Google Maps API Experiments

Monash University
Falculty of IT
Vacation Scholarship Project

February 2009

Supervisor: Dr. David Taniar
Tran Quoc Thai

CONTENTS

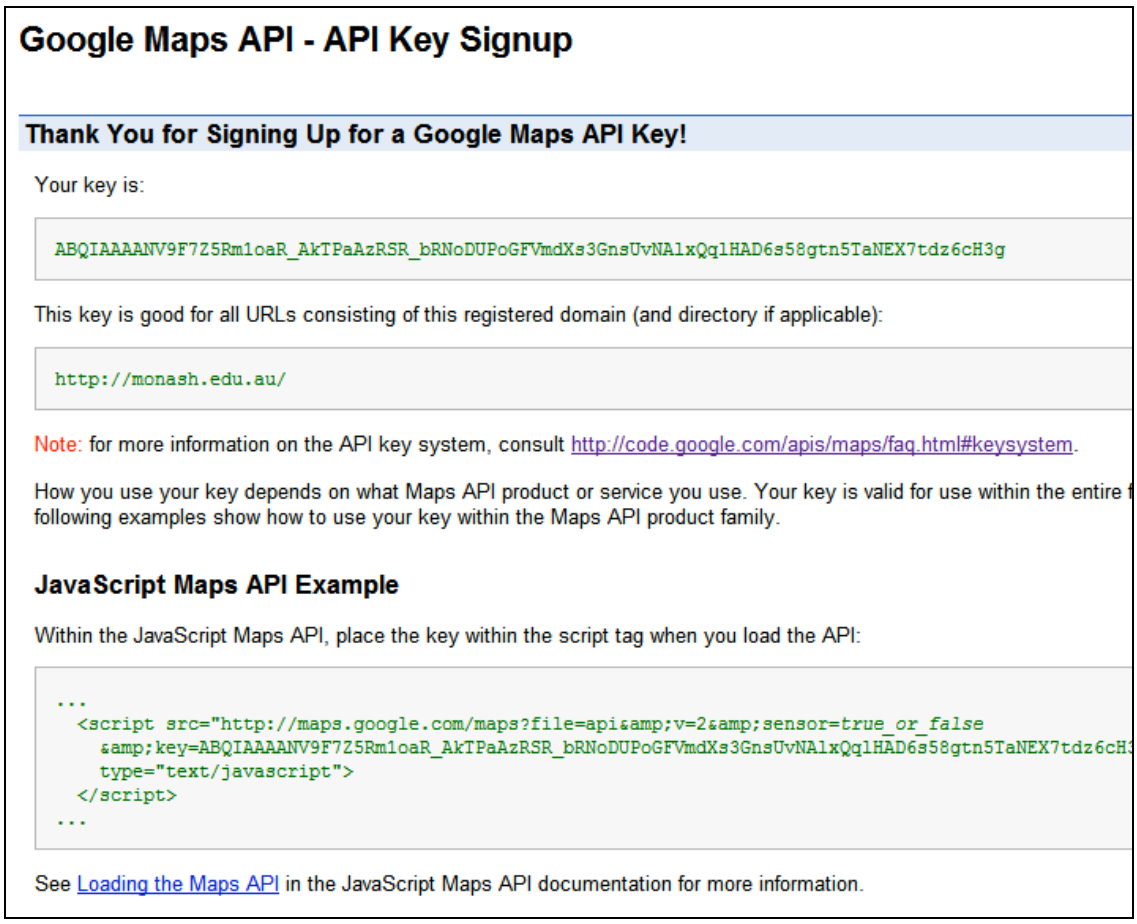
1. INTRODUCTION

Web developers use Google Maps API to embed Google Maps in their web pages. It is a code library from which a number of maps applications can be built using JavaScript.

This paper provides some examples as a result of the Google Maps API experiments some of which are borrowed from the website [1].

2. GOOGLE MAPS API KEY REGISTRATION

- Go to the link: http://code.google.com/apis/maps/signup.html
- Sign in a Google Account and Agree to Google Maps Terms of Service. The number of keys that can be obtained under a single account is unlimited.
- Simply register a domain name. The key which is registered for a domain will also be valid for its subdomains and all URLs on hosts in those domains, and all ports on those hosts.
    - For example, if we sign up for a key using ***http://www.mygooglemapsite.com/***, our key is valid for:
        - http://www.mygooglemapsite.com/
        - http://www.mygooglemapsite.com/mysite
    - However, this key will not be valid for:
        - http://mygooglemapsite.com/
        - http://host1.mygooglemapsite.com/
        - http://host2.mygooglemapsite.com/mysite
    - A key which is registered for ***http://mygooglemapsite.com/*** will be valid for all URLs mentioned above.
    - A key for http://www.mygooglemapsite.com/ will not be valid if the site is accessed by IP address (eg. http://10.1.2.3/) or a parked domain.
    - The key check is skipped when it is being developed on a local drive (file://)
    - The function alert(window.location.host) can be used to obtain the domain for which the key should be registered.



Google Maps API - API Key Signup

Thank You for Signing Up for a Google Maps API Key!

Your key is:

```
ABQIAAAANV9F7Z5Rm1oaR_AkTPaAzRSR_bRNoDUPoGFVmdXs3GnsUvNAlxQqlHAD6s58gtn5TaNEX7tdz6cH3g
```

This key is good for all URLs consisting of this registered domain (and directory if applicable):

```
http://monash.edu.au/
```

Note: for more information on the API key system, consult http://code.google.com/apis/maps/faq.html#keysystem.

How you use your key depends on what Maps API product or service you use. Your key is valid for use within the entire f
following examples show how to use your key within the Maps API product family.

JavaScript Maps API Example

Within the JavaScript Maps API, place the key within the script tag when you load the API:

```
...
  <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=true_or_false
    &amp;key=ABQIAAAANV9F7Z5Rm1oaR_AkTPaAzRSR_bRNoDUPoGFVmdXs3GnsUvNAlxQqlHAD6s58gtn5TaNEX7tdz6cH3
    type="text/javascript">
  </script>
...
```

See Loading the Maps API in the JavaScript Maps API documentation for more information.

- Copy and paste the code in the examples into a text editor (eg. Notepad).
- Find this line of code and replace ***"abcdefg"*** part by the registered key:

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

- Rename and save it as an html file.
- Upload to a server which domain name has been registered for the key.
- For more information, please refer to [3].

3. EXPERIMENTS

3.1. EXAMPLE 1: FINDING DIRECTIONS

- URL: http://www.geocities.com/quocthai0512/example1.html
- Description:
  - This example allows users to enter two addresses (e.g. source and destination) and select the language to get directions.
  - The formatted directions will be displayed on the left column whereas the right column will display the corresponding map and polylines.
  - The markers A and B in the screenshot denote the starting and ending points.
  - Users may click on one of the numbers on the left column to have a closer view of different points. The information window can also be turned off by users.
  - Next, a 3-D map control replaces the normal map control used in later examples. This map control is added by using two external javascript files called 'extlargemapcontrol_packed.js' and 'extlargemapcontrol.js' whose codes will be given later in this section.
  - Map type control is used to enable users to select different map types (map, satellite or hybrid).
  - A search textbox is added as an overlay of this map. As users enter an address into the textbox, the address will be set center to this map.
- Screenshots:

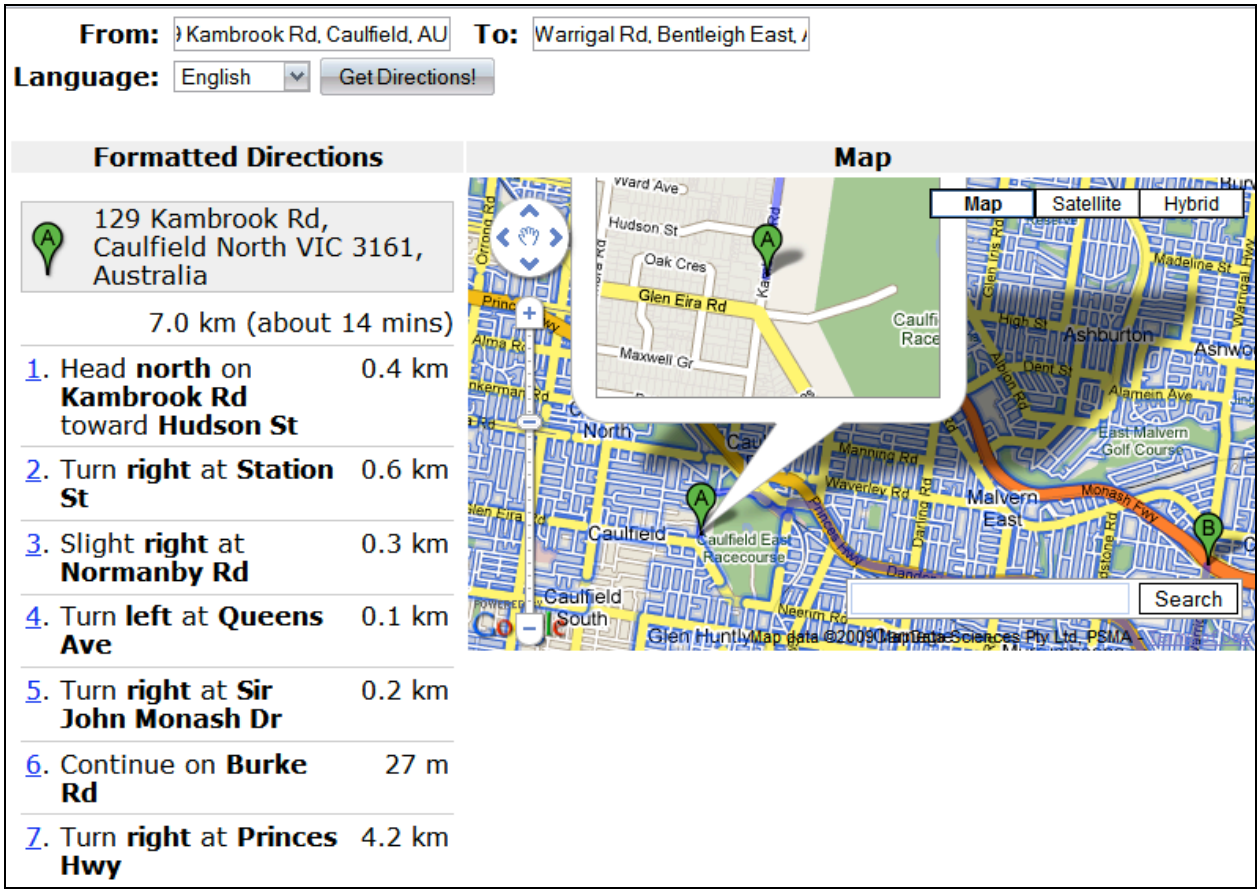


Figure 1 – Finding Directions

EXAMPLE1.HTML

```
<html>

<head>

<style type="text/css">

	@import url("http://www.google.com/uds/css/gsearch.css");

	@import url("http://www.google.com/uds/solutions/localsearch/gmlocalsearch.css"

	body {
		font-family: Verdana, Arial, sans serif;
		font-size: 11px;
		margin: 2px;
	}

	table.directions th {
		background-color:#EEEEEE;
	}

	img {
		color: #000000;
	}

</style>

<script
src="http://maps.google.com/maps?file=api&amp;v=2.x&amp;key=ABQIAAAAtWCOvsQE3_q3_qb8k
Y-VG3o8b6wZyHyJA" type="text/javascript"></script>

<script src="http://www.google.com/uds/api?file=uds.js&amp;v=1.0" type="text/javascri

<script src="http://www.google.com/uds/solutions/localsearch/gmlocalsearch.js" type="

<script type="text/javascript">
	document.write('<script type="text/javascript" src="extlargemapcontrol'+(docume
	1?'_packed':'')+'.js"><'+'/script>');
</script>

<script type="text/javascript">

	var map;
	var gdir;
	var geocoder = null;
	var addressMarker;


	function initialize() {

		if (GBrowserIsCompatible()) {
```

```
            map = new GMap2(document.getElementById("map_canvas"));
            map.setCenter(new GLatLng(-37.91170058826019,145.1345443725586), 14

            var extLargeMapControl = new ExtLargeMapControl();
            map.addControl(extLargeMapControl);
            map.addControl(new GMapTypeControl());

            map.addControl(new google.maps.LocalSearch(), new GControlPosition(

            var myPano = new GStreetviewPanorama(document.getElementById("pano"
            GEvent.addListener(myPano, "error", handleNoFlash);

            svOverlay = new GStreetviewOverlay();
            map.addOverlay(svOverlay);
            GEvent.addListener(map,"click", function(overlay,latlng) {
            myPano.setLocationAndPOV(latlng); });

            gdir = new GDirections(map, document.getElementById("directions"));
            GEvent.addListener(gdir, "load", onGDirectionsLoad);
            GEvent.addListener(gdir, "error", handleErrors);
      }
}

function handleNoFlash(errorCode) {
      if (errorCode == FLASH_UNAVAILABLE) {
            alert("Error: Flash doesn't appear to be supported by your browser"
            return;
      }
}

GSearch.setOnLoadCallback(initialize);

function setDirections(fromAddress, toAddress, locale) {
      gdir.load("from: " + fromAddress + " to: " + toAddress,{ "locale": locale
}

function handleErrors(){

      if (gdir.getStatus().code == G_GEO_UNKNOWN_ADDRESS)

            alert("No corresponding geographic location could be found for one
            to the fact that the address is relatively new, or it may be incorr

      else if (gdir.getStatus().code == G_GEO_SERVER_ERROR)

            alert("A geocoding or directions request could not be successfully
            failure is not known.\n Error code: " + gdir.getStatus().code);

      else if (gdir.getStatus().code == G_GEO_MISSING_QUERY)

            alert("The HTTP q parameter was either missing or had no value. For
            address was specified as input. For directions requests, this means
```

```
                  Error code: " + gdir.getStatus().code);

            else if (gdir.getStatus().code == G_UNAVAILABLE_ADDRESS)
            <--- Doc bug... this is either not defined, or Doc is wrong

                  //     alert("The geocode for the given address or the route for th
                  due to legal or contractual reasons.\n Error code: " + gdir.getStat

            else if (gdir.getStatus().code == G_GEO_BAD_KEY)

                  alert("The given key is either invalid or does not match the domain
                  gdir.getStatus().code);

            else if (gdir.getStatus().code == G_GEO_BAD_REQUEST)

                  alert("A directions request could not be successfully parsed.\n Err

            else

                  alert("An unknown error occurred.");
      }

      function onGDirectionsLoad(){
      // Use this function to access information about the latest load()
      // results.

      // e.g.
      // document.getElementById("getStatus").innerHTML = gdir.getStatus().code;
      }
</script>

</head>

<body onload="initialize()" onunload="GUnload()">

<form action="#" onsubmit="setDirections(this.from.value, this.to.value, this.locale.

<table>
      <tr>
            <th align="right">From: </th>
            <td><input type="text" size="25" id="fromAddress" name="from" value=""/><
            <th align="right">  To: </th>
            <td align="right"><input type="text" size="25" id="toAddress" name="to" v
      </tr>
      <tr>
            <th>Language: </th>
            <td colspan="3">
                  <select id="locale" name="locale">
                        <option value="en" selected>English</option>
                        <option value="fr">French</option>
                        <option value="de">German</option>
                        <option value="ja">Japanese</option>
                        <option value="es">Spanish</option>
```

```
                  </select>
                  <input name="submit" type="submit" value="Get Directions!" />
            </td>
      </tr>
</table>

</form>

<br />

<table class="directions">
      <tr><th>Formatted Directions</th><th>Map</th></tr>
      <tr>
            <td valign="top" rowspan="2"><div id="directions" style="width: 275px"></
            <td valign="top"><div id="map_canvas" style="width: 500px; height: 300px"
      </tr>
      <tr>
            <td valign="top"><div id="pano" style="width: 500px; height: 200px"></div
      </tr>
</table>

</body>

</html>
```

class GMap2

Instantiate class GMap2 in order to create a map. This is the central class in the API. Everything else is auxiliary.

Constructor: GMap2(container:Node, opts?:GMapOptions)

Description: Creates a new map inside of the given HTML container, which is typically a DIV element. If no set of map types is given in the optional argument opts.mapTypes, the default set G_DEFAULT_MAP_TYPES is used. If no size is given in the optional argument opts.size, then the size of the container is used. If opts.size is given, then the container element of the map is resized accordingly. See class GMapOptions. Note: a Map needs to be centered before it can be used. You should immediately call GMap2.setCenter() to initialize a map created with this constructor.

Examples:

map = new GMap2(document.getElementById("map_canvas"));

map.setCenter(new GLatLng(-37.91170058826019,145.1345443725586), 14);

Method: addControl(control:GControl, position?:GControlPosition)

Return value: None

Description: Adds the control to the map. The position on the map is determined by the optional position argument. If this argument is absent, the default position of the control is used, as determined by the GControl.getDefaultPosition() method. A control instance must not be added more than once to the map.

Examples:

map.addControl(extLargeMapControl);

// a large pan/zoom control as now used on Google Maps. Appears in the top left corner of the map.

map.addControl(new GMapTypeControl());

// Add functionality to change the GMapType currently used by the map within the Google Maps API.

// By default, a GMapTypeaControl displays a set of standard map types:

// G_NORMAL_MAP displays the normal, default 2D tiles of Google Maps

// G_SATELLITE_MAP displays photographic tiles

// G_HYBRID_MAP displays a mix of photographic tiles and a tile layer for prominent features (roads, city names)

map.addControl(new google.maps.LocalSearch(), new GControlPosition(G_ANCHOR_BOTTOM_RIGHT, new GSize(10,20)));

//search textbox

class GStreetviewPanorama

A GStreetviewPanorama object holds an instance of the Flash® Street View Panorama viewer. Each object of this class can hold a separate instance, showing a unique view. This object is normally embedded within a container object such as a DIV and can be manipulated to change its view using the methods of this class. Street View data is not available for all locations. Use the GStreetviewClient object or the GStreetviewOverlay object to determine if Street View data is available for your location.

Constructor: GStreetviewPanorama(container:Node, opts?:GStreetviewPanoramaOptions)

Description: Creates a new GStreetviewPanorama object with a corresponding flash viewer in the provided container. The viewer will not be shown until a location has been specified, either in the optional GStreetviewPanoramaOptions opts object or by calling setLocationAndPOV.

Example: var myPano = new GStreetviewPanorama(document.getElementById("pano"));

---

namespace GEvent

This namespace contains functions that you use to register event handlers, both for custom events and for DOM events, and to fire custom events. All the events defined by this API are custom events that are internally fired by GEvent.trigger().

Method: GEvent.addListener(source:Object, event:String, handler:Function)

Return value: GEventListener

Description: Registers an event handler for a custom event on the source object. Returns a handle that can be used to eventually deregister the handler. The event handler will be called with this set to the source object.

Example: GEvent.addListener(myPano, "error", handleNoFlash);

---

class GStreetviewOverlay

A GStreetviewOverlay object is a tileset highlighting locations where Street View data is available. This class implements the GOverlay interface and can be added to the map using map.addOverlay and removed using map.removeOverlay.

Constructor: GStreetviewOverlay()

Description: Creates a new GStreetviewOverlay which implements the GOverlay interface.

Example: svOverlay = new GStreetviewOverlay();

---

class GMap2

Method: addOverlay(overlay:GOverlay)

Return value: None

Description: Adds an overlay to the map and fires the addoverlay event.

Example: map.addOverlay(svOverlay);

class GStreetviewPanorama

Method: setLocationAndPOV(latlng:GLatLng, opt_pov:GPov)

Return value: None

Description: Sets the location and POV of the flash viewer. After calling this function, the viewer will display the nearest location to the latlng provided if one is available. If no data is available for this location, then the flash player will remain unchanged and emit an error code. See GStreetviewClient.ReturnValues for the possible response codes.

Example: myPano.setLocationAndPOV(latlng)

---

class GDirections

Constructor: GDirections(map?:GMap2, panel?:Element)

Description: Creates a new instance of a directions object to request and store direction results. This directions object can then create directions given a query using the GDirections.load() method.

The constructor takes an optional map object (to display a polyline of the computed directions) and/or a panel DIV element (to display textual direction results.

If passed a map argument, whenever a new directions result is computed, the polyline and markers associated with the result are automatically added as overlays on the map. Similarly, when passed a panel argument, the textual directions associated with the result are added to the indicated DIV, replacing any existing content in the DIV.

If either of these arguments is null, the associated elements are not retrieved unless explicitly requested in the GDirections.load() method. (See below.)

