMaps demo

The TMS VCL WebGMaps Demo program shows the various configuration possibilities of the TWebGMaps component. It allows to interactively set various properties and the changes will be immediately reflected in the map.

Main screen:



Note that the latitude and longitude values (i.e. center position of the map) can be changed at the bottom of the screen and the position is changed on exiting those controls. The zoom level can be modified with the slider control on the right bottom of the screen.

File menu

From here the displayed map can be saved as BMP,PNG or JPEG image. Choose the image type in the Save dialog.

View menu

From the view menu, the different map types can be selected: satellite, map, hybrid, terrain. In addition, on the displayed map type, the bicycle roads, panoramio pictures, traffic (where available), streetview (where available) can be displayed.

Markers menu

From here, panning can be done to the longitude, latitude coordinates as entered at the bottom of the screen, a marker can be added, all markers can be saved or loaded from a CSV file, markers can be deleted or a marker can be set at a specific address.

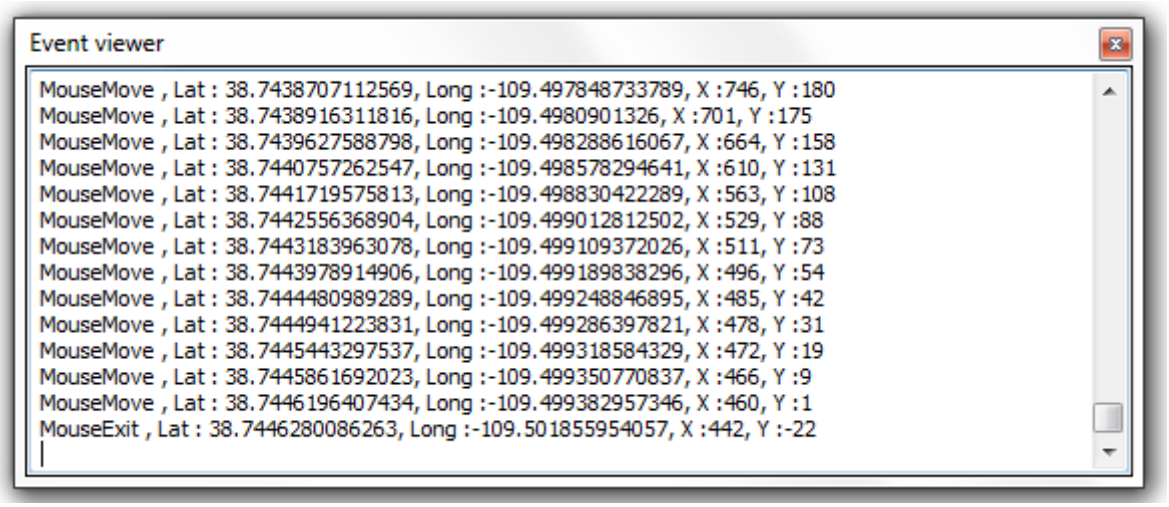
Controls menu

From here the location & visibility of the various controls on the Google map can be set.

Options menu

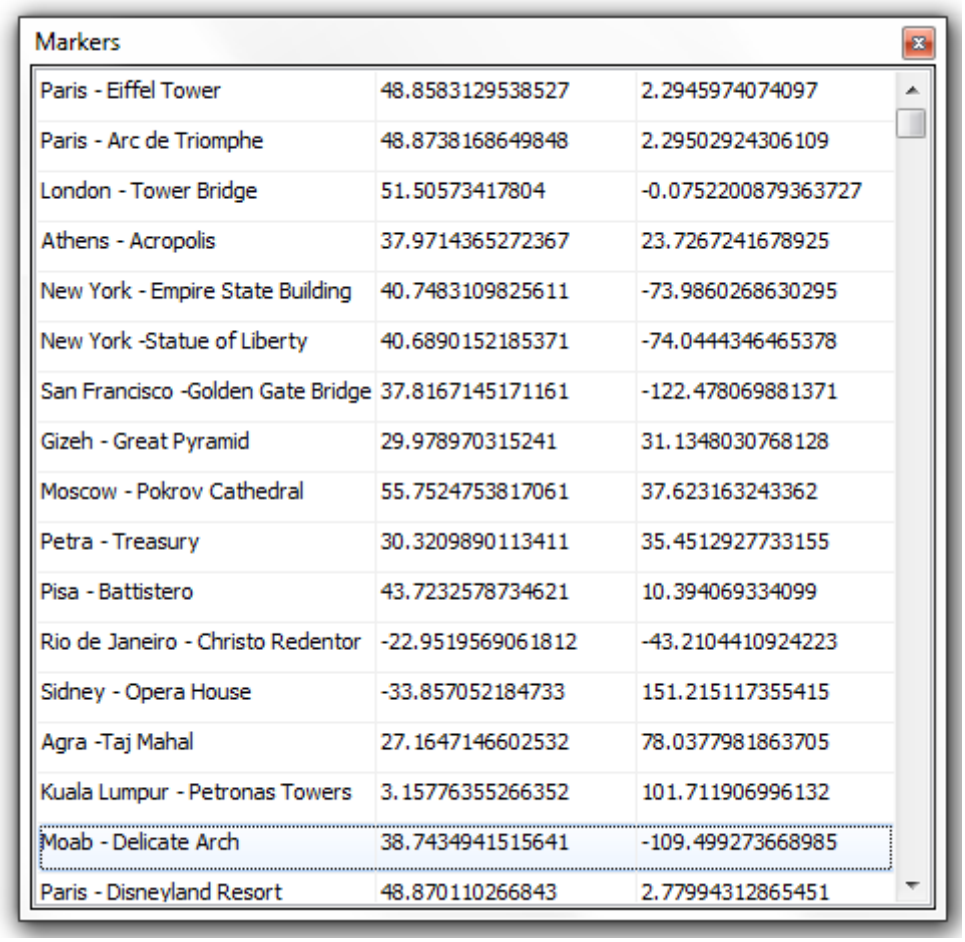
Extra views can be shown from the Options menu.