Additionally, the object contains three event listeners which you can intercept:

- "load": This event is triggered when the results of a directions query issued via GDirections.load() are available. Note that the load() method initiates a new query, which in turn triggers a "load" event once the query has finished loading. The "load" event is triggered before any overlay elements are added to the map/panel.
- "addoverlay": This event is triggered after the polyline and/or textual directions components are added to the map and/or DIV elements. Note that the "addoverlay" event is not triggered if neither of these elements are attached to a GDirections object.
- "error": This event is triggered if a directions request results in an error. Callers can use GDirections.getStatus() to get more information about the error. When an "error" event occurs, no "load" or "addoverlay" events will be triggered.

Example: GDirections(map, document.getElementById("directions"));

EXTLARGEMAPCONTROL.JS

```
/*global GKeyboardHandler, GDraggableObject*/

/**
 * @desc Creates an ExtLargeMapControl. No configuration options are available.
 *
 * @constructor
 */
function ExtLargeMapControl() {
  this.sliderStep = 9;
  this.imgSrc = "http://maps.google.com/intl/en_ALL/mapfiles/mapcontrols3d.png";
}


/**
 * @private
 */
ExtLargeMapControl.prototype = new GControl();


/**
 * @private
 * @type GMap2
 */
ExtLargeMapControl.prototype._map = null;


/**
 * @private
 * @type Element
 */
ExtLargeMapControl.prototype._container = null;


/**
 * @private
 * @type Element
 */
ExtLargeMapControl.prototype._slider = null;


/**
 * @private
 * @type GKeyboardHandler
 */
ExtLargeMapControl.prototype._keyboardhandler = null;


/**
 * @desc Initialize the map control
```

```
 * @private
 */
ExtLargeMapControl.prototype.initialize = function (map) {
  ExtLargeMapControl.prototype._map = map;

  var _handleList = {};

  this._keyboardhandler = new GKeyboardHandler(map);
  var agt = navigator.userAgent.toLowerCase();

  this._is_ie    = ((agt.indexOf("msie") !== -1) && (agt.indexOf("opera") === -
1));
  this._is_gecko = (agt.indexOf('gecko') !== -1);
  this._is_opera = (agt.indexOf("opera") !== -1);

  //common image
  var commonImg = new Image();
  commonImg.src = this.imgSrc;

  // calculation of controller size
  var currentMapType = map.getCurrentMapType();
  var minZoom = parseInt(currentMapType.getMinimumResolution(), 10);
  var maxZoom = 0;
  var maptypes = map.getMapTypes();
  for (var i = 0; i < maptypes.length; i++) {
    if (maptypes[i].getMaximumResolution() > maxZoom) {
      maxZoom = maptypes[i].getMaximumResolution();
    }
  }
  ExtLargeMapControl.prototype._maxZoom = parseInt(maxZoom, 10);
  ExtLargeMapControl.prototype._step = this.sliderStep;
  var ctrlHeight = (86 + 5) + (maxZoom - minZoom + 1) * this.sliderStep + 5;

  // create container
  var container = document.createElement("div");
  container.style.width = "59px";
  container.style.height = (ctrlHeight + this.sliderStep + 2) + "px";
  container.style.overflow = "hidden";
  container.style.padding = "0";
  container.style.MozUserSelect = "none";
  container.style.textAlign = "left";
  _handleList.container = container;
  ExtLargeMapControl.prototype._container = container;

  //image load
  var imgContainer = document.createElement("div");
  imgContainer.style.width = "59px";
  imgContainer.style.height = "62px";
  imgContainer.style.overflow = "hidden";
  if (this._is_ie) {
    imgContainer.style.filter =
"progid:DXImageTransform.Microsoft.AlphaImageLoader(src='" + this.imgSrc + "')";
  }
```

```
  container.appendChild(imgContainer);

  if (!this._is_ie) {
    var baseImg = commonImg.cloneNode(false);
    baseImg.style.position = "absolute";
    baseImg.style.left = "0px";
    baseImg.style.top = "0px";
    baseImg.style.width = "59px";
    baseImg.style.height = "458px";
    imgContainer.appendChild(baseImg);
  }

  //top arrow button
  var topBtn = document.createElement("div");
  topBtn.style.position = "absolute";
  topBtn.style.left = "20px";
  topBtn.style.top = "0px";
  topBtn.style.width = "18px";
  topBtn.style.height = "18px";
  topBtn.style.cursor = "pointer";
  topBtn.style.overflow = "hidden";
  topBtn.title = "up";
  container.appendChild(topBtn);

  //left arrow button
  var leftBtn = topBtn.cloneNode(true);
  leftBtn.style.left = "0px";
  leftBtn.style.top = "20px";
  leftBtn.title = "left";
  container.appendChild(leftBtn);

  //right arrow button
  var rightBtn = topBtn.cloneNode(true);
  rightBtn.style.left = "40px";
  rightBtn.style.top = "20px";
  rightBtn.title = "right";
  container.appendChild(rightBtn);

  //bottom arrow button
  var bottomBtn = topBtn.cloneNode(true);
  bottomBtn.style.left = "20px";

  bottomBtn.style.top = "40px";
  bottomBtn.title = "bottom";
  container.appendChild(bottomBtn);

  //center button
  var homeBtn = topBtn.cloneNode(true);
  homeBtn.style.left = "20px";
  homeBtn.style.top = "20px";
  homeBtn.title = "home position";
  container.appendChild(homeBtn);
```

```
  _handleList.topBtn = topBtn;
  _handleList.leftBtn = leftBtn;
  _handleList.rightBtn = rightBtn;
  _handleList.bottomBtn = bottomBtn;
  _handleList.homeBtn = homeBtn;

  // zoom slider Button
  var zoomSlideBarContainer = document.createElement("div");
  zoomSlideBarContainer.style.position  = "absolute";
  zoomSlideBarContainer.style.left = "19px";
  zoomSlideBarContainer.style.top = "86px";
  zoomSlideBarContainer.style.width = "22px";
  zoomSlideBarContainer.style.height = ((maxZoom - minZoom + 1) * this.sliderStep)
+ "px";
  zoomSlideBarContainer.style.overflow = "hidden";
  zoomSlideBarContainer.style.cursor = "pointer";
  container.appendChild(zoomSlideBarContainer);
  _handleList.slideBar = zoomSlideBarContainer;

  var zoomLevel = map.getZoom();
  //zoomOut Btn load
  var zoomSliderContainer = document.createElement("div");
  zoomSliderContainer.style.position  = "absolute";
  zoomSliderContainer.style.left = 0;
  zoomSliderContainer.style.top = ((maxZoom - zoomLevel) * this.sliderStep + 1) +
"px";
  zoomSliderContainer.style.width = "22px";
  zoomSliderContainer.style.height = "14px";
  zoomSliderContainer.style.overflow = "hidden";
  zoomSliderContainer.style.cursor =
"url(http://maps.google.com/intl/en_ALL/mapfiles/openhand.cur), default";
  zoomSlideBarContainer.appendChild(zoomSliderContainer);
  _handleList.slideBarContainer = zoomSliderContainer;

  if (this._is_ie) {
    var zoomSliderBtnImg = document.createElement("div");
    zoomSliderBtnImg.style.position = "relative";
    zoomSliderBtnImg.style.left = 0;
    zoomSliderBtnImg.style.top = "-384px";
    zoomSliderBtnImg.style.width = "22px";
    zoomSliderBtnImg.style.height = "14px";
    zoomSliderBtnImg.style.overflow = "hidden";
    zoomSliderBtnImg.style.filter =
"progid:DXImageTransform.Microsoft.AlphaImageLoader(src = '" + this.imgSrc + "')";
    zoomSliderContainer.appendChild(zoomSliderBtnImg);
    _handleList.zoomSlider = zoomSliderBtnImg;
  } else {
    var slideImg = commonImg.cloneNode(false);
    slideImg.style.position = "absolute";
    slideImg.style.left = "0px";
    slideImg.style.top = "-384px";
    slideImg.style.MozUserSelect = "none";
    slideImg.style.border = "0px none";
```

```
    slideImg.style.margin = "0px";
    slideImg.style.padding = "0px";
    slideImg.style.width = "59px";
    slideImg.style.height = "458px";
    zoomSliderContainer.appendChild(slideImg);
    _handleList.zoomSlider = slideImg;
  }

  //zoomOut Btn load
  var zoomOutBtnContainer = document.createElement("div");
  zoomOutBtnContainer.style.position = "absolute";
  zoomOutBtnContainer.style.left = 0;
  zoomOutBtnContainer.style.top = (86 + (maxZoom - minZoom + 1) * this.sliderStep)
+ "px";
  zoomOutBtnContainer.style.width = "59px";
  zoomOutBtnContainer.style.height = "23px";
  zoomOutBtnContainer.style.overflow = "hidden";
  container.appendChild(zoomOutBtnContainer);


  if (this._is_ie) {
    var zoomOutBtnImg = document.createElement("div");
    zoomOutBtnImg.style.position = "relative";
    zoomOutBtnImg.style.left = 0;
    zoomOutBtnImg.style.top = "-360px";
    zoomOutBtnImg.style.width = "59px";
    zoomOutBtnImg.style.height = "23px";
    zoomOutBtnImg.style.overflow = "hidden";
    zoomOutBtnImg.style.filter =
"progid:DXImageTransform.Microsoft.AlphaImageLoader(src='" + this.imgSrc + "')";
    zoomOutBtnContainer.appendChild(zoomOutBtnImg);
  } else {
    var btnImg = commonImg.cloneNode(false);
    btnImg.style.position = "absolute";
    btnImg.style.left = "0px";
    btnImg.style.top = "-360px";
    btnImg.style.width = "59px";
    btnImg.style.height = "458px";
    zoomOutBtnContainer.appendChild(btnImg);
  }

  //zoomOut button
  var zoomOutBtn = document.createElement("div");
  zoomOutBtn.style.position = "absolute";
  zoomOutBtn.style.left = "20px";
  zoomOutBtn.style.top = (91 + (maxZoom - minZoom + 1) * this.sliderStep) + "px";
  zoomOutBtn.style.width = "18px";
  zoomOutBtn.style.height = "23px";
  zoomOutBtn.style.cursor = "pointer";
  zoomOutBtn.style.overflow = "hidden";
  zoomOutBtn.title = "zoom out";
  container.appendChild(zoomOutBtn);
  _handleList.zoomOutBtn = zoomOutBtn;
```

```
  //zoomIn button
  var zoomInBtn = document.createElement("div");
  zoomInBtn.style.position = "absolute";
  zoomInBtn.style.left = "20px";
  zoomInBtn.style.top = "65px";
  zoomInBtn.style.width = "18px";
  zoomInBtn.style.height = "23px";
  zoomInBtn.style.cursor = "pointer";
  zoomInBtn.style.overflow = "hidden";
  zoomInBtn.title = "zoom in";
  container.appendChild(zoomInBtn);
  _handleList.zoomInBtn = zoomInBtn;

  // events
  GEvent.addDomListener(_handleList.topBtn, "click",
      GEvent.callback(this, this._eventTop));
  GEvent.addDomListener(_handleList.leftBtn, "click",
      GEvent.callback(this, this._eventLeft));
  GEvent.addDomListener(_handleList.rightBtn, "click",
      GEvent.callback(this, this._eventRight));
  GEvent.addDomListener(_handleList.bottomBtn, "click",
      GEvent.callback(this, this._eventBottom));
  GEvent.addDomListener(_handleList.homeBtn, "click",
      GEvent.callback(this, this._eventHome));
  GEvent.addDomListener(_handleList.zoomOutBtn, "click",
      GEvent.callback(this, this._eventZoomOut));
  GEvent.addDomListener(_handleList.zoomInBtn, "click",
      GEvent.callback(this, this._eventZoomIn));
  GEvent.addDomListener(_handleList.slideBar, "click",
      GEvent.callback(this, this._eventSlideBar));
  GEvent.addListener(map, "zoomend",
      GEvent.callback(this, this._eventZoomEnd));

  var drgOpt = {
    container : _handleList.slideBar
  };
  var drgCtrl = new GDraggableObject(_handleList.slideBarContainer, drgOpt);
  GEvent.addDomListener(drgCtrl, "dragend",
      GEvent.callback(this, this._eventSlideDragEnd));
  ExtLargeMapControl.prototype._slider =  drgCtrl;

  //set current slider position
  this._eventZoomEnd(map.getZoom(), map.getZoom());

  map.getContainer().appendChild(container);

  return container;
};


/**
 * @private
```

```
 */
ExtLargeMapControl.prototype._eventTop = function () {
  ExtLargeMapControl.prototype._map.panDirection(0, 1);
};


/**
 * @private
 */
ExtLargeMapControl.prototype._eventLeft = function () {
  ExtLargeMapControl.prototype._map.panDirection(1, 0);
};

/**
 * @private
 */
ExtLargeMapControl.prototype._eventRight = function () {
  ExtLargeMapControl.prototype._map.panDirection(-1, 0);
};

/**
 * @private
 */
ExtLargeMapControl.prototype._eventBottom = function () {
  ExtLargeMapControl.prototype._map.panDirection(0, -1);
};

/**
 * @private
 */
ExtLargeMapControl.prototype._eventZoomOut = function () {
  ExtLargeMapControl.prototype._map.zoomOut();
};

/**
 * @private
 */
ExtLargeMapControl.prototype._eventZoomIn = function () {
  ExtLargeMapControl.prototype._map.zoomIn();
};


/**
 * @private
 */
ExtLargeMapControl.prototype._eventSlideBar = function (e) {
  var map = this._map;
  //calculate zoomlevel
  var mouseY = e.clientY;
  var slideStep = this._step;
  var maxZoom = this._maxZoom;
  var container = this._container;
```

```
  //set new zoomLevel
  var ctrlPos = this._getDomPosition(container);
  mouseY -= (ctrlPos.y + 91);
  var zoomLevel = Math.floor(maxZoom - (mouseY / slideStep));
  zoomLevel = zoomLevel < 0 ? 0 : zoomLevel;
  map.setZoom(zoomLevel);
};

/**
  * @private
 */
ExtLargeMapControl.prototype._getDomPosition = function (that) {
  var targetEle = that;
  var pos = { x : 0, y : 0 };

  while (targetEle) {
    pos.x += targetEle.offsetLeft;
    pos.y += targetEle.offsetTop;
    targetEle = targetEle.offsetParent;

    if (targetEle && this._is_ie) {
      pos.x += (parseInt(ExtLargeMapControl.getElementStyle(targetEle,
          "borderLeftWidth", "border-left-width"), 10) || 0);
      pos.y += (parseInt(ExtLargeMapControl.getElementStyle(targetEle,
          "borderTopWidth", "border-top-width"), 10) || 0);
    }
  }

  if (this._is_gecko) {
    var bd = document.getElementsByTagName("BODY")[0];
    pos.x += 2 * (parseInt(ExtLargeMapControl.getElementStyle(bd,
        "borderLeftWidth", "border-left-width"), 10) || 0);
    pos.y += 2 * (parseInt(ExtLargeMapControl.getElementStyle(bd,
        "borderTopWidth", "border-top-width"), 10) || 0);
  }
  return pos;
};


/**
 * @private
 */
ExtLargeMapControl.getElementStyle = function(targetElm, IEStyleProp,
CSSStyleProp) {
  var elem = targetElm;
  if (elem.currentStyle) {
    return elem.currentStyle[IEStyleProp];
  } else if (window.getComputedStyle) {
    var compStyle = window.getComputedStyle(elem, "");
    return compStyle.getPropertyValue(CSSStyleProp);
  }
};
```

```
/**
 * @private
 */
ExtLargeMapControl.prototype._eventSlideDragEnd = function (e) {
  //calculate zoomlevel
  var maxZoom = this._maxZoom;
  var mouseY = this._slider.top;
  var step = this._step;

  //set new zoomLevel
  var zoomLevel = Math.floor(maxZoom - (mouseY / step));
  zoomLevel = zoomLevel < 0 ? 0 : zoomLevel;
  this._map.setZoom(zoomLevel);
};


/**
 * @private
 */
ExtLargeMapControl.prototype._eventHome = function () {
  this._map.returnToSavedPosition();
};


/**
 * @private
 */
ExtLargeMapControl.prototype._eventZoomEnd = function (oldZoom, newZoom) {
  var maxZoom = this._maxZoom;
  var step = this._step;
  this._slider.moveTo(new GPoint(0, (maxZoom - newZoom) * step));
};


/**
 * @private
 * @ignore
 */
ExtLargeMapControl.prototype.getDefaultPosition = function () {
  return new GControlPosition(G_ANCHOR_TOP_LEFT, new GSize(10, 10));
};


/**
 * @private
 * @ignore
 */
ExtLargeMapControl.prototype.selectable = function () {
  return false;
};

/**
```

```
 * @private
 * @ignore
 */
ExtLargeMapControl.prototype.printable = function () {
  return true;
};
```

EXTLARGEMAPCONTROL_PACKED.JS

```
eval(function(p,a,c,k,e,r){e=function(c){return(c<a?'':e(parseInt(c/a)))+((c=c%a)>35?
)if(k[c])p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);return p}('H 7(){4.Z=9;4.
1)&&(y.1n("2e")===-1));4.2b=(y.1n(\'36\')!==-1);4.34=(y.1n("2e")!==-1);6 o=13 31();o.
q=M.S("R");q.3.I="12";q.3.K=(g+4.Z+2)+"1b";q.3.Q="P";q.3.2c="0";q.3.2a="1z";q.3.35="F
e=o.11(1i);e.3.L="U";e.3.F="T";e.3.G="T";e.3.I="12";e.3.K="1B";j.J(e)}6 D=M.S("R");D.
c=D.11(18);c.3.F="1K";c.3.G="Y";c.15="2z";q.J(c);6 B=D.11(18);B.3.F="Y";B.3.G="1K";B.
r+1)*4.Z)+"1b";w.3.Q="P";w.3.1d="1o";q.J(w);k.1p=w;6 n=a.1x();6 p=M.S("R");p.3.L="U";
2k";f.3.I="1q";f.3.K="22";f.3.Q="P";f.3.1t="1s:1w.1v.1u(1g = \'"+4.19+"\')";p.J(f);k.
l=M.S("R");l.3.L="U";l.3.F=0;l.3.G=(1C+(m-r+1)*4.Z)+"1b";l.3.I="12";l.3.K="1m";l.3.Q=
A=o.11(1i);A.3.L="U";A.3.F="T";A.3.G="-2h";A.3.I="12";A.3.K="1B";l.J(A)}6 u=M.S("R");
s=M.S("R");s.3.L="U";s.3.F="Y";s.3.G="38";s.3.I="1f";s.3.K="1m";s.3.1d="1o";s.3.Q="P"
37";q.J(s);k.1D=s;E.V(k.1I,"X",E.O(4,4.29));E.V(k.1H,"X",E.O(4,4.28));E.V(k.1F,"X",E.
z={26:k.1p};6 x=13 2Z(k.2m,z);E.V(x,"2Y",E.O(4,4.1X));7.8.1j=x;4.1r(a.1x(),a.1x());a.
1)};7.8.23=H(){7.8.N.2W()};7.8.21=H(){7.8.N.2V()};7.8.1P=H(e){6 b=4.N;6 f=e.2T;6 c=4.
a={x:0,y:0};2S(d){a.x+=d.2R;a.y+=d.2Q;d=d.2P;W(d&&4.17){a.x+=(16(7.1a(d,"1Q","1c-F-I"
a};7.1a=H(a,d,c){6 b=a;W(b.1N){14 b.1N[d]}1A W(2i.1M){6 e=2i.1M(b,"");14 e.2K(c)}};7.
2G(3f,13 2F(10,10))};7.8.2E=H(){14 1i};7.8.2D=H(){14
18};',62,206,'|||style|this||var|ExtLargeMapControl|prototype|||||||||||||||||||||
|cloneNode|59px|new|return|title|parseInt|_is_ie|true|imgSrc|getElementStyle|px|borde
om|_container|none|else|458px|86|zoomInBtn|homeBtn|rightBtn|getMaximumResolution|left
idth|_keyboardhandler|com|_eventZoomIn|14px|_eventZoomOut|_eventHome|_eventBottom|con
6px|19px|navigator|png|home|length|GKeyboardHandler|mapcontrols3d|bottom|right|for|ge
ze|offsetParent|offsetTop|offsetLeft|while|clientY|getCurrentMapType|zoomIn|zoomOut|g
toLowerCase'.split('|'),0,{}))
```

3.2. EXAMPLE 2: DISPLAYING PHOTOS/WIKIPEDIA INFORMATION

- URL: http://www.geocities.com/quocthai0512/example2.html
- Description:
  - This example allows users to enter an address and select to view photos or Wikipedia information of corresponding place.
- Screenshots:

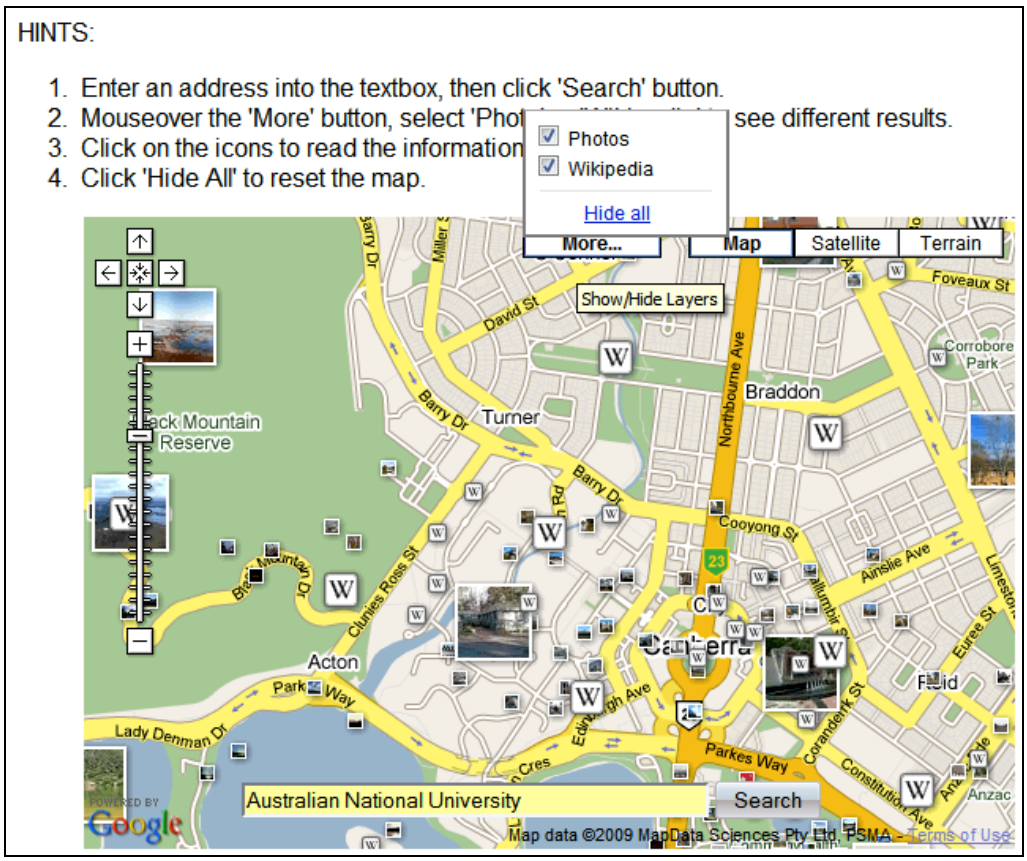


Figure 2 – Display Photos or Wikipedia Information

EXAMPLE2.HTML

```
<html>

<head>

	<link rel="stylesheet" type="text/css" href="include.css" />

	<style type="text/css">

		body {
			height:600px;
			font-family: Arial;
		}

		#mapdiv {
			position: absolute;
			left:50px;
			width:590px;
			height:400px;
			overflow: hidden;
		}

		#search {
			position: absolute;
			top:500px; left:150px;
		}

		#box {
			text-align:left;
			font-size:12px;
			padding: 6px 4px;
			width:120px;
			background-color: #fff;
			border: 1px solid gray;
			border-top:1px solid #a5a5a5;
			display: none;
			cursor:default;
		}

		#box.highlight {
			width:119px;
			border-width:2px;
			border-top:1px solid #a5a5a5;
		}

		#more_inner {
			text-align:center;
			font-size:12px;
			background-color: #fff;
			border: 1px solid #fff;
			border-bottom-color: #b0b0b0;
```

```
                  border-right-color: #b0b0b0;
                  width:7em;
                  cursor: pointer;
            }

            #more_inner.highlight {
                  font-weight: bold;
                  border: 1px solid #483D8B;
                  border-bottom-color: #6495ed;
                  border-right-color: #6495ed;
            }

            #boxlink {
                  color: #a5a5a5;
                  text-decoration: none;
                  cursor: default;
                  margin-left: 33px;
            }

            #boxlink.highlight {
                  color: #0000cd;
                  text-decoration: underline;
                  cursor: pointer;
            }

      </style>

<script
src="http://maps.google.com/maps?file=api&amp;v=2.x&amp;key=ABQIAAAAtWCOvsQE3_q3_qb8k
Y-VG3o8b6wZyHyJA" type="text/javascript"></script>

</head>

<body onload="buildMap()" onunload="GUnload()">

<p>HINTS:</p>

<ol>
      <li>Enter an address into the textbox, then click 'Search' button.</li>
      <li>Mouseover the 'More' button, select 'Photo' or 'Wikipedia' to see different
      <li>Click on the icons to read the information or view pictures.</li>
      <li>Click 'Hide All' to reset the map.</li>
</ol>

<div id="mapdiv"> </div>

<form action="" onsubmit="showAddress(this.addr.value);return false">
<div id="search">
      <input type="text" size="45" id="addr" name="addr" title="Placename or address"
      <input type="submit" value=" Search " />
</div>
</form>
```

```
<!-- Second part of More button html. The first part will be created inside MoreContr

<form action="">
<div id="box" onmouseout="setClose(event)">
      <input name="mark" type="checkbox" onclick="switchLayer(this.checked, layers[0]
      <input name="mark" type="checkbox" onclick="switchLayer(this.checked, layers[1]
      <hr style="width:92%;text-align:center;height:1px;border:1px;color:#e2e2e2;back
      <a id="boxlink" href="javascript:void(0)" onclick="hideAll()">Hide all</a>
</div>
</form>

<script type="text/javascript">

//<![CDATA[

      var map, timer;
      var chosen = [];

      / * Array of GLayers
        * The 'name' label is not being used here
        * /
      var layers = [
            { name: "Pano", obj: new GLayer("lmc:panoramio") },
            { name: "Wiki", obj: new GLayer("lmc:wikipedia_en") }
      ];

      function buildMap() {

            map = new GMap2(document.getElementById("mapdiv"));
            map.setCenter(new GLatLng(-37.911700588260l9,145.1345443725586), 2);
            map.addControl(new GLargeMapControl());
            map.enableScrollWheelZoom();

            // Add selectable terrain map
            map.addMapType(G_PHYSICAL_MAP);
            var hControl = new GHierarchicalMapTypeControl();
            hControl.addRelationship(G_SATELLITE_MAP, G_HYBRID_MAP, "Labels",  false)
            map.addControl(hControl);

            // Add the self created control
            map.addControl(new MoreControl());

      }

      function hideAll() {

            var boxes = document.getElementsByName("mark");

            for(var i = 0; i < boxes.length; i++) {

                  if(boxes[i].checked) {
                        boxes[i].checked = false;
                        switchLayer(false, layers[i].obj);
```

```
                    chosen.push(i);
                }

        }

}


function checkChecked() {

/ * Returns true if a checkbox is still checked
  * otherwise false
  * /

var boxes = document.getElementsByName("mark");

for(var i = 0; i < boxes.length; i++) {
      if(boxes[i].checked) return true;
}

return false;

}

function switchLayer(checked, layer) {

/ * Function was originally borrowed from Esa:
  * http://esa.ilmari.googlepages.com/dropdownmenu.htm
  * /

      var layerbox = document.getElementById("box");
      var boxlink = document.getElementById("boxlink");
      var button = document.getElementById("more_inner");

      if(checked) {

            map.addOverlay(layer);

            // Reset chosen array
            chosen.length = 0;

            / * Highlight the link and
              * make the button font bold.
              * /
            boxlink.className ="highlight";
            layerbox.className ="highlight";
            button.className ="highlight";

      }

      else {

            map.removeOverlay(layer);
```

```
            /* Reset the link and the button
             * if all checkboxes were unchecked.
             */

            if(!checkChecked()) {
                  boxlink.blur();
                  boxlink.className ="";
                  layerbox.className ="";
                  button.className ="";
            }
      }
}


function showLayerbox() {

      var layerbox = document.getElementById("box");
      // Left size of more control plus mapdiv.style.left
      var offsetX = 278 + 50;
       // Top size of more control plus mapdiv.style.top plus more button heigh
       var offsetY = 7 + 50 + 18;

      var lpos = new GControlPosition(G_ANCHOR_TOP_LEFT, new GSize(offsetX, off
      lpos.apply(layerbox);
      if(window.timer) clearTimeout(timer);
      layerbox.style.display = "block";
}


function setClose(e) {

      if(!e) e = window.event;

      var layerbox = document.getElementById("box");

      if(checkMouseLeave(layerbox, e))
            timer = setTimeout(function() {layerbox.style.display = "none"; },

}

function checkMouseLeave(element, evt) {

/* Avoid firing a mouseout event
 * when the mouse moves over a child element.
 * Borrowed from:
 * http://www.faqts.com/knowledge_base/view.phtml/aid/1606/fid/145
 */

      if(element.contains && evt.toElement) {
            return !element.contains(evt.toElement);
      }
      else if(evt.relatedTarget) {
```

```
                return !containsDOM(element, evt.relatedTarget);
        }

}


function containsDOM(container, containee) {

        var isParent = false;

        do {
                if((isParent = container == containee))
                break;
                containee = containee.parentNode;
        }
        while(containee != null);

        return isParent;

}


function toggleLayers() {

        if(chosen.length > 0 ) {

        /* Make an independent copy of chosen array since switchLayer()
         * resets the chosen array, which may not be useful here.
         */

        var copy = chosen.slice();

                for(var i = 0; i < copy.length; i++) {

                        var index = parseInt(copy[i]);
                        switchLayer(true, layers[index].obj);
                        document.getElementsByName("mark")[index].checked = true;

                }
        }

        else {
                hideAll();
        }

}


function MoreControl(){};

MoreControl.prototype = new GControl();
MoreControl.prototype.initialize = function(map) {
```

```
            var more = document.createElement("div");
            more.style.border = "1px solid black";
            more.title = "Show/Hide Layers";
            var inner = document.createElement("div");
            inner.id = "more_inner";
            inner.appendChild(document.createTextNode("More..."));
            more.appendChild(inner);
            more.onmouseover = showLayerbox;
            more.onmouseout = setClose;
            more.onclick = toggleLayers;
            map.getContainer().appendChild(more);
            return more;

      }

      MoreControl.prototype.getDefaultPosition = function() {
            return new GControlPosition(G_ANCHOR_TOP_LEFT, new GSize(278, 7));
      }

      function showAddress(address) {

            var geocoder = new GClientGeocoder();
            geocoder.getLatLng(address, function(point) {

                  if(!point) {
                        alert(address + " not found.");
                  }
                  else {
                        map.setCenter(point, 14);
                  }
            });
      }

//]]>
</script>

</body>
</html>
```

class GLayer

This class instantiates a predefined "layer" overlay consisting of a collection of related items. It implements the GOverlay interface and thus is added to the map using the GMap2.addOverlay() method.

Constructor: GLayer(layerId:String)

Description: Creates a layer using the given unique Layer ID. http://spreadsheets.google.com/pub?key=p9pdwsai2hDN-cAocTLhnag contains a list of the currently supported layers.

Example: new GLayer("lmc:panoramio")

---

class GMap2

Method: enableScrollWheelZoom()

Return value: none

Description: Enables zooming using a mouse's scroll wheel. Note: scroll wheel zoom is disabled by default.

Example: map.enableScrollWheelZoom();

Method: addMapType(type:GMapType)

Return value: none

Description: Adds a new map type to the map.

Example: map.addMapType(G_PHYSICAL_MAP);

---

class GHierarchicalMapTypeControl

The GHierarchicalMapTypeControl provides a "nested" map type control for selecting and switching between supported map types via buttons and nested checkboxes. Controls will be made available for all map types currently attached to the map at the time the control is constructed.

Map types added to the map appear as buttons as in the normal GMapTypeControl. However, map types set as sub-types of other map types (see the addRelationship() method below) will appear as checkbox sub-menu items below the parent button.

By default, maps support the set of G_DEFAULT_MAP_TYPES, though maps may also add map types explicitly via GMap2.addMapType(). Controls will be made available for all map types currently attached to the map at the time the control is constructed. Note that because sub-types appear as checkboxes, you can toggle their appearance with their parents or their siblings in the sub-menu.

Constructor: GHierarchicalMapTypeControl()

Description: Constructs the control. By default, the G_HYBRID_MAP map type is made a child of the G_SATELLITE_MAP map type. If this is not desired, the relationship can be removed by calling the clearRelationships() method.

Example: var hControl = new GHierarchicalMapTypeControl();

Method: addRelationship(parentType:GMapType, childType:GMapType, childText?:String, isDefault?:Boolean)

Return value: None

Description: Registers a parent/child relationship between map types with the control. If childText is given, it will be displayed next to the checkbox for the child map type instead of its name. If isDefault is true, the child map type will be selected by default. Note that all relationships must be set up before the control is added. (Adding relationships after the control is added will have no effect.)

Example: hControl.addRelationship(G_SATELLITE_MAP, G_HYBRID_MAP, "Labels", false);

3.3. EXAMPLE 3: EXPERIMENTING OVERLAPS

- URL: http://www.geocities.com/quocthai0512/example3.html
- Description:
  - First, users are required to enter an address into the textbox at the bottom left corner.
  - The marker A in figure 3a denotes the query result.
  - Users may click on any place of the map to drag a circle which radius will be displayed in the Filter box.
  - Then, users may resize the circle so that it can overlap point A. If an overlap has occurred, then the link to A (bottom left) is enabled (Figure 3b). Otherwise, it is disabled (Figure 3a).
  - Circles can also be removed from the Filter box.
- Screenshots:

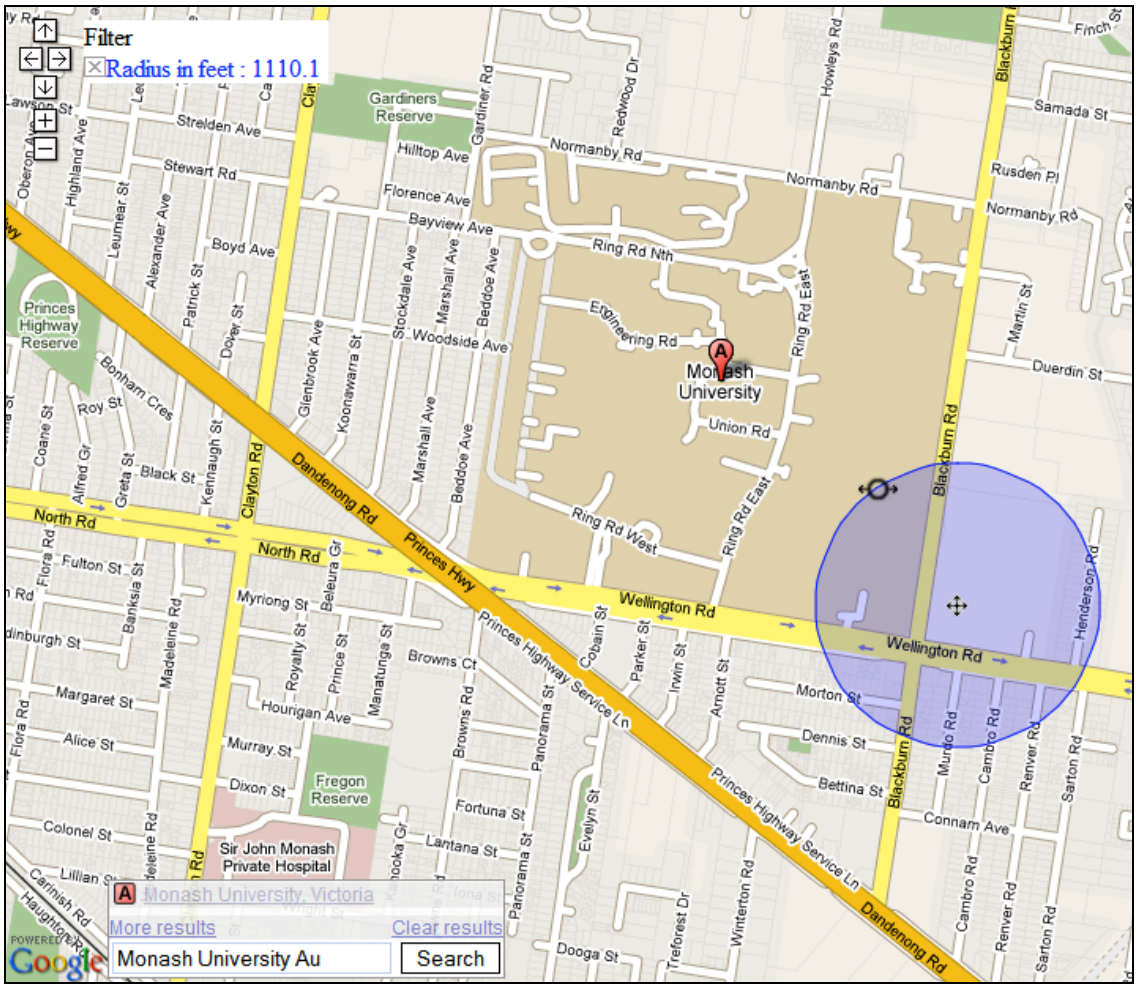


Figure 3a

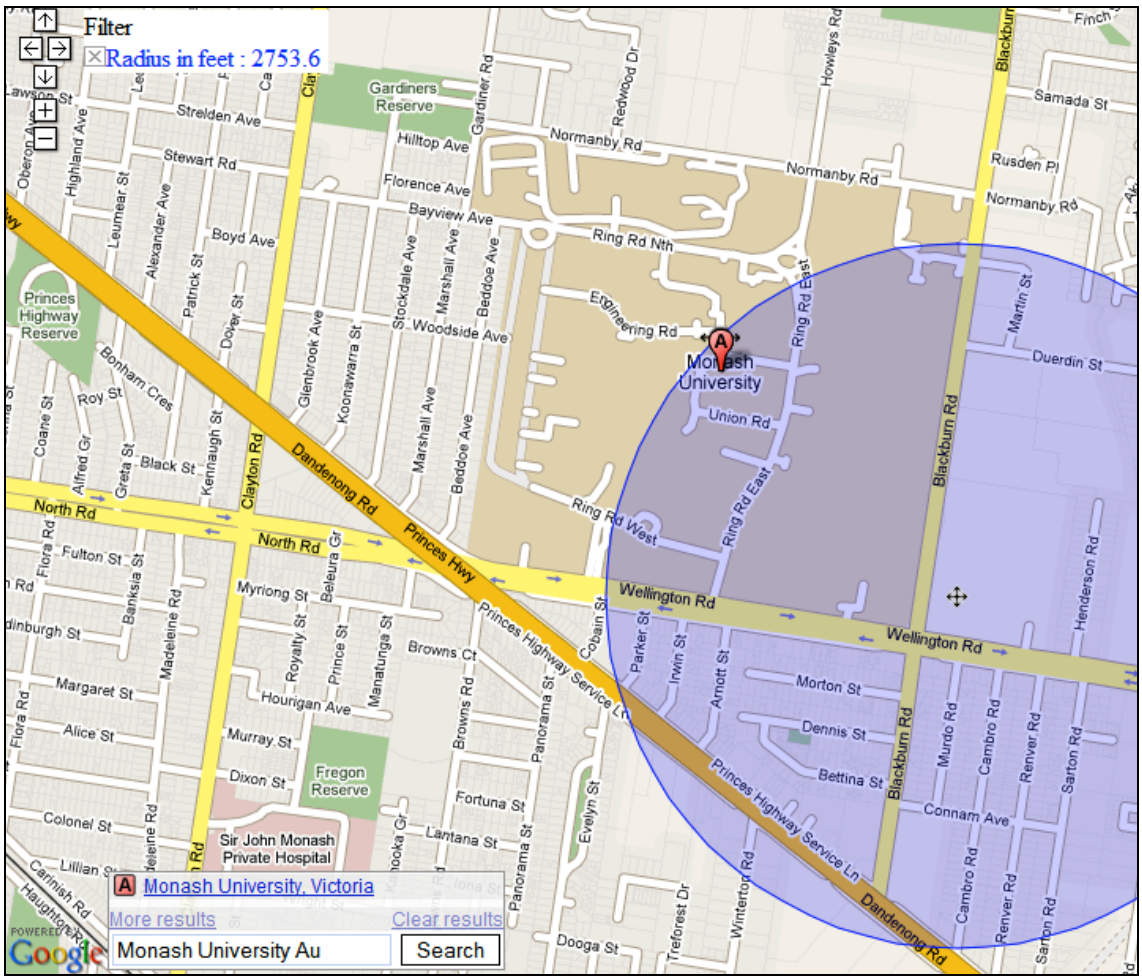

Filter
Radius in feet : 2753.6
Monash University
Monash University, Victoria
More results
Clear results
Monash University Au
Search
Gardiners Reserve
Princes Highway Reserve
Fregon Reserve
Sir John Monash Private Hospital
Dandenong Rd
Princes Hwy
North Rd
Wellington Rd
Princes Highway Service Ln
Ring Rd Nth
Ring Rd West
Ring Rd East
Blackburn Rd
Union Rd
Normanby Rd
Clayton Rd
Woodside Ave
Bayview Ave
Florence Ave
Hilltop Ave
Stewart Rd
Strelden Ave
Boyd Ave
Myrriong St
Hourigan Ave
Margaret St
Alice St
Murray St
Dixon St
Colonel St
Fortuna St
Lantana St
Browns Ct
Browns Rd
Morton St
Dennis St
Bettina St
Connam Ave
Duerdin St
Martin St
Samada St
Rusden Pl
Henderson Rd
Sarton Rd
Renver Rd
Cambro Rd
Murdo Rd
Arnott St
Irwin St
Parker St
Cobain St
Dooga St
Wintertod Rd
Treforest Dr
Evelyn St
Panorama St
Manatunga St
Prince St
Royalty St
Belaura Gr
Madeleine Rd
Banksia St
Flora Rd
Fulton St
Roy St
Bonham Cres
Coane St
Alfred Gr
Greta St
Black St
Kennaugh St
Dover St
Patrick St
Alexander Ave
Leumear St
Highland Ave
Oberon Ave
Carinish Rd
Haughton Rd
Lillian St
Colonel St
Gardiner Rd
Redwood Dr
Howleys Rd
Finch
Stockdale Ave
Marshall Ave
Beddoe Ave
Koonawarra St
Glenbrook Ave
Engineering Rd
Powered by Google