- The event viewer shows all occurring events:



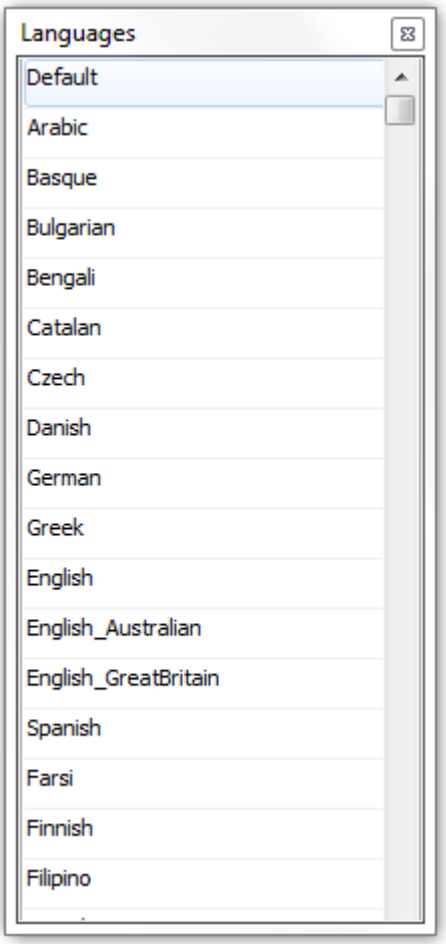
From left to right, the event viewer shows the type of the event, latitude and longitude, and position within the map control.

- The markers view shows all created markers, with name, latitude and longitude information:



When a marker is clicked, the map pans to the position of that clicked marker.

- The language view shows the possible map text language, and allows selecting the required language:



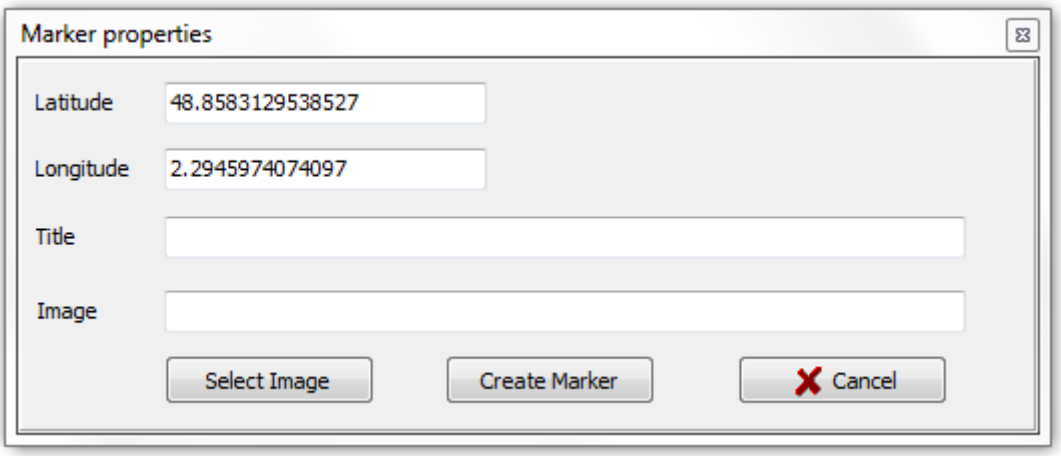
The language of the text fields on the map are changed to the selected language.

- Custom Pan Control example:

The arrows on left side move the map with a predefined step value, the right side arrows move the map in steps of one page.

The left center button pans the map to a position previously clicked in the map.

The right center button creates a marker loads the add marker at latitude/longitude screen.



Latitude, longitude, marker title text can be modified. An image URL can be defined, to load an image for that marker instead of the standard Google Maps marker icon.

A previously selected marker can be deleted via menu option Markers > Delete selected marker.