Figure 3b

EXAMPLE3.HTML

```
<html">

<head>

<script
src="http://maps.google.com/maps?file=api&amp;v=2.x&amp;key=ABQIAAAAtWCOvsQE3_q3_qb8k
Y-VG3o8b6wZyHyJA" type="text/javascript"></script>

<script src="http://www.google.com/uds/api?file=uds.js&amp;v=1.0&key=ABQIAAAAxe4iZkOi
P40HsMRTNwwIDB3X2PJ0_br5ee44ut2pm8RRiA2ku6cwsTFtWlCY7kcRdnEPIDA" type="text/javascrip

<script src="http://www.google.com/uds/solutions/localsearch/gmlocalsearch.js?adsense
type="text/javascript"></script>

<script type="text/javascript" language="JavaScript">

      var myMap = null;
      var localSearch = null;
      var myQueryControl = null;

      function displayMap(){

            myMap = new GMap2(document.getElementById("map"));
            myMap.setCenter(new GLatLng(41.967044,-73.835253), 14);

            myMap.addControl(new GSmallMapControl());
            myMap.addControl(new GMapTypeControl());

            localSearch = new google.maps.LocalSearch();
            //{externalAds : document.getElementById("ads")});
            myMap.addControl(localSearch);
            myQueryControl = new QueryControl(localSearch);
            myMap.addControl(myQueryControl);

            GEvent.addListener(myMap, "click", function(overlay, point) {

                  if (point) {

                        singleClick = !singleClick;
                        setTimeout("if (singleClick) createCircle(new GLatLng("+ poin

                  }

            });
      }

</script>

<style type="text/css">
```

```
	@import url("http://www.google.com/uds/css/gsearch.css");
	@import url("http://www.google.com/uds/solutions/localsearch/gmlocalsearch.css"

	div#GQueryControl {
		background-color: white;
		width: 155;
	}

</style>

</head>

<body onload="displayMap();" style="width: 100%; height: 100%; margin: 0px;">

<div id="map" style="width: 100%; height: 100%;"></div>

<script>

	var metric = false;
	var singleClick = false;
	var queryCenterOptions = new Object();
	var queryLineOptions = new Object();

	queryCenterOptions.icon = new GIcon();
	queryCenterOptions.icon.image = "http://jfno.net/images/centerArrow.png";
	queryCenterOptions.icon.iconSize = new GSize(20,20);
	queryCenterOptions.icon.shadowSize = new GSize(0, 0);
	queryCenterOptions.icon.iconAnchor = new GPoint(10, 10);
	queryCenterOptions.draggable = true;
	queryCenterOptions.bouncy = false;

	queryLineOptions.icon = new GIcon();
	queryLineOptions.icon.image = "http://jfno.net/images/resizeArrow.png";
	queryLineOptions.icon.iconSize = new GSize(25,20);
	queryLineOptions.icon.shadowSize = new GSize(0, 0);
	queryLineOptions.icon.iconAnchor = new GPoint(12, 10);
	queryLineOptions.draggable = true;
	queryLineOptions.bouncy = false;

	function createCircle(point, radius) {

		singleClick = false;
		geoQuery = new GeoQuery();
		geoQuery.initializeCircle(radius, point, myMap);
		myQueryControl.addGeoQuery(geoQuery);
		geoQuery.render();

	}

	function destination(orig, hdng, dist) {

		var R = 6371; // earth's mean radius in km
		var oX, oY;
```

```
        var x, y;
        var d = dist/R;  // d = angular distance covered on earth's surface
        hdng = hdng * Math.PI / 180; // degrees to radians
        oX = orig.x * Math.PI / 180;
        oY = orig.y * Math.PI / 180;

        y = Math.asin( Math.sin(oY)*Math.cos(d) + Math.cos(oY)*Math.sin(d)*Math.c
        x = oX + Math.atan2(Math.sin(hdng)*Math.sin(d)*Math.cos(oY), Math.cos(d)-

        y = y * 180 / Math.PI;
        x = x * 180 / Math.PI;
        return new GLatLng(y, x);

}

function distance(point1, point2) {

        var R = 6371; // earth's mean radius in km
        var lon1 = point1.lng()* Math.PI / 180;
        var lat1 = point1.lat() * Math.PI / 180;
        var lon2 = point2.lng() * Math.PI / 180;
        var lat2 = point2.lat() * Math.PI / 180;

        var deltaLat = lat1 - lat2
        var deltaLon = lon1 - lon2

        var step1 = Math.pow(Math.sin(deltaLat/2), 2) + Math.cos(lat2) * Math.cos
        var step2 = 2 * Math.atan2(Math.sqrt(step1), Math.sqrt(1 - step1));

        return step2 * R;

}

function GeoQuery() {}

        GeoQuery.prototype.CIRCLE='circle';
        GeoQuery.prototype.COLORS=["#0000ff", "#00ff00", "#ff0000"];
        var COLORI=0;

        GeoQuery.prototype = new GeoQuery();
        GeoQuery.prototype._map;
        GeoQuery.prototype._type;
        GeoQuery.prototype._radius;
        GeoQuery.prototype._dragHandle;
        GeoQuery.prototype._centerHandle;
        GeoQuery.prototype._polyline;
        GeoQuery.prototype._color ;
        GeoQuery.prototype._control;
        GeoQuery.prototype._points;
        GeoQuery.prototype._dragHandlePosition;
        GeoQuery.prototype._centerHandlePosition;

        GeoQuery.prototype.initializeCircle = function(radius, point, map) {
```

```
        this._type = this.CIRCLE;
        this._radius = radius;
        this._map = map;
        this._dragHandlePosition = destination(point, 90, this._radius/1000
        this._dragHandle = new GMarker(this._dragHandlePosition, queryLineO
        this._centerHandlePosition = point;
        this._centerHandle = new GMarker(this._centerHandlePosition, queryC
        this._color = this.COLORS[COLORI++ % 3];
        map.addOverlay(this._dragHandle);
        map.addOverlay(this._centerHandle);
        var myObject = this;

        GEvent.addListener (this._dragHandle, "dragend", function()
        {myObject.updateCircle(1);});

        GEvent.addListener (this._dragHandle, "drag", function() myObject.u

        GEvent.addListener(this._centerHandle, "dragend", function() {myObj

        GEvent.addListener(this._centerHandle, "drag", function() {myObject

}

GeoQuery.prototype.updateCircle = function (type) {
      this._map.removeOverlay(this._polyline);

      if (type==1) {

            this._dragHandlePosition = this._dragHandle.getPoint();
            this._radius = distance(this._centerHandlePosition, this._dra
            this.render();

      } else {

            this._centerHandlePosition =this._centerHandle.getPoint();
            this.render();
            this._dragHandle.setPoint(this.getEast());

      }

}

GeoQuery.prototype.render = function() {

      if (this._type == this.CIRCLE) {

            this._points = [];
            var distance = this._radius/1000;

            for (i = 0; i < 72; i++) {

                  this._points.push(destination(this._centerHandlePositio
```

```
                }

                this._points.push(destination(this._centerHandlePosition, 0,
                //this._polyline = new GPolyline(this._points, this._color, 6
                this._polyline = new GPolygon(this._points, this._color, 1, 1
                this._map.addOverlay(this._polyline)
                this._control.render();

        }

}

GeoQuery.prototype.remove = function() {

        this._map.removeOverlay(this._polyline);
        this._map.removeOverlay(this._dragHandle);
        this._map.removeOverlay(this._centerHandle);

}

GeoQuery.prototype.getRadius = function() {
        return this._radius;
}

GeoQuery.prototype.getHTML = function() {

        return "<span><font color='"+ this._color + "'>" + this.getDistHtm

}

GeoQuery.prototype.getDistHtml = function() {
        result = "<img src='http://jfno.net/images/close.gif' onClick='myQu
        this._control.getIndex(this) + ");'/>Radius ";

        if (metric) {
                if (this._radius < 1000) {
                        result += "in meters : " + this._radius.toFixed(1);
                } else {
                        result += "in kilometers : " + (this._radius / 1000).to
                }
        } else {
                var radius = this._radius * 3.2808399;
                if (radius < 5280) {
                        result += "in feet : " + radius.toFixed(1);
                } else {
                        result += "in miles : " + (radius / 5280).toFixed(1);
                }
        }
        return result;
}

GeoQuery.prototype.getNorth = function() {
        return this._points[0];
```

```
}

GeoQuery.prototype.getSouth = function() {
      return this._points[(72/2)];
}

GeoQuery.prototype.getEast = function() {
      return this._points[(72/4)];
}

GeoQuery.prototype.getWest = function() {
      return this._points[(72/4*3)];
}

function QueryControl (localSearch) {
      this._localSearch = localSearch;
}

QueryControl.prototype = new GControl();
QueryControl.prototype._geoQueries ;
QueryControl.prototype._mainDiv;
QueryControl.prototype._queriesDiv;
QueryControl.prototype._minStar;
QueryControl.prototype._minPrice;
QueryControl.prototype._maxPrice;
QueryControl.prototype._timeout;
QueryControl.prototype._localSearch;

QueryControl.prototype.initialize = function(map) {
      this._mainDiv = document.createElement("div");
      this._mainDiv.id = "GQueryControl";
      titleDiv = document.createElement("div");
      titleDiv.id = "GQueryControlTitle";
      titleDiv.appendChild(document.createTextNode("Filter"));
      this._mainDiv.appendChild(titleDiv);
      this._queriesDiv = document.createElement("div");
      this._queriesDiv.id = "queriesDiv";
      this._mainDiv.appendChild(this._queriesDiv);

      map.getContainer().appendChild(this._mainDiv);
      this._geoQueries = new Array();
      return this._mainDiv;
}

QueryControl.prototype.getDefaultPosition = function() {
      return new GControlPosition(G_ANCHOR_TOP_LEFT, new GSize(50, 10));
}

QueryControl.prototype.addGeoQuery = function(geoQuery) {
      this._geoQueries.push(geoQuery);
      geoQuery._control = this;
      newDiv = document.createElement("div");
      newDiv.innerHTML = geoQuery.getHTML();
```

```
        this._queriesDiv.appendChild(newDiv);
    }

    QueryControl.prototype.render = function() {
        for (i = 0; i < this._geoQueries.length; i++) {
        geoQuery = this._geoQueries[i];
        this._queriesDiv.childNodes[i].innerHTML = geoQuery.getHTML();
        }

        if (this._timeout == null) {
            this._timeout = setTimeout(myQueryControl.query, 1000);
        } else {
            clearTimeout(this._timeout);
            this._timeout = setTimeout(myQueryControl.query, 1000);
        }
    }

    QueryControl.prototype.query = function() {
        listMarkers = myQueryControl._localSearch.markers.slice();
        for (i = 0; i < listMarkers.length; i++) {
        marker = listMarkers[i].marker;
        result = listMarkers[i].resultsListItem;
        listImage = marker.getIcon().image;
        inCircle = true;
        for (j = 0; j < myQueryControl._geoQueries.length; j++) {
            geoQuery = myQueryControl._geoQueries[j];
        dist = distance(marker.getLatLng(), geoQuery._centerHandlePosition)
            if (dist > geoQuery._radius / 1000) {
                inCircle = false;
                break;
            }
        }
        if (inCircle) {
            marker.setImage(listImage);
            result.childNodes[1].style.color = '#0000cc';
            } else {
                var re = new RegExp(".*(marker.\.png)");
                marker.setImage(listImage.replace(re, "img/$1"));
                result.childNodes[1].style.color = '#b0b0cc';
            }
        }
    }

    QueryControl.prototype.remove = function(index) {
        this._geoQueries[index].remove();
        this._queriesDiv.removeChild(this._queriesDiv.childNodes[index];
        delete this._geoQueries[index];
        this._geoQueries.splice(index,1);
        this.render();
    }

    QueryControl.prototype.getIndex = function(geoQuery) {
        for (i = 0; i < this._geoQueries.length; i++) {
```

```
                        if (geoQuery == this._geoQueries[i]) {
                              return i;
                        }
                  }
                  return -1;
            }
</script>

</body>

</html>
```

3.4. EXAMPLE 4: FINDING NEAREST BARS/RESTAURANTS

- URL: http://www.geocities.com/quocthai0512/example4.html
- Description:
  - Users are required to enter an address to search for nearest bars or restaurants to that address.
  - The query results on the right column are provided by TrueLocal.
  - Users may click on the markers on the map to view information corresponding to that place.
  - We can change the types of business (Schools, Shopping Centres, etc.) by modifying the "catInput" value in the code.
  - Example 9 is a modified version of this example by adding a marker to denote the address being searched.
- Screenshots:

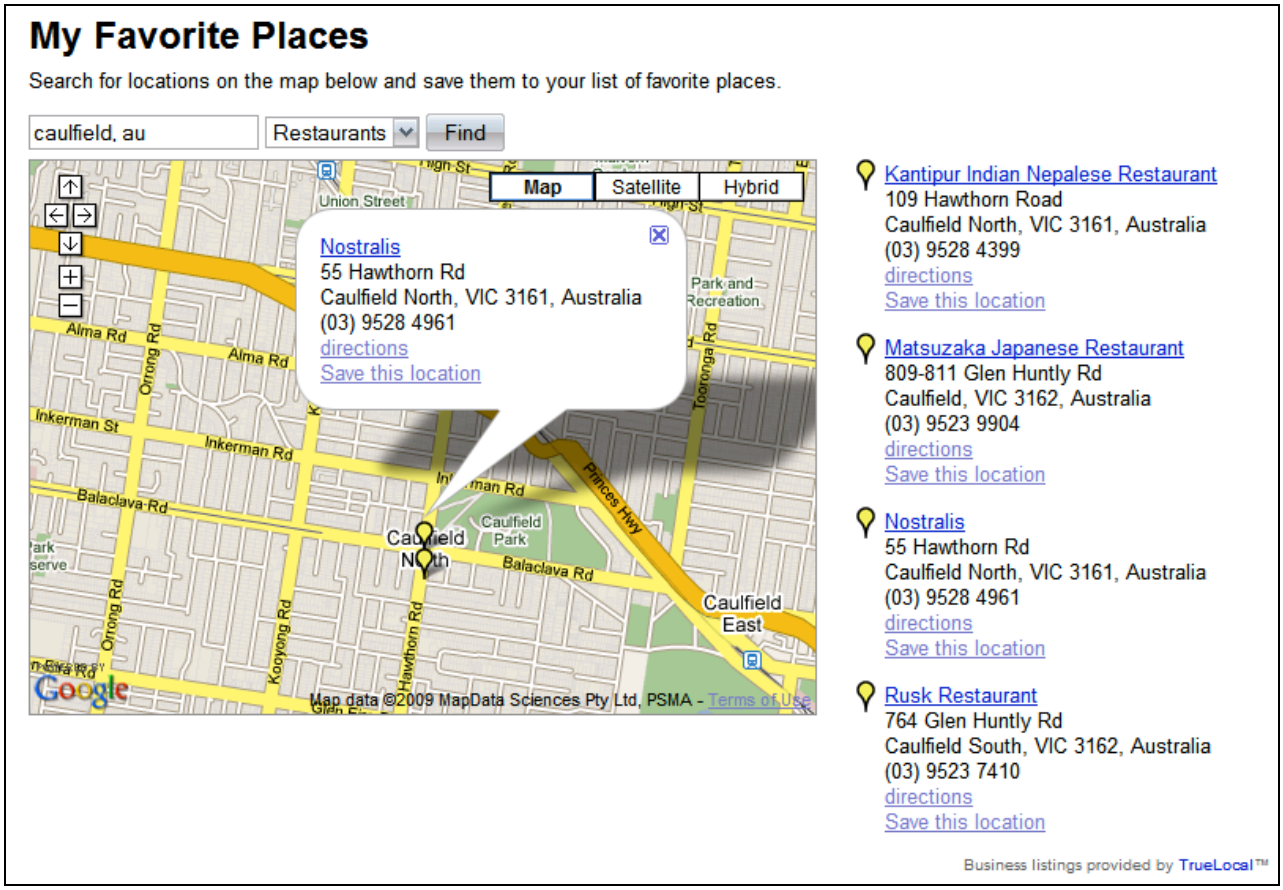


Figure 4 – Display Nearest Bars/Restaurants

EXAMPLE4.HTML

```
<html>

<head>

<link href="http://www.google.com/uds/css/gsearch.css" rel="stylesheet" type="text/cs

<style type="text/css">

      body {
            background-color: white;
            color: black;
            font-family: Arial, sans-serif;
            font-size: small;
            margin: 15px;
      }

      a:link {
            color: #0000cc;
      }

      a:active {
            color: red;
      }

      a:visited {
            color: #551a8b;
      }

      h1 {
            font-size: x-large;
            margin: 0;
            margin-bottom: 0.25em;
      }

      p {
            margin-top: 0;
            margin-bottom: 1em;
      }

      #placelist {
            width: 500px;
      }

      #search {
            margin-bottom: 5px;
      }

      #searchform {
            width : 100%;
      }
```

```
#map {
      border: 1px solid #979797;
      height: 350px;
}

#results {
      position: absolute;
      left: 540px;
}

#searchwell {
      width : 330px;
}

#searchwell .unselected {
      padding-left: 18px;
      padding-top: 1px;
      background-image:url("http://labs.google.com/ridefinder/images/mm_20_yell
      background-repeat: no-repeat;
      background-position: top left;
}

.unselected .gs-watermark {
      display: none;
}

#searchwell .select {
      margin-bottom: 1em;
}

.unselected .select {
      cursor: pointer;
      text-decoration: underline;
      color: #7777cc;
}

#selected {
      margin-top: 1em;
}

#selected .gs-result {
      margin-bottom: 1em;
}

#selected .gs-result {
      padding-left: 30px;
      padding-top: 3px;
      background-image: url("http://www.google.com/mapfiles/icon.png");
      background-repeat: no-repeat;
      background-position: top left;
}
```

```
</style>

<script
src="http://maps.google.com/maps?file=api&amp;v=2.x&amp;key=ABQIAAAAtWCOvsQE3_q3_qb8k
Y-VG3o8b6wZyHyJA" type="text/javascript"></script>

<script src="http://www.google.com/uds/api?file=uds.js&amp;v=1.0" type="text/javascri

<script type="text/javascript">

//<![CDATA[

      // Our global state
      var gLocalSearch;
      var gMap;
      var gSelectedResults = [];
      var gCurrentResults = [];
      var gSearchForm;

      // Create our "tiny" marker icon
      var gSmallIcon = new GIcon();
      gSmallIcon.image = "http://labs.google.com/ridefinder/images/mm_20_yellow.png";

      gSmallIcon.shadow = "http://labs.google.com/ridefinder/images/mm_20_shadow.png"

      gSmallIcon.iconSize = new GSize(12, 20);
      gSmallIcon.shadowSize = new GSize(22, 20);
      gSmallIcon.iconAnchor = new GPoint(6, 20);
      gSmallIcon.infoWindowAnchor = new GPoint(5, 1);

      // Set up the map and the local searcher.
      function OnLoad() {

            // Initialize the map
            gMap = new GMap(document.getElementById("map"));
            gMap.addControl(new GSmallMapControl());
            gMap.addControl(new GMapTypeControl());
            gMap.setCenter(new GLatLng(-37.91170058826019,145.1345443725586), 14);

            // Initialize the local searcher
            gLocalSearch = new GlocalSearch();
            gLocalSearch.setSearchCompleteCallback(null, OnLocalSearch);

            // Execute the initial search
      }

      function doSearch() {
            var zip = document.getElementById("zipInput").value;
            var cat = document.getElementById("catInput").value;
            gLocalSearch.setCenterPoint(zip);
            gLocalSearch.execute(cat);
      }
```

```
// Called when Local Search results are returned, we clear the old
// results and load the new ones.
function OnLocalSearch() {

      if (!gLocalSearch.results) return;
      var searchWell = document.getElementById("searchwell");

      // Clear the map and the old search well
      searchWell.innerHTML = "";
      for (var i = 0; i < gCurrentResults.length; i++) {
            if (!gCurrentResults[i].selected()) {
                  gMap.removeOverlay(gCurrentResults[i].marker());
            }
      }

      gCurrentResults = [];
      for (var i = 0; i < gLocalSearch.results.length; i++) {
      gCurrentResults.push(new LocalResult(gLocalSearch.results[i]));
      }

      var attribution = gLocalSearch.getAttribution();
      if (attribution) {
            document.getElementById("searchwell").appendChild(attribution);
      }

      // move the map to the first result
      var first = gLocalSearch.results[0];
      gMap.recenterOrPanToLatLng(new GPoint(parseFloat(first.lng), parseFloat(f

}

// Cancel the form submission, executing an AJAX Search API search.
function CaptureForm(searchForm) {
      gLocalSearch.execute(searchForm.input.value);
      return false;
}

// A class representing a single Local Search result returned by the
// Google AJAX Search API.
function LocalResult(result) {
      this.result_ = result;
      this.resultNode_ = this.unselectedHtml();
      document.getElementById("searchwell").appendChild(this.resultNode_);
      gMap.addOverlay(this.marker(gSmallIcon));
}

// Returns the GMap marker for this result, creating it with the given
// icon if it has not already been created.
LocalResult.prototype.marker = function(opt_icon) {

      if (this.marker_) return this.marker_;

      var marker = new GMarker(new GLatLng(parseFloat(this.result_.lat), parseF
```

```
        GEvent.bind(marker, "click", this, function() {
              marker.openInfoWindow(this.selected() ? this.selectedHtml() : this.
        });

        this.marker_ = marker;
        return marker;
  }

  // "Saves" this result if it has not already been saved
  LocalResult.prototype.select = function() {

        if (!this.selected()) {
              this.selected_ = true;

              // Remove the old marker and add the new marker
              gMap.removeOverlay(this.marker());
              this.marker_ = null;
              gMap.addOverlay(this.marker(G_DEFAULT_ICON));

              // Add our result to the saved set
              document.getElementById("selected").appendChild(this.selectedHtml()

              // Remove the old search result from the search well
              this.resultNode_.parentNode.removeChild(this.resultNode_);
        }
  }

  // Returns the HTML we display for a result before it has been "saved"
  LocalResult.prototype.unselectedHtml = function() {

        var container = document.createElement("div");
        container.className = "unselected";
        container.appendChild(this.result_.html.cloneNode(true));
        var saveDiv = document.createElement("div");
        saveDiv.className = "select";
        saveDiv.innerHTML = "Save this location";
        GEvent.bindDom(saveDiv, "click", this, function() {
        gMap.closeInfoWindow();
        this.select();
        gSelectedResults.push(this);

        });

  container.appendChild(saveDiv);
  return container;

  }

  // Returns the HTML we display for a result after it has been "saved"
  LocalResult.prototype.selectedHtml = function() {
        return this.result_.html.cloneNode(true);
  }
```

```
      // Returns true if this result is currently "saved"
      LocalResult.prototype.selected = function() {
            return this.selected_;
      }

      GSearch.setOnLoadCallback(OnLoad);

//]]>

</script>

</head>

<body onunload="GUnload()">

<h1>My Favorite Places</h1>

<p>Search for locations on the map below and save them to your list of favorite place

<div id="placelist">

      <div id="search">

      <div id="searchform">

            <input type="text" id="zipInput" />

            <select id="catInput">
                  <option value="restaurants">Restaurants</option>
                  <option value="bars">Bars</option>
            </select>

            <input type="button" value="Find" onclick="doSearch()"/>

      </div>

</div>

<div id="results">
      <div id="searchwell"></div>
</div>

<div id="map"></div>
      <div id="selected"></div>
</div>

</body>

</html>
```

class GIcon

An icon specifies the images used to display a GMarker on the map. For browser compatibility reasons, specifying an icon is actually quite complex. Note that you can use the default Maps icon G_DEFAULT_ICON if you don't want to specify your own.

Constructor: GIcon(copy?:GIcon, image?:String)

Description: Creates a new icon object. If another icon is given in the optional copy argument, its properties are copied, otherwise they are left empty. The optional argument image sets the value of the image property.

Example: var gSmallIcon = new GIcon();

Property: image

Type: String

Description: The foreground image URL of the icon.

Example: gSmallIcon.image = "http://labs.google.com/ridefinder/images/mm_20_yellow.png";

Property: shadow

Type: String

Description: The shadow image URL of the icon.

Example: gSmallIcon.shadow = "http://labs.google.com/ridefinder/images/mm_20_shadow.png";

Property: iconSize

Type: GSize

Description: The pixel size of the foreground image of the icon.

Example: gSmallIcon.iconSize = new GSize(12, 20);

Property: shadowSize

Type: GSize

Description: The pixel size of the shadow image.

Example: gSmallIcon.shadowSize = new GSize(22, 20);

Property: iconAnchor

Type: GPoint

Description: The pixel coordinate relative to the top left corner of the icon image at which this icon is anchored to the map.

Example: gSmallIcon.iconAnchor = new GPoint(6, 20);

Property: infoWindowAnchor

Type: GPoint

Description: The pixel coordinate relative to the top left corner of the icon image at which the info window is anchored to this icon.

Example: gSmallIcon.infoWindowAnchor = new GPoint(5, 1);

---

class GMap2

Method: addOverlay(overlay:GOverlay)

Return value: none

Description: Adds an overlay to the map and fires the addoverlay event.

Example: gMap.addOverlay(this.marker(G_DEFAULT_ICON));

Method: removeOverlay(overlay:GOverlay)

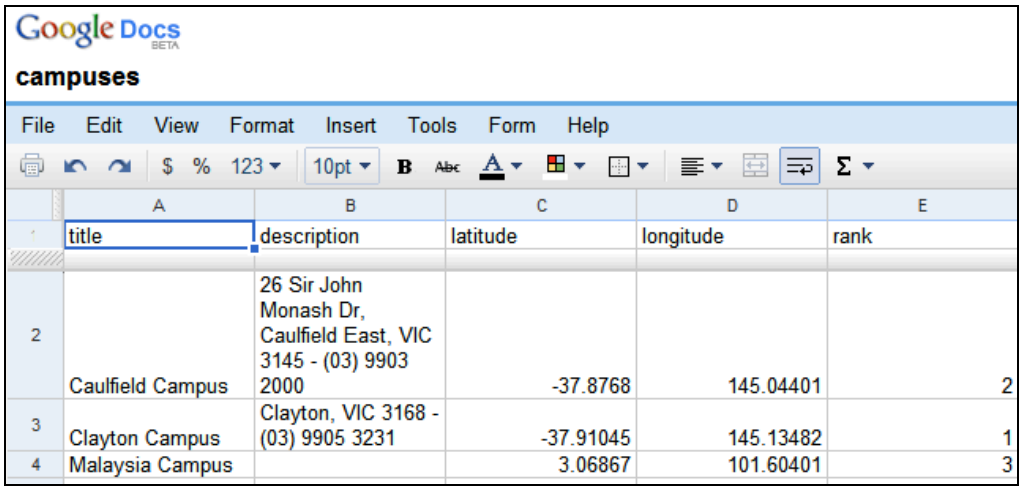
Return value: none

Description: Removes the overlay from the map. It is an error to try to remove an overlay that is not on the map. If the call is successful, it fires the removeoverlay event.

Example: gMap.removeOverlay(this.marker());

3.5. EXAMPLE 5: SPREADSHEETS MAP WIZARD

- URL: http://www.geocities.com/quocthai0512/example5.html
- Description:
  - This example illustrates how to get data from Google Spreadsheet and display it on the map.
  - Figure 5a shows my spreadsheet called campuses which stores information about three uni campuses including their addresses, latitudes, longitudes and their ranks in the search results.
  - This spreadsheet is published as ATOM file format by pressing the 'Share' button.
  - Then, we pass the spreadsheet key to a spreadsheets map wizard [2] and customize the spreadsheet fields to generate the codes.
  - Figure 5b shows the resulting map.
- Screenshots:



Google Docs BETA

campuses

File Edit View Format Insert Tools Form Help

	A	B	C	D	E
	title	description	latitude	longitude	rank
2	Caulfield Campus	26 Sir John Monash Dr, Caulfield East, VIC 3145 - (03) 9903 2000	-37.8768	145.04401	2
3	Clayton Campus	Clayton, VIC 3168 - (03) 9905 3231	-37.91045	145.13482	1
4	Malaysia Campus		3.06867	101.60401	3

Figure 5a – Google Spreadsheet

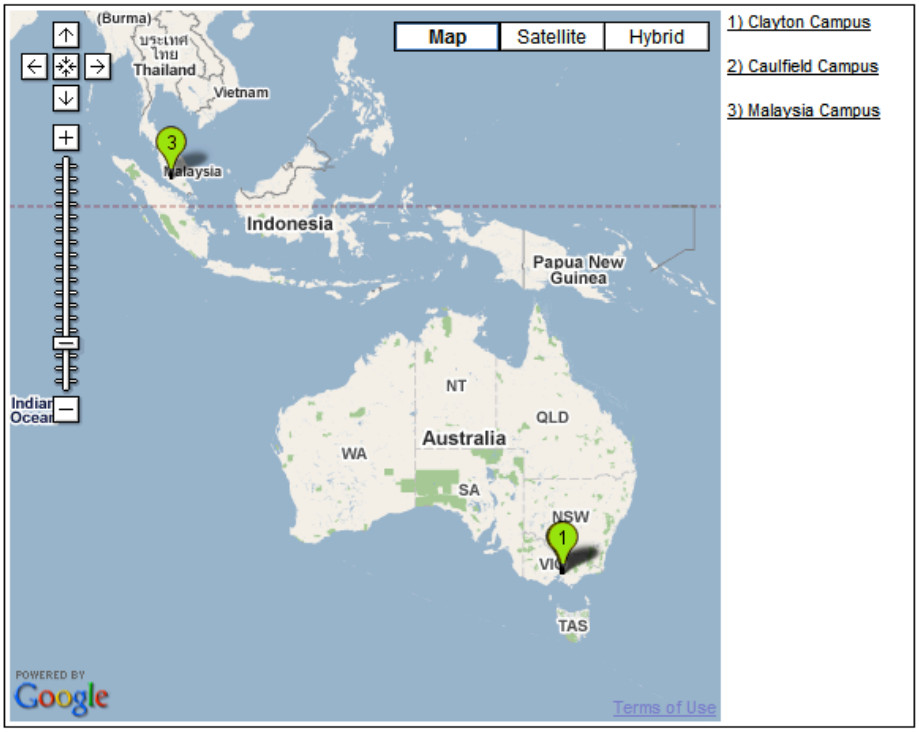


Figure 5a – Google Spreadsheet Map

EXAMPLE 5.HTML

```
<html>

<head></head>

<body>

<p>The campus data is stored in Google Spreadsheet.</p>

<div style="width:575px; font-family:Arial, sans-serif; font-size:11px; border:1px so

      <table id="cm_mapTABLE"><tbody><tr id="cm_mapTR">
            <td><div id="cm_map" style="width:450px; height:450px"></div> </td>
      </tr></tbody></table>

</div>

<script
src="http://maps.google.com/maps?file=api&amp;v=2.x&amp;key=ABQIAAAAtWCOvsQE3_q3_qb8k
Y-VG3o8b6wZyHyJA" type="text/javascript"></script>

<script type="text/javascript">

//<![CDATA[

      var cm_map;
      var cm_mapMarkers = [];
      var cm_mapHTMLS = [];

      // Create a base icon for all of our markers that specifies the
      // shadow, icon dimensions, etc.
      var cm_baseIcon = new GIcon();
      cm_baseIcon.shadow = "http://www.google.com/mapfiles/shadow50.png";
      cm_baseIcon.iconSize = new GSize(20, 34);
      cm_baseIcon.shadowSize = new GSize(37, 34);
      cm_baseIcon.iconAnchor = new GPoint(9, 34);
      cm_baseIcon.infoWindowAnchor = new GPoint(9, 2);
      cm_baseIcon.infoShadowAnchor = new GPoint(18, 25);

      // Change these parameters to customize map
      var param_wsId = "od6";
      var param_ssKey = "p-BUs7ktuNV0HkI0Uh8pGAA";
      var param_useSidebar = true;
      var param_titleColumn = "title";
      var param_descriptionColumn = "description";
      var param_latColumn = "latitude";
      var param_lngColumn = "longitude";
      var param_rankColumn = "rank";
      var param_iconType = "green";
      var param_iconOverType = "orange";
```

```
      /**
      * Loads map and calls function to load in worksheet data.
      */
      function cm_load() {
if (GBrowserIsCompatible()) {
// create the map
cm_map = new GMap2(document.getElementById("cm_map"));
cm_map.addControl(new GLargeMapControl());
cm_map.addControl(new GMapTypeControl());
cm_map.setCenter(new GLatLng( 43.907787,-79.359741), 14);
cm_getJSON();
} else {
alert("Sorry, the Google Maps API is not compatible with this browser");
            }
}

/**
      * Function called when marker on the map is clicked.
      * Opens an info window (bubble) above the marker.
      * @param {Number} markerNum Number of marker in global array
      */
function cm_markerClicked(markerNum) {
            cm_mapMarkers[markerNum].openInfoWindowHtml(cm_mapHTMLS[markerNum]);
}

/**
      * Function that sorts 2 worksheet rows from JSON feed
      * based on their rank column. Only called if column is defined.
      * @param {rowA} Object Represents row in JSON feed
      * @param {rowB} Object Represents row in JSON feed
      * @return {Number} Difference between row values
      */
function cm_sortRows(rowA, rowB) {
            var rowAValue = parseFloat(rowA["gsx$" + param_rankColumn].$t);
            var rowBValue = parseFloat(rowB["gsx$" + param_rankColumn].$t);
            return rowAValue - rowBValue;
}

/**
      * Called when JSON is loaded. Creates sidebar if param_sideBar is true.
* Sorts rows if param_rankColumn is valid column. Iterates through worksheet rows,
      * creating marker and sidebar entries for each row.
      * @param {JSON} json Worksheet feed
      */
function cm_loadMapJSON(json) {
            var usingRank = false;

            if(param_useSidebar == true) {
                  var sidebarTD = document.createElement("td");
                  sidebarTD.setAttribute("width","150");
                  sidebarTD.setAttribute("valign","top");
                  var sidebarDIV = document.createElement("div");
                  sidebarDIV.id = "cm_sidebarDIV";
```

```
                  sidebarDIV.style.overflow = "auto";
                  sidebarDIV.style.height = "450px";
                  sidebarDIV.style.fontSize = "11px";
                  sidebarDIV.style.color = "#000000";
                  sidebarTD.appendChild(sidebarDIV);
                  document.getElementById("cm_mapTR").appendChild(sidebarTD);
            }

            var bounds = new GLatLngBounds();

            if(json.feed.entry[0]["gsx$" + param_rankColumn]) {
                  usingRank = true;
                  json.feed.entry.sort(cm_sortRows);
            }

            for (var i = 0; i < json.feed.entry.length; i++) {
                  var entry = json.feed.entry[i];
                  if(entry["gsx$" + param_latColumn]) {
                        var lat = parseFloat(entry["gsx$" + param_latColumn].$t);
                        var lng = parseFloat(entry["gsx$" + param_lngColumn].$t);
                        var point = new GLatLng(lat,lng);
                        var html = "<div style='font-size:12px'>";
                        html += "<strong>" + entry["gsx$"+param_titleColumn].$t
+ "</strong>";
                        var label = entry["gsx$"+param_titleColumn].$t;
                        var rank = 0;
                        if(usingRank && entry["gsx$" + param_rankColumn]) {
                        rank = parseInt(entry["gsx$"+param_rankColumn].$t);
                        }
                        if(entry["gsx$" + param_descriptionColumn]) {
                  html += "<br/>" + entry["gsx$"+param_descriptionColumn].$t;
                        }
                        html += "</div>";

                        // create the marker
                        var marker = cm_createMarker(point,label,html,rank);
                        cm_map.addOverlay(marker);
                        cm_mapMarkers.push(marker);
                        cm_mapHTMLS.push(html);
                        bounds.extend(point);

                        if(param_useSidebar == true) {
                              var markerA = document.createElement("a");
markerA.setAttribute("href","javascript:cm_markerClicked('" + i +"')");
                              markerA.style.color = "#000000";
var sidebarText= "";
                              if(usingRank) {
                                    sidebarText += rank + ") ";
                              }
                              sidebarText += label;
markerA.appendChild(document.createTextNode(sidebarText));
sidebarDIV.appendChild(markerA);
sidebarDIV.appendChild(document.createElement("br");
```

```
sidebarDIV.appendChild(document.createElement("br");
                        }
                  }
            }//endof for loop

            cm_map.setZoom(cm_map.getBoundsZoomLevel(bounds));
            cm_map.setCenter(bounds.getCenter());

}endof cm_loadMapJSON

/**
      * Creates marker with ranked Icon or blank icon,
      * depending if rank is defined. Assigns onclick function.
      * @param {GLatLng} point Point to create marker at
      * @param {String} title Tooltip title to display for marker
      * @param {String} html HTML to display in InfoWindow
      * @param {Number} rank Number rank of marker, used in creating icon
      * @return {GMarker} Marker created
      */
function cm_createMarker(point, title, html, rank) {
var markerOpts = {};
var nIcon = new GIcon(cm_baseIcon);

if(rank > 0 && rank < 100) {
nIcon.imageOut = "http://gmaps-samples.googlecode.com/svn/trunk/" + "markers/" + para
nIcon.imageOver = "http://gmaps-samples.googlecode.com/svn/trunk/" +
        "markers/" + param_iconOverType + "/marker" + rank + ".png";
nIcon.image = nIcon.imageOut;
} else {
nIcon.imageOut = "http://gmaps-samples.googlecode.com/svn/trunk/" +
        "markers/" + param_iconType + "/blank.png";
nIcon.imageOver = "http://gmaps-samples.googlecode.com/svn/trunk/" +
        "markers/" + param_iconOverType + "/blank.png";
nIcon.image = nIcon.imageOut;
      }

            markerOpts.icon = nIcon;
            markerOpts.title = title;
            var marker = new GMarker(point, markerOpts);

            GEvent.addListener(marker, "click", function() {
                  marker.openInfoWindowHtml(html);
            });

GEvent.addListener(marker, "mouseover", function() {
                  marker.setImage(marker.getIcon().imageOver);
            });

            GEvent.addListener(marker, "mouseout", function() {
                  marker.setImage(marker.getIcon().imageOut);
            });

GEvent.addListener(marker, "infowindowopen", function() {
```

```
                  marker.setImage(marker.getIcon().imageOver);
            });

            GEvent.addListener(marker, "infowindowclose", function() {
                  marker.setImage(marker.getIcon().imageOut);
            });

            return marker;
}

/**
      * Creates a script tag in the page that loads in the
      * JSON feed for the specified key/ID.
      * Once loaded, it calls cm_loadMapJSON.
      */
function cm_getJSON() {

            // Retrieve the JSON feed.
            var script = document.createElement('script');

script.setAttribute('src', 'http://spreadsheets.google.com/feeds/list' + '/' + param_
'?alt=json-in-script&callback=cm_loadMapJSON');

script.setAttribute('id', 'jsonScript');
script.setAttribute('type', 'text/javascript');
document.documentElement.firstChild.appendChild(script);
}

setTimeout('cm_load()', 500);

//]]>

</script>

</body>

</html>
```

3.6. EXAMPLE 6: STREET VIEW ANIMATION

- URL: http://www.geocities.com/quocthai0512/example6.html
- Description:
  - Users are required to enter two locations (e.g. Melbourne, Canberra).
  - Then, we can choose various speed options from the drop-down menu.
  - When the 'Route' button is clicked, the application will display the directions on the right column from which users can choose to see different street view.
  - When users click the 'Drive' button, the street view will be moving in accordance with the directions.
- Screenshots:

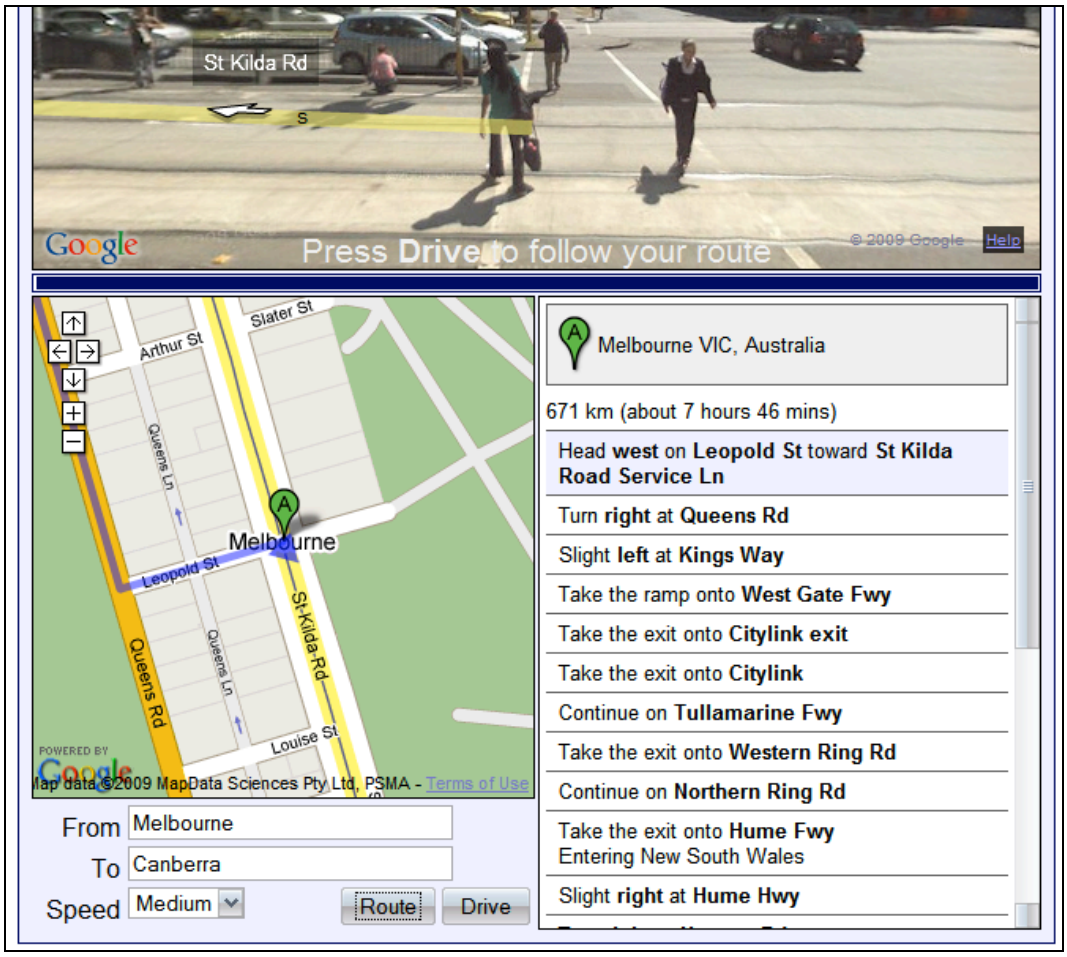


Figure 6 – Directions with Animating Street View

EXAMPLE6.HTML

```
<html>

<head>

<script
src="http://maps.google.com/maps?file=api&amp;v=2.x&amp;key=ABQIAAAAtWCOvsQE3_q3_qb8k
Y-VG3o8b6wZyHyJA" type="text/javascript"></script>

<script type="text/javascript">

      /**
      * GMap2 of the route map.
      */
      var map;

      /**
      * GStreetviewPanorama of the Flash viewer.
      */
      var pano;

      /**
      * GStreetviewClient used to retrieve meta data for panoramas we reach
      * by following links.
      */
      var svClient;

      /**
      * GDirections object used to submit driving direction requests.
      */
      var directions;

      /**
      * The GRoute we are following extracted from the directions response.
      */
      var route;

      /**
      * It turns out that the polyline generated for a driving directions
      * route normally has a lot of repeated vertices. This causes problems
      * when trying to determine how close we are to the next vertex, so it's
      * better to collapse these duplicated vertices out.
      *
      * The array of route vertices with duplicates removed.
      */
      var vertices;

      /**
      * Array that maps the polyline vertex indices to the
      * index of the same point in the vertices array.
      *
```

```
* For example, if the polyline vertices are
* [a, a, b, c, d, d, e], the vertices array will be
* [ a, b, c, d, e ] and the vertexMap array will be
* [ 1, 1, 2, 3, 4, 4, 5 ].
*/
var vertexMap;

/**
* Array that contains the index in the vertices array of
* the point at the start of the n'th step in the route
*/
var stepToVertex;

/**
* An array that gives the route step number that each
* point in the vertices array is part of.
*/
var stepMap;

/**
* The current position of the panorama and vehicle marker.
*/
var currentLatLng;

/**
* Metadata for the current panorama including the list of
* available links, loaded using GStreetviewClient.
*/
var panoMetaData;

/**
* boolean flag set when we are so close to the next vertex that we should
* check links in the panoramas we load for the next turning we need.
*/
var close = false;

/**
* The direction in degrees from our current location to the next
* vertex on the route. Used to select the most suitable link to follow.
*/
var bearing;

/**
* The direction from the next vertex on the route to the vertex
* after that. Used when we are close to a vertex and are looking
* for links that represent the next turn we need to make.
*/
var nextBearing;

/**
* The index of the vertex we are heading towards on the route in the
* vertices array.
*/
```

```
var nextVertexId;

/**
* GLatLng of the vertex we are heading towards on the route.
*/
var nextVertex;

/**
* An array that at any time contains the GLatLng of each vertex
* from the start of the current route step to the next vertex
* ahead of our current position. This is used to work out how
* far we are along the current step.
*/
var progressArray;

/**
* The distance in meters covered by traversing the points in the
* progressArray. By subtracting the distance from our current location
* to the next vertex from this value we find how far along the step
* we are, and use this to update the progress bar.
*/
var progressDistance;

/**
* Index of the route step we are currently on.
*/
var currentStep;

/**
* The marker on our map that shows the current location. For IE6 this
* is a standard red maps pushpin. For all other browsers it is a
* directional arrow.
*/
var carMarker;

/**
* A copy of the current step index used to unhighlight the previously
* highlighted step in the textual driving directions when the
* current step changes.
*/
var selectedStep = null;

/**
* boolean flag indicating whether we are currently driving (automatically
* following) links, or are stationary.
*/
var driving = false;

/**
* Id of the timer that adds a delay between following each link to give the
* panorama time to load. We need this to cancel the timer if the user clicks
* Stop while we are waiting to follow the next link.
*/
```

```
var advanceTimer = null;

/**
* Delay in seconds between following each link.
*/
var advanceDelay = 1;

/**
* Set up the initial view and register the various event listeners.
*/
function load() {
      if (GBrowserIsCompatible()) {
            var start = new GLatLng(-37.84134,144.97742);
            map = new GMap2(document.getElementById("map"));
            map.setCenter(start, 3);
            map.addControl(new GSmallMapControl());

            carMarker = getCarMarker(start);
            map.addOverlay(carMarker);
            carMarker.hide();

            svClient = new GStreetviewClient();
            pano = new GStreetviewPanorama(document.getElementById("streetview"

            GEvent.addListener(pano, "initialized", function(loc) {
                  panoMetaData = loc;
                  moveCar();
            });

            GEvent.addListener(pano, "error", function(errorCode) {
                  showStatus("The requested panorama could not be displayed");
            });

            directions = new GDirections(map);

            GEvent.addListener(directions, "load", function() {
                  jumpInMyCar();
            });

            GEvent.addListener(directions, "error", function() {
                  showStatus("Could not generate a route for the current start
            });
      }//endif
}

/**
* Create a vehicle marker suited to the user's browser
* Avoids a wierd IE6 bug that causes black backgrounds
* if we use the driving direction arrows as marker icons
* @param {GLatLng} start Initial location of the vehicle marker
* @return {GMarker}
*/
function getCarMarker(start) {
```

```
        /*@cc_on @*/
        /*@if (@_jscript_version < 5.7)
        return new GMarker(start);
        /*@end @*/
        return new GMarker(start, getArrowIcon(0.0));
    }

    /**
    * Set the vehicle marker icon based on the user's browser
    * Avoids a wierd IE6 bug that causes black backgrounds
    * if we use the driving direction arrows as marker icons
    * @param {number} bearing The heading in degrees of the arrow we need
    */
    function setCarMarkerImage(bearing) {
        /*@cc_on @*/
        /*@if (@_jscript_version < 5.7)
        return;
        /*@end @*/
        carMarker.setImage(getArrowUrl(bearing));
    }

    /**
    * Submit the driving directions request
    * The load event handler calls jumpInMyCar() when this returns
    */
    function generateRoute() {
        var from = document.getElementById("from").value;
        var to = document.getElementById("to").value;
        directions.load("from: " + from + " to: " + to, { preserveViewport: true,
    }

    /**
    * It's too far to walk on your own
    */
    function jumpInMyCar() {
        /* Extract the one and only route from this response */
        route = directions.getRoute(0);

        /* Simplify the list of polyline vertices by removing duplicates */
        collapseVertices(directions.getPolyline());

        /* Center the map on the start of the route at street level */
        map.setCenter(vertices[0], 16);

        /* Display the textual driving directions */
        renderTextDirections();

        /* Begin checking the Street View coverage along this route */
        checkCoverage(0);
    }

    /**
    * Check that a Street View panorama exists at the start
```

```
      * of this route step. This is a recursive function that
      * checks every step along the route until it reaches the
      * end of the route or no panorama is found for a step.
      * @param {number} step The route step to check
      */
      function checkCoverage(step) {
            if (step > route.getNumSteps()) {
                  /* Coverage check across whole route passed */
                  hideStatus();
                  stopDriving();
                  jumpToVertex(0);
            } else {
                  if (step == route.getNumSteps()) {
                        ll = route.getEndLatLng();
                  } else {
                        ll = route.getStep(step).getLatLng();
                  }

                  svClient.getNearestPanorama(ll, function(svData) {
                  if (svData.code == 500) {
                        /* Server error, retry once per second */
                        setTimeout("checkCoverage(" + step + ")", 1000);
                  } else if (svData.code == 600) {
                        /* Coverage check failed */
                        showStatus("Street View coverage is not available for this ro
                  } else {
                        /* Confirmed coverage for this step.
                        * Now check coverage for next step.
                        */
                        checkCoverage(step + 1);
                  }
                  });
            }
      }

      /**
      * Jump to a particular point on the route. This is used to
      * queue up the start of the route, when a user selects a step
      * in the driving directions, and when there is a gap in coverage
      * that we need to jump over.
      * @param {number} idx The vertex number in the vertices array
      */
      function jumpToVertex(idx) {

            currentLatLng = vertices[idx];
            nextVertex = vertices[idx + 1];
            nextVertexId = idx + 1;

            bearing = getBearingFromVertex(idx);
            nextBearing = getBearingFromVertex(idx + 1);

            setCarMarkerImage(bearing);
            carMarker.setLatLng(currentLatLng);
```

```
            carMarker.show();

            currentStep = stepMap[idx];
            constructProgressArray(idx);
            setProgressDistance();
            updateProgressBar(0);

            map.panTo(currentLatLng, 16);
            highlightStep(currentStep);
            checkDistanceFromNextVertex();

            pano.setLocationAndPOV(currentLatLng, { yaw:bearing, pitch:0 });
            svClient.getNearestPanorama(currentLatLng, function(loc) {
            if (loc.code == 500) {
                  setTimeout("jumpToVertex(" + idx + ")", 1000);
            } else if (loc.code == 600) {
                  jumpToVertex(nextVertexId);
            } else {
                  panoMetaData = loc.location;
                  panoMetaData.pov.yaw = bearing;
                  moveCar();
            }
            });

     }

    /**
     * Called by the panorama's initialized event handler in
     * response to a link being followed. Updates the location
     * of the vehicle marker and the center of the map to match
     * the location of the panorama loaded by following the link.
     */
     function moveCar() {
            currentLatLng = panoMetaData.latlng;
            carMarker.setLatLng(currentLatLng);
            map.panTo(currentLatLng);

            /* Now retrieve the links for this panorama so we can
            * work out where to go next.
            */
            svClient.getNearestPanorama(panoMetaData.latlng, function(svData) {
                  if (svData.code == 500) {
                        /* Server error. Retry once a second */
                        setTimeout("moveCar()", 1000);
                  } else if (svData.code == 600) {
                        /* No panorama. Should never happen as we have
                        * already loaded this panorama in the Flash viewer.
                        */
                        jumpToVertex(nextVertexId);
                  } else {
                        panoMetaData.links = svData.links;
                        checkDistanceFromNextVertex();
                        if (driving) {
```

```
                        advanceTimer = setTimeout("advance()", advanceDelay * 1000);
                        }
                  }
            });
 }

/**
 * Check if we have already passed the next vertex, or if we are
 * close enough to the next vertex to look out for the next turn.
 */
 function checkDistanceFromNextVertex() {

            close = false;
            var d = currentLatLng.distanceFrom(nextVertex);
            var b = getBearing(currentLatLng, nextVertex);

            /* If the bearing of the next vertex is more than 90 degrees away from
            * the bearing we have been travelling in, we must have passed it already.
            */
            if (getYawDelta(bearing, b) > 90) {
                  incrementVertex();

                  /* If the vertices are closely spaced we may
                  * already be close to the next vertex
                  */
                  if (driving) {
                        checkDistanceFromNextVertex();
                  }
            } else {
                  /* Recalculate how far we have travelled within the current step
                  * and update the progress bar accordingly.
                  */
                  updateProgressBar(progressDistance - d);
                  if (driving) {
                        updateViewerDirections(progressDistance - d);
                  }

                  /* If we are less than 10m from a vertex we consider ourself to
                  * be close enough to preferentially follow links that take us in
                  * the direction we should be going when the vertex has been
                  * passed.
                  */

                  if (d < 10) {
                        close = true;
                  }
            }
 }

/**
 * Move forward one link
 */
 function advance() {
```

```
            /* chose the best link for our current heading */
            var selected = selectLink(bearing);

            /* If we're very close to a vertex, also check for a
            * link in the direction we should be turning next.
            * If there is a link in that direction (to a
            * tolerance of 15 degrees), chose that turning
            */
            if (close && nextBearing) {
                  var selectedTurn = selectLink(nextBearing);
                  if (selectedTurn.delta < 15) {
                        selected = selectedTurn;
                        incrementVertex();
                  }
            }

            if (selected.delta > 40) {
                  /* If the chosen link is in a direction more than 40
                  * degrees different from the heading we want it
                  * will not take us in the right direction. As no
                  * better link has been found this implies that the
                  * route has no coverage in the direction we need so
                  * jump to the start of the next step.
                  */
                  jumpToVertex(nextVertexId);
            } else {
                  /* Pan the viewer round to face the direction of the
                  * link we want to follow and then follow the link. We
                  * need to give the pan time to complete before we follow
                  * the link for it to look smooth. The amount of time
                  * depends on the extent of the pan.
                  */
                  var panAngle = getYawDelta(panoMetaData.pov.yaw, panoMetaData.links
                  pano.panTo({ yaw:panoMetaData.links[selected.idx].yaw, pitch:0 });
                  setTimeout(function() {
                  pano.followLink(panoMetaData.links[selected.idx].yaw);
                  }, panAngle * 10);
            }
      }

      /**
      * Select which link in the current panorama most closely
      * matches the directions we should be going in.
      * @param {number} yaw The direction we are looking to move in
      * @return {Object} The number of the closest link and the
      *       difference between it's yaw and the desired direction
      */
      function selectLink(yaw) {
            var Selected = new Object();

            for (var i = 0; i < panoMetaData.links.length; i++) {

                  var d = getYawDelta(yaw, panoMetaData.links[i].yaw);
```

```
                if (Selected.delta == null || d < Selected.delta) {
                        Selected.idx = i;
                        Selected.delta = d;
                }
        }

        return Selected;
}

/**
 * Called when we have reached a vertex
 * and now need to head towards the next
 */
function incrementVertex() {
        if (! vertices[nextVertexId + 1]) {
                /* we are at the end of the route */
                endReached();
        } else {
                nextVertexId++;
                nextVertex = vertices[nextVertexId];

                /* Rotate the vehicle marler to face the new bearing */
                bearing = getBearingFromVertex(nextVertexId - 1);
                nextBearing = getBearingFromVertex(nextVertexId);
                setCarMarkerImage(bearing);

                /* Check if we have reached the next step */
                if (stepMap[nextVertexId - 1] == currentStep) {
                        /* Still on the same step so just extend the
                        * progressArray with the next vertex we are
                        * heading towards.
                        */
                        progressArray.push(nextVertex);
                } else {
                        /* We've moved on to the next step so start a new
                        * progressArray and update the text highlight and
                        * progress bar.
                        */
                        currentStep = stepMap[nextVertexId - 1];
                        highlightStep(currentStep);
                        progressArray = [ currentLatLng, nextVertex ];
                        updateProgressBar(0);
                }

                setProgressDistance();
        }
}

/**
 * Called when the last vertex on the route is reached.
 */
function endReached() {
```

```
            stopDriving();
            updateProgressBar(0);
            showInstruction("You have reached your destination");
            document.getElementById("step" + selectedStep).style.backgroundColor = "w
            selectedStep = null;
    }

    /**
     * Get the direction to head in from a particular vertex
     * @param {number} n Index of the vertex in the vertices array
     * @return {number} bearing in degrees
     */
     function getBearingFromVertex(n) {
            var origin = vertices[n];
            var destination = vertices[n+1];
            if (destination != undefined) {
                  return getBearing(origin, destination);
            } else {
                  return null;
            }
     }

    /**
     * Update the in Flash viewer next step driving directions
     * @param {number} distanceFromStartOfStep Distance in meters from the start
     *     of the current route step
     */
     function updateViewerDirections(distanceFromStartOfStep) {
            var lengthOfStep = route.getStep(currentStep).getDistance().meters;
            var distanceFromEndOfStep = (lengthOfStep - distanceFromStartOfStep);

            /* Convert from meters to feet */
            distanceFromEndOfStep *= 3.2808399;

            var uiDistance, unit;

            /* Convert to human friendly representation */
            if (distanceFromEndOfStep > 7920) {
                  distanceFromEndOfStep /= 5280;
                  uiDistance = distanceFromEndOfStep.toFixed(0);
                  unit = 'miles';
            } else if (distanceFromEndOfStep > 4620) {
                  uiDistance = '1';
                  unit = 'mile';
            } else if (distanceFromEndOfStep > 3300) {
                  /* Display "3/4 mile" between 5/8 and 7/8 of a mile */
                  uiDistance = '&frac34;';
                  unit = 'mile';
            } else if (distanceFromEndOfStep > 1980) {
                  /* Display "1/2 mile" between 3/8 and 5/8 of a mile */
                  uiDistance = '&frac12;';
                  unit = 'mile';
            } else if (distanceFromEndOfStep >  660) {
```

```
                /* Display "1/4 mile" between 1/8 and 3/8 of a mile */
                uiDistance = '&frac14;';
                unit = 'mile';
          } else {
                uiDistance = (Math.round(distanceFromEndOfStep / 10)) * 10;
                unit = "ft";
          }

          if (route.getStep(currentStep + 1) != undefined) {
                showInstruction('In ' + uiDistance + ' ' + unit + ': ' + route.getS
          } else {
                showInstruction('In ' + uiDistance + ' ' + unit + ': You will reach
          }
    }

    /**
     * Rebuild the progressArray after jumping to a particular vertex
     * @param {number} vertexId The vertex number in the vertices array
     */
     function constructProgressArray(vertexId) {
            progressArray = new Array();
            var stepStart = stepToVertex[currentStep];
            for (var i = stepToVertex[currentStep]; i <= vertexId + 1; i++) {
                  progressArray.push(vertices[i]);
            }
     }

    /**
     * Calculate the distance in meters from the start of this
     * step to the next vertex by building a polyline from the
     * intermediate points.
     */
     function setProgressDistance() {
            var polyline = new GPolyline(progressArray);
            progressDistance = polyline.getLength();
     }

    /**
     * Update the progress bar to reflect our position within this step
     * @param {number} progress Distance in meters travelled
     * since the start of this step
     */
     function updateProgressBar(progress) {
            progress = (progress < 0 ? 0 : progress);
            var stepLength = route.getStep(currentStep).getDistance().meters;
            setProgressBarLength(1 - (progress / stepLength));
     }

    /**
     * Size the progress bar
     * @param {number} progress Progress expressed as a fraction (0 to 1)
     */
     function setProgressBarLength(progress) {
```

```
            var width = (636 * progress);
            if (width < 0) {
                  width = 0;
            }
            document.getElementById("progressBar").style.width = width + "px";
     }

    /**
     * Calculate the difference in degrees between two bearings.
     * @param {number} a bearing in degrees
     * @param {number} b bearing in degrees
     * @return {number} The angle between a and b
     */
     function getYawDelta(a, b) {
            var d = Math.abs(sanitiseYaw(a) - sanitiseYaw(b));
            if (d > 180) {
                  d = 360 - d;
            }
            return d;
     }

    /**
     * Sometimes after following a link the yaw is > 360
     * @param {number} yaw bearing in degrees
     * @return {number} yaw as a value between -360 and +360
     */
     function sanitiseYaw(yaw) {
            if (yaw > 360 || yaw < 360) {
                  yaw = yaw % 360;
            }
            return yaw;
     }

    /**
     * Generate a GMarker icon of an direction arrow
     * @param {number} bearing Direction arrow should point
     * @return {GIcon}
     */
     function getArrowIcon(bearing) {
            var icon = new GIcon();
            icon.image = getArrowUrl(bearing);
            icon.iconSize = new GSize(24, 24);
            icon.iconAnchor = new GPoint(12, 12);
            return icon;
     }

    /**
     * Determine URL of correct direction arrow image to use
     * @param {number} bearing Direction arrow should point
     * @return {String}
     */
    function getArrowUrl(bearing) {
            var id = (3 * Math.round(bearing / 3)) % 120;
```

```
        return "http://maps.google.com/mapfiles/dir_" + id + ".png";
 }

/**
 * Build the vertices, vertexMap, stepToVertex, and stepMap
 * arrays from the vertices of the route polyline.
 * @param {GPolyline} path The route polyline to process
 */
function collapseVertices(path) {
        vertices = new Array();
        vertexMap = new Array(path.getVertexCount());

        vertices.push(path.getVertex(0));
        vertexMap[0] = 0;

        /* Copy vertices from the polyline to the vertices array
        * skipping any duplicates. Build the vertexMap as we go along */
        for (var i = 1; i < path.getVertexCount(); i++) {
              if (! path.getVertex(i).equals(vertices[vertices.length - 1])) {
                    vertices.push(path.getVertex(i));
              }
              vertexMap[i] = vertices.length - 1;
        }

        stepToVertex = new Array(route.getNumSteps());
        stepMap      = new Array(vertices.length);

        for (var i = 0; i < route.getNumSteps(); i++) {
              stepToVertex[i] = vertexMap[route.getStep(i).getPolylineIndex()];
        }

        var step = 0;
        for (var i = 0; i < vertices.length; i++) {
              if (stepToVertex[step + 1] == i) {
                    step++;
              }
              stepMap[i] = step;
        }
}

/**
 * Because we want to be highlight text directions steps and make them
 * clickable  we must render them ourselves rather than let GDirections do so
 * as normal.
 */
 function renderTextDirections() {

   /* Get the addresses to display at the start and end of the directions */
        var startAddress = route.getStartGeocode().address;
        var   endAddress = route.getEndGeocode().address;

   /* Write the start address title, marker, and summary */
        var html  =  getDirectionsWaypointHtml(startAddress, "A");
```

```
            html +=  getDivHtml("summary", "", route.getSummaryHtml());

      /* Build up the textual directions step by step */
            for (var n = 0; n < route.getNumSteps(); n++) {
                  html += '<a onclick="selectStep(' + n + ')">';
                  html += getDivHtml("step" + n, "dstep", route.getStep(n).getDescrip
                  html += '</a>';
            }

      /* Write the end address title and marker */
            html += getDirectionsWaypointHtml(endAddress, "B");

      /* Fill in the div on the page with the generated HTML */
            document.getElementById("directions").innerHTML = html;

      /* Set the icons used in the address blocks using the appropriate
       * technique for the browser to preserve PNG transparency */
            setWaypointIcon('A');
            setWaypointIcon('B');
 }

/**
 * Generate the HTML of the header and footer of each set of textual directions
 * @param {String} address The address of this endpoint
 * @param {String} letter The letter to display in the marker
 *      (A at the start, B at the end)

 * @return {String} the generated HTML
 */
 function getDirectionsWaypointHtml(address, letter) {
            var content = getDivHtml('letter' + letter, 'letterIcon', "");
            content += '<span class="waypointAddress">' + address + '</span>';
            return getDivHtml("wayPoint" + letter, "waypoint", content);
 }

/**
 * Set the icons used in the textual directions address blocks
 * using the appropriate technique for the browser to preserve
 * PNG transparency
 */
 function setWaypointIcon(letter) {

            var png = 'http://maps.google.com/intl/en_us/mapfiles/icon_green' + lette

            /*@cc_on @*/
            /*@if (@_jscript_version < 5.7)

            document.getElementById('letter' + letter).style.filter = 'progid:DXImage
            png + '", sizingMethod="scale");';
            return;

            /*@end @*/
```

```
            document.getElementById('letter' + letter).style.backgroundImage = 'url('

    }

      /**
      * Utility function to wrap content in a div tags
      * @param {String} id The id attribute value for this div element
      * @param {String} cssClass The class attribute value for this div element
      * @param {String} content The HTML content to wrap in this div element
      * @return {String} the HTML of the generated div element
      */
      function getDivHtml(id, cssClass, content) {

            var div = "<div";

            if (id != "") {
                  div += ' id="' + id + '"';
            }

            if (cssClass != "") {
                  div += ' class="' + cssClass + '"';
            }

            div += '>' + content + '</div>';

            return div;

      }

      /**
      * Handle a click on one of the text directions steps.
      * @param {number} i The index of the step that was clicked on
      */
      function selectStep(i) {
            var vertex = vertexMap[route.getStep(i).getPolylineIndex()];
            stopDriving();
            jumpToVertex(vertex);
      }

      /**
      * Highlight with a blue background one of the text directions steps.
      * @param {number} i The index of the step to highlight
      */
      function highlightStep(i) {
            /* Remove highlighting from the currently highlighted step */
            if (selectedStep != null) {
                  document.getElementById("step" + selectedStep).style.backgroundColo
            }

            /* Highlight the indicated step as requested */
            document.getElementById("step" + i).style.backgroundColor = "#eeeeff";
            selectedStep = i;
      }
```

```
/**
* Update the UI for driving and start following links
*/
function startDriving() {
	hideInstruction();
	document.getElementById("route").disabled = true;
	document.getElementById("stopgo").value = "Stop";
	document.getElementById("stopgo").setAttribute('onclick', 'stopDriving()'
	document.getElementById("stopgo").onclick = function() { stopDriving(); }
	driving = true;
	advance();
}

/**
* Stop following links and update the UI
*/
function stopDriving() {
	driving = false;

	if (advanceTimer != null) {
		clearTimeout(advanceTimer);
		advanceTimer = null;
	}

	document.getElementById("route").disabled = false;
	document.getElementById("stopgo").disabled = false;
	document.getElementById("stopgo").value = "Drive";
	document.getElementById("stopgo").setAttribute('onclick', 'startDriving()
	document.getElementById("stopgo").onclick = function() { startDriving();
	showInstruction('Press <b>Drive</b> to follow your route');
}

/**
* Change the speed at which driving progresses by adjusting the delay
* between following links from one panorama to the next
*/
function setSpeed() {
	advanceDelay = document.getElementById('speed').selectedIndex;
}

/**
* Display a status message that fills the Flash viewer
* @param {String} message The message to display
*/
function showStatus(message) {
	hideInstruction();
	document.getElementById("status").innerHTML = message;
	document.getElementById("status").style.display = "block";
	document.getElementById("streetview").style.display = "none";
}

/**
```

```
* Hide the currently displayed status message
*/
function hideStatus() {
      document.getElementById("status").style.display = "none";
      document.getElementById("streetview").style.display = "block";
}

/**
* Display an instructional message at the bottom of the Flash viewer
* @param {String} message The message to display
*/
function showInstruction(message) {
      document.getElementById("instruction").innerHTML = message;
      document.getElementById("instruction").style.display = "block";
}

/**
* Hide the currently displayed instructional message
*/
function hideInstruction() {
      document.getElementById("instruction").style.display = "none";
}

/* Following functions based on those provided at:
* http://www.movable-type.co.uk/scripts/latlong.html
* Copyright 2002-2008 Chris Veness
*/

/**
* Calculate the bearing in degrees between two points
* @param {number} origin      GLatLng of current location
* @param {number} destination GLatLng of destination
* @return {number}
*/
function getBearing(origin, destination) {
      if (origin.equals(destination)) {
            return null;
      }
      var lat1 = origin.lat().toRad();
      var lat2 = destination.lat().toRad();
      var dLon = (destination.lng()-origin.lng()).toRad();

      var y = Math.sin(dLon) * Math.cos(lat2);
      var x = Math.cos(lat1)*Math.sin(lat2) -
        Math.sin(lat1)*Math.cos(lat2)*Math.cos(dLon);
      return Math.atan2(y, x).toBrng();
}

/**
* Convert an angle in degrees to radians
*/
Number.prototype.toRad = function() {
      return this * Math.PI / 180;
```

```
	}

	/**
	* Convert an angle in radians to degrees (signed)
	*/
	Number.prototype.toDeg = function() {
		return this * 180 / Math.PI;
	}

	/**
	* Convert radians to degrees (as bearing: 0...360)
	*/
	Number.prototype.toBrng = function() {
		return (this.toDeg()+360) % 360;
	}
</script>
<style>

	body {
		position: relative;
	}

	#content {
		width: 640px;
		padding: 8px;
		background-color: #eeeeff;
		border: 1px solid #000066;
	}

	#svPanel {
		width: 638px;
		height: 319px;
		position: relative;
	}

	#status {
		position: absolute;
		top: 120px;
		left: 0px;
		width: 638px;
		text-align: center;
		font: 32pt sans-serif;
		color: #666666;
		background-color: white;
	}

	#instruction {
		position: absolute;
		top: 295px;
		left: 0px;
		width: 638px;
		text-align: center;
		font: 16pt sans-serif;
```

```
        color: #eeeeee;
        display: none;
    }

    #svPanel, #directions, #map {
        border: 1px solid black;
        background-color: white;
    }

    #streetview {
        position: absolute;
        top: 0px;
        left: 0px;
        width: 638px;
        height: 319px;
    }

    #progressBorder {
        position: relative;
        width: 638px;
        height: 10px;
        margin: 2px 0px 2px 0px;
        border: 1px solid #000066;
        background-color: white;
        overflow: hidden;
    }

    #progressBar {
        position: absolute;
        background-color: #000066;
        width: 636px;
        height: 8px;
        top: 1px;
        right:1px;
    }

    #map {
        width: 317px;
        height: 317px;
        margin-right: 1px;
        overflow: hidden;
    }

    #directions {
        width: 317px;
        height: 400px;
        margin-left: 1px;
        position: relative;
        overflow: auto;
    }

    .waypoint {
        position: relative;
```

```
            background-color: #eeeeee;
            border: 1px solid #666666;
            padding: 6px;
            margin: 4px;
            font: 10pt sans-serif;
      }

      .letterIcon {
            width: 24px;
            height: 38px;
            background-image: none;
      }

      .waypointAddress {
            position: absolute;
            top: 17px;
            left: 32px;
      }

      #summary {
            padding: 4px;
            font: 10pt sans-serif;
      }

      .dstep {
            border-top: 1px solid #666666;
            padding: 4px;
            padding-left: 8px;
            font: 10pt sans-serif;
            margin-left: 4px;
            margin-right: 4px;
            cursor: pointer;
            background-color: white;
      }

      .label {
            width: 52px;
            text-align: right;
            font: 12pt sans-serif;
            float: left;
            position: relative;
            top: 4px;
            margin-right: 5px;
      }

      .input {
            float: left;
            width: 252px;
            text-align: left;
      }

      .controls {
            clear: both;
```

```
            padding: 4px;
      }

      #speed {
            float: left;
      }

      #buttons {
            float: right;
      }

      table {
            border-collapse: collapse;
      }

      td {
            vertical-align: top;
      }

</style>

</head>

<body onload="load();" onunload="GUnload();">

<div id="content">

<table cellpadding="0" cellspacing="0">

<tr>

      <td colspan="2">

      <div id="svPanel">

            <div id="streetview" style="width: 638px; height: 319px;"></div>

            <div id="status">Enter your start and end addresses and click <b>Route</b

            <div id="instruction"></div>

      </div>

      <div id="progressBorder">
            <div id="progressBar"></div>
      </div>

      </td>

</tr>

<tr>
```

```
	<td>

	<div id="map"></div>

	<div class="controls">

		<div class="label">From</div>

		<div class="input"><input id="from" size="30" value="Melbourne"/></div>

	</div>

	<div class="controls">

		<div class="label">To</div>

		<div class="input"><input id="to" size="30" value="Canberra"/></div>

	</div>

	<div class="controls">

		<div class="label">Speed</div>

		<div id="actions">

			<select id="speed" onchange="setSpeed()">
				<option value="0">Fast</option>
				<option value="1" SELECTED>Medium</option>
				<option value="2">Slow</option>
			</select>

			<div id="buttons">

			<input type="button" value="Route" id="route" onclick="generateRout

			<input type="button" value="Drive" id="stopgo"  onclick="startDrivi

			</div>

		</div>

	</div>

	</td>

	<td>
		<div id="directions"></div>
	</td>

</tr>

</table>
```

```
</div>
</body>
</html>
```

class GLatLng

GLatLng is a point in geographical coordinates longitude and latitude.

Notice that although usual map projections associate longitude with the x-coordinate of the map, and latitude with the y-coordinate, the latitude cooridnate is always written first, followed by the longitude, as it is custom in cartography.

Notice also that you cannot modify the coordinates of a GLatLng. If you want to compute another point, you have to create a new one.

Constructor: GLatLng(lat:Number, lng:Number, unbounded?:Boolean)

Description: Notice the ordering of latitude and longitude. If the unbounded flag is true, then the numbers will be used as passed, otherwise latitude will be clamped to lie between -90 degrees and +90 degrees, and longitude will be wrapped to lie between -180 degrees and +180 degrees.

Example: var start = new GLatLng(-37.84134,144.97742);

class GMap2

Method: setCenter(center:GLatLng, zoom?:Number, type?:GMapType)

Return value: none

Description: Sets the map view to the given center. Optionally, also sets zoom level and map type. The map type must be known to the map. See the constructor, and the method addMapType(). This method must be called first after construction to set the initial state of the map. It is an error to call operations on a newly constructed GMap2 object until after this function is invoked.

Example: map.setCenter(start, 3);

---

class GControl

These implementations of interface GControl are available.

Constructor: GSmallMapControl()

Description: Creates a control with buttons to pan in four directions, and zoom in and zoom out.

Example: map.addControl(new GSmallMapControl());

---

class GStreetviewClient

A GStreetviewClient object performs searches for Street View data based on parameters that are passed to its methods.

Constructor: GStreetviewClient()

Description: Creates a new GStreetviewClient

Example: svClient = new GStreetviewClient();

---

class GMarker

A GMarker marks a position on the map. It implements the GOverlay interface and thus is added to the map using the GMap2.addOverlay() method.

A marker object has a latlng, which is the geographical position where the marker is anchored on the map, and an icon. If the icon is not set in the constructor, the default icon G_DEFAULT_ICON is used.

After it is added to a map, the info window of that map can be opened through the marker. The marker object will fire mouse events and infowindow events.

Constructor: GMarker(latlng:GLatLng, opts?:GMarkerOptions)

Description: Creates a marker at the latlng with options specified in GMarkerOptions. By default markers are clickable & have the default icon G_DEFAULT_ICON.

Example: new GMarker(start, getArrowIcon(0.0));

3.7. EXAMPLE 7: REVERSE GEOCODING MAP

- URL: http://www.geocities.com/quocthai0512/example7.html
- Description:
  - This is an example of a reverse geocoding map.
  - Users enter latitude and longitude to display the location corresponding to those values.
  - A marker will be added to denote the query result which is set center to the map. Also, a window box is used to display the latlng values, address and country of the resulting location. This window can be turned of by the users.
  - Example 8 is similar to this example except that a map control is added and users can click on any place of the map to check the latlng values.
- Screenshots:

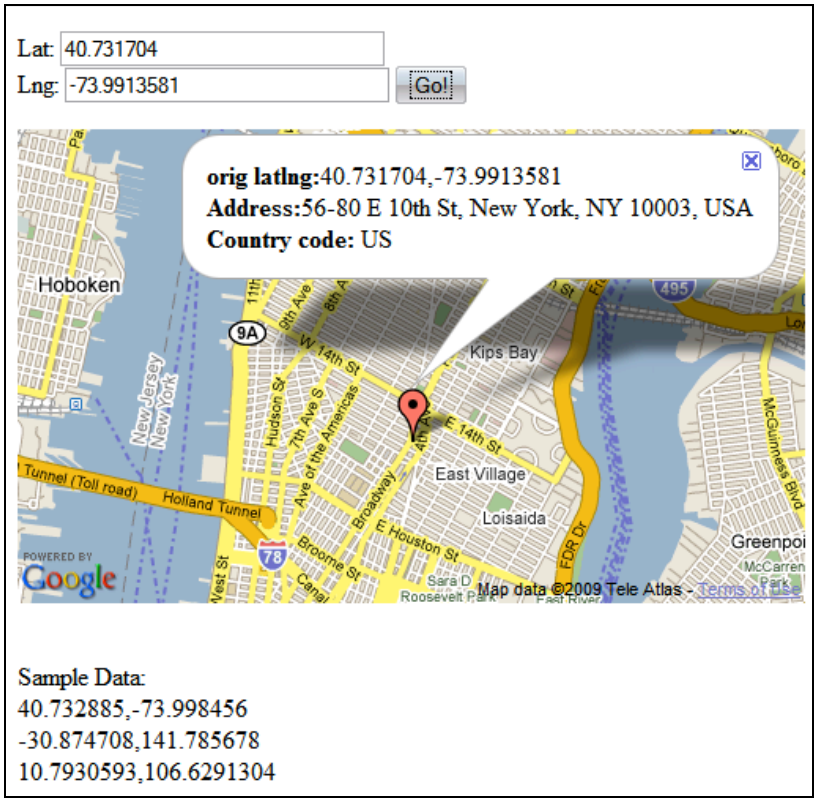


Figure 7 – Reverse Geocoding Google Map

EXAMPLE7.HTML

```
<html>

<head>

<script
src="http://maps.google.com/maps?file=api&amp;v=2.x&amp;key=ABQIAAAAtWCOvsQE3_q3_qb8k
Y-VG3o8b6wZyHyJA" type="text/javascript"></script>

<script type="text/javascript">

      var map = null;
      var geocoder = null;

      function initialize() {
            if (GBrowserIsCompatible()) {
                  map = new GMap2(document.getElementById("map_canvas"));
                  map.setCenter(new GLatLng(37.4419, -122.1419), 13);
                  geocoder = new GClientGeocoder();

            }
      }

      function getAddress(lat, lng) {
            latlng=lat+","+lng;
            if (latlng != null) {
                  address = latlng;
                  geocoder.getLocations(latlng, showAddress);
            }
      }

      function showAddress(response) {
            map.clearOverlays();
            if (!response || response.Status.code != 200) {
                  alert("Not Found.");
            } else {
                  place = response.Placemark[0];
                  point = new GLatLng(place.Point.coordinates[1], place.Point.coordin
                  map.setCenter(point, 13);
                  var marker = new GMarker(point);
                  map.addOverlay(marker);
                  marker.openInfoWindowHtml(
                  '<b>orig latlng:</b>' + response.name + '<br/>' +
                  '<b>Address:</b>' + place.address + '<br>' +
                  '<b>Country code:</b> ' + place.AddressDetails.Country.CountryNameC
            }
      }

</script>
</head>
```

```
<body onload="initialize()" onunload="GUnload()">

<form action="#" onsubmit="getAddress(this.lat.value, this.lng.value); return false">

<p>Lat: <input type="text" size="30" name="lat" value="40.731704" /><br />Lng: <input
73.9913581" /><input type="submit" value="Go!" /></p>

<div id="map_canvas" style="width: 500px; height: 300px"></div>

</form>

<br />Sample Data:
<br />40.732885,-73.998456
<br />-30.874708,141.785678
<br />10.7930593,106.6291304

</body>

</html>
```

class GClientGeocoder

This class is used to communicate directly with Google servers to obtain geocodes for user specified addresses. In addition, a geocoder maintains its own cache of addresses, which allows repeated queries to be answered without a round trip to the server. As a general best practice, it is not recommended to use GClientGeocoder functions in a loop. Developers that have multiple addresses to geocode should probably use our HTTP Geocoder instead.

Constructor: GClientGeocoder(cache?:GGeocodeCache)

Description: Creates a new instance of a geocoder that talks directly to Google servers. The optional cache parameter allows one to specify a custom client-side cache of known addresses. If none is specified, a GFactualGeocodeCache is used.

Example: geocoder = new GClientGeocoder();

Method: getLocations(address:String, callback:function)

Return Value: none

Description: Sends a request to Google servers to geocode the specified address. If the address was successfully located, the user-specified callback function is invoked with a GLatLng point. Otherwise, the callback function is given a null point. In case of ambiguous addresses, only the point for the best match is passed to the callback function.

Example: geocoder.getLocations(latlng, showAddress);

class GMap2

Method: clearOverlays()

Return value: none

Description: Removes all overlays from the map, and fires the clearoverlays event.

Example: map.clearOverlays();

---

class GLatLng

GLatLng is a point in geographical coordinates longitude and latitude.

Notice that although usual map projections associate longitude with the x-coordinate of the map, and latitude with the y-coordinate, the latitude cooridnate is always written first, followed by the longitude, as it is custom in cartography.

Notice also that you cannot modify the coordinates of a GLatLng. If you want to compute another point, you have to create a new one.

Constructor: GLatLng(lat:Number, lng:Number, unbounded?:Boolean)

Description: Notice the ordering of latitude and longitude. If the unbounded flag is true, then the numbers will be used as passed, otherwise latitude will be clamped to lie between -90 degrees and +90 degrees, and longitude will be wrapped to lie between -180 degrees and +180 degrees.

Example: point = new GLatLng(place.Point.coordinates[1], place.Point.coordinates[0]);

---

class GMap2

Method: setCenter(center:GLatLng, zoom?:Number, type?:GMapType)

Return value: none

Description: Sets the map view to the given center. Optionally, also sets zoom level and map type. The map type must be known to the map. See the constructor, and the method addMapType(). This method must be called first after construction to set the initial state of the map. It is an error to call operations on a newly constructed GMap2 object until after this function is invoked.

Example: map.setCenter(point, 13);

---

class GMarker

A GMarker marks a position on the map. It implements the GOverlay interface and thus is added to the map using the GMap2.addOverlay() method.

A marker object has a latlng, which is the geographical position where the marker is anchored on the map, and an icon. If the icon is not set in the constructor, the default icon G_DEFAULT_ICON is used.

After it is added to a map, the info window of that map can be opened through the marker. The marker object will fire mouse events and infowindow events.

Constructor: GMarker(latlng:GLatLng, opts?:GMarkerOptions)

Description: Creates a marker at the latlng with options specified in GMarkerOptions. By default markers are clickable & have the default icon G_DEFAULT_ICON.

Example: var marker = new GMarker(point);

---

class GMap2

Method: addOverlay(overlay:GOverlay)

Return value: none

Description: Adds an overlay to the map and fires the addoverlay event.

Example: map.addOverlay(marker);

---

interface GOverlay

This interface is implemented by the GMarker, GPolyline, GTileLayerOverlay and GInfoWindow classes in the maps API library. You can implement it if you want to display custom types of overlay objects on the map. An instance of GOverlay can be put on the map with the method GMap2.addOverlay(). The map will then call the method GOverlay.initialize() on the overlay instance to display itself on the map initially. Whenever the map display changes, the map will call GOverlay.redraw() so that the overlay can reposition itself if necessary. The overlay instance can use the method GMap2.getPane() to get hold of one or more DOM container elements to attach itself to.

Constructor: GOverlay()

Description: This constructor creates dummy implementations for the methods. Still, when inheriting from this class, your derived class constructor should call this constructor for completeness.

---

class GMarker

Method: openInfoWindowHtml(content:String, opts?:GInfoWindowOptions)

Return value: none

Description: Opens the map info window over the icon of the marker. The content of the info window is given as a DOM node. Only option GInfoWindowOptions.maxWidth is applicable.

Example:

marker.openInfoWindowHtml('orig latlng:' + response.name + '
' +

'Address:' + place.address + '
' +

'Country code: ' + place.AddressDetails.Country.CountryNameCode);

3.8. EXAMPLE 8: MODIFIED REVERSE GEOCODING MAP

- URL: http://www.geocities.com/quocthai0512/example8.html
- Description:
  - Example 8 is similar to this example except that a map control is added and users can click on any place of the map to check the latlng values.
  - A function called clearOverlays() is called to clean up the markers.
- Screenshots:

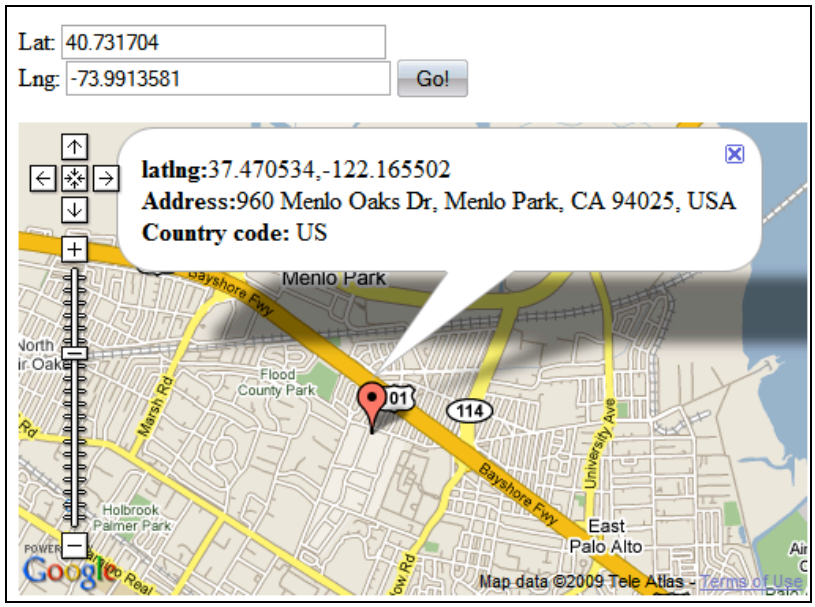


Figure 8 – Another Reverse Geocoding Google Map

EXAMPLE8.HTML

```
<html>

<head>

<script
src="http://maps.google.com/maps?file=api&amp;v=2.x&amp;key=ABQIAAAAtWCOvsQE3_q3_qb8k
Y-VG3o8b6wZyHyJA" type="text/javascript"></script>

<script type="text/javascript">

      var map = null;
      var geocoder = null;
      var address;

      function initialize() {
            if (GBrowserIsCompatible()) {
                  map = new GMap2(document.getElementById("map_canvas"));
                  map.setCenter(new GLatLng(37.4419, -122.1419), 13);
                  map.addControl(new GLargeMapControl);
                  GEvent.addListener(map, "click", getAddress2);
                  geocoder = new GClientGeocoder();
            }
      }

      function getAddress2(overlay, latlng) {
            if (latlng != null) {
                  address = latlng;
                  geocoder.getLocations(latlng, showAddress2);
            }
      }

      function showAddress2(response) {
            map.clearOverlays();
            if (!response || response.Status.code != 200) {
                  alert("Not Found.");
            } else {
                  place = response.Placemark[0];
                  point = new GLatLng(place.Point.coordinates[1],
                  place.Point.coordinates[0]);
                  marker = new GMarker(point);
                  map.addOverlay(marker);
                  marker.openInfoWindowHtml(
                  '<b>latlng:</b>' + place.Point.coordinates[1] + "," + place.Point.c
                  '<b>Address:</b>' + place.address + '<br>' +
                  '<b>Country code:</b> ' + place.AddressDetails.Country.CountryNameC
            }
      }


      function getAddress(lat, lng) {
```

```
            latlng=lat+","+lng;
            if (latlng != null) {
                  address = latlng;
                  geocoder.getLocations(latlng, showAddress);
            }
      }

      function showAddress(response) {
            map.clearOverlays();
            if (!response || response.Status.code != 200) {
                  alert("Not Found.");
            } else {
                  place = response.Placemark[0];
                  point = new GLatLng(place.Point.coordinates[1], place.Point.coordin
                  map.setCenter(point, 13);
                  var marker = new GMarker(point);
                  map.addOverlay(marker);
                  marker.openInfoWindowHtml(
                  '<b>latlng:</b>' + response.name + '<br/>' +
                  '<b>Address:</b>' + place.address + '<br>' +
                  '<b>Country code:</b> ' + place.AddressDetails.Country.CountryNameC
            }
      }

</script>

</head>

<body onload="initialize()" onunload="GUnload()">

<form action="#" onsubmit="getAddress(this.lat.value, this.lng.value); return false">

<p>Lat: <input type="text" size="30" name="lat" value="40.731704" /><br />Lng: <input
73.9913581" /><input type="submit" value="Go!" /></p>

<div id="map_canvas" style="width: 500px; height: 300px"></div>

</form>

</body>

</html>
```

class GMap2

Constructor: GLargeMapControl()

Description: Creates a control with buttons to pan in four directions, and zoom in and zoom out, and a zoom slider.

Example: map.addControl(new GLargeMapControl);

class GClientGeocoder

This class is used to communicate directly with Google servers to obtain geocodes for user specified addresses. In addition, a geocoder maintains its own cache of addresses, which allows repeated queries to be answered without a round trip to the server. As a general best practice, it is not recommended to use GClientGeocoder functions in a loop. Developers that have multiple addresses to geocode should probably use our HTTP Geocoder instead.

Constructor: GClientGeocoder(cache?:GGeocodeCache)

Description: Creates a new instance of a geocoder that talks directly to Google servers. The optional cache parameter allows one to specify a custom client-side cache of known addresses. If none is specified, a GFactualGeocodeCache is used.

Example: geocoder = new GClientGeocoder();

Method: getLocations(address:String, callback:function)

Return Value: none

Description: Sends a request to Google servers to geocode the specified address. If the address was successfully located, the user-specified callback function is invoked with a GLatLng point. Otherwise, the callback function is given a null point. In case of ambiguous addresses, only the point for the best match is passed to the callback function.

Example: geocoder.getLocations(latlng, showAddress);

---

class GMap2

Method: clearOverlays()

Return value: none

Description: Removes all overlays from the map, and fires the clearoverlays event.

Example: map.clearOverlays();

class GLatLng

GLatLng is a point in geographical coordinates longitude and latitude.

Notice that although usual map projections associate longitude with the x-coordinate of the map, and latitude with the y-coordinate, the latitude cooridnate is always written first, followed by the longitude, as it is custom in cartography.

Notice also that you cannot modify the coordinates of a GLatLng. If you want to compute another point, you have to create a new one.

Constructor: GLatLng(lat:Number, lng:Number, unbounded?:Boolean)

Description: Notice the ordering of latitude and longitude. If the unbounded flag is true, then the numbers will be used as passed, otherwise latitude will be clamped to lie between -90 degrees and +90 degrees, and longitude will be wrapped to lie between -180 degrees and +180 degrees.

Example: point = new GLatLng(place.Point.coordinates[1], place.Point.coordinates[0]);

---

class GMap2

Method: setCenter(center:GLatLng, zoom?:Number, type?:GMapType)

Return value: none

Description: Sets the map view to the given center. Optionally, also sets zoom level and map type. The map type must be known to the map. See the constructor, and the method addMapType(). This method must be called first after construction to set the initial state of the map. It is an error to call operations on a newly constructed GMap2 object until after this function is invoked.

Example: map.setCenter(point, 13);

---

class GMarker

A GMarker marks a position on the map. It implements the GOverlay interface and thus is added to the map using the GMap2.addOverlay() method.

A marker object has a latlng, which is the geographical position where the marker is anchored on the map, and an icon. If the icon is not set in the constructor, the default icon G_DEFAULT_ICON is used.

After it is added to a map, the info window of that map can be opened through the marker. The marker object will fire mouse events and infowindow events.

Constructor: GMarker(latlng:GLatLng, opts?:GMarkerOptions)

Description: Creates a marker at the latlng with options specified in GMarkerOptions. By default markers are clickable & have the default icon G_DEFAULT_ICON.

Example: var marker = new GMarker(point);

class GMap2

Method: addOverlay(overlay:GOverlay)

Return value: none

Description: Adds an overlay to the map and fires the addoverlay event.

Example: map.addOverlay(marker);

interface GOverlay

This interface is implemented by the GMarker, GPolyline, GTileLayerOverlay and GInfoWindow classes in the maps API library. You can implement it if you want to display custom types of overlay objects on the map. An instance of GOverlay can be put on the map with the method GMap2.addOverlay(). The map will then call the method GOverlay.initialize() on the overlay instance to display itself on the map initially. Whenever the map display changes, the map will call GOverlay.redraw() so that the overlay can reposition itself if necessary. The overlay instance can use the method GMap2.getPane() to get hold of one or more DOM container elements to attach itself to.

Constructor: GOverlay()

Description: This constructor creates dummy implementations for the methods. Still, when inheriting from this class, your derived class constructor should call this constructor for completeness.

---

class GMarker

Method: openInfoWindowHtml(content:String, opts?:GInfoWindowOptions)

Return value: none

Description: Opens the map info window over the icon of the marker. The content of the info window is given as a DOM node. Only option GInfoWindowOptions.maxWidth is applicable.

Example:

marker.openInfoWindowHtml('orig latlng:' + response.name + '
' +

'Address:' + place.address + '
' +

'Country code: ' + place.AddressDetails.Country.CountryNameCode);

3.9. EXAMPLE 9: ANOTHER FINDING NEAREST BARS/RESTAURANTS

- URL: http://www.geocities.com/quocthai0512/example9.html
- Description:
  - Example 9 is an enhancement of example 4 in which a marker that denotes the address to be searched has been added.
- Screenshots:

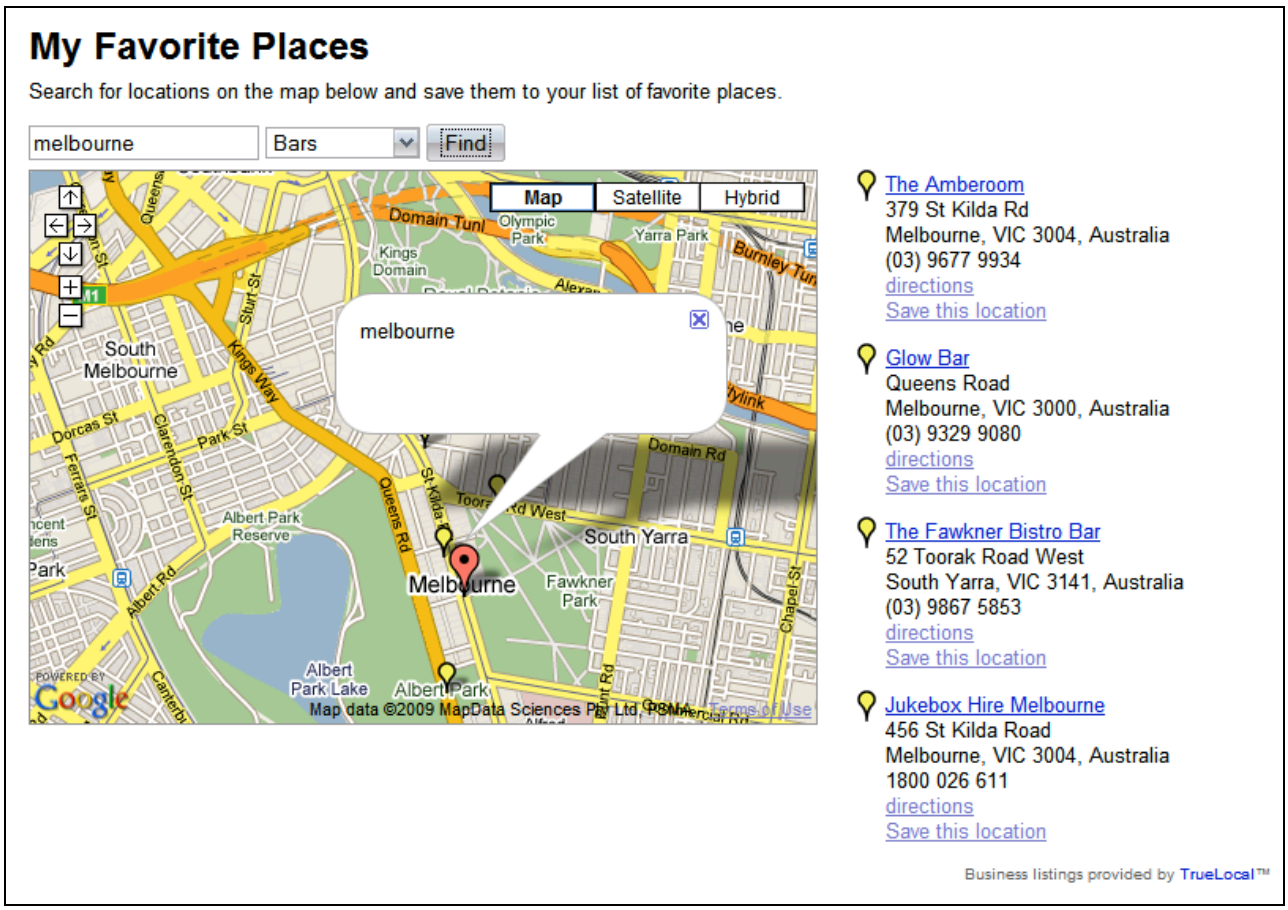


Figure 9 – Another Finding Nearest Bars/Restaurants Example

EXAMPLE9.HTML

```
<html>

<head>

<link href="http://www.google.com/uds/css/gsearch.css" rel="stylesheet" type="text/cs

<style type="text/css">

      body {
            background-color: white;
            color: black;
            font-family: Arial, sans-serif;
            font-size: small;
            margin: 15px;
      }

      a:link {
            color: #0000cc;
      }

      a:active {
            color: red;
      }

      a:visited {
            color: #551a8b;
      }

      h1 {
            font-size: x-large;
            margin: 0;
            margin-bottom: 0.25em;
      }

      p {
            margin-top: 0;
            margin-bottom: 1em;
      }

      #placelist {
            width: 500px;
      }

      #search {
            margin-bottom: 5px;
      }

      #searchform {
            width : 100%;
      }
```

```
#map {
      border: 1px solid #979797;
      height: 350px;
}

#results {
      position: absolute;
      left: 540px;
}

#searchwell {
      width : 330px;
}

#searchwell .unselected {
      padding-left: 18px;
      padding-top: 1px;
      background-image: url("http://labs.google.com/ridefinder/images/mm_20_yel
      background-repeat: no-repeat;
      background-position: top left;
}

.unselected .gs-watermark {
      display: none;
}

#searchwell .select {
      margin-bottom: 1em;
}

.unselected .select {
      cursor: pointer;
      text-decoration: underline;
      color: #7777cc;
}

#selected {
      margin-top: 1em;
}

#selected .gs-result {
      margin-bottom: 1em;
}

#selected .gs-result {
      padding-left: 30px;
      padding-top: 3px;
      background-image: url("http://www.google.com/mapfiles/icon.png");
      background-repeat: no-repeat;
      background-position: top left;
}
```

```
</style>

<script
src="http://maps.google.com/maps?file=api&amp;v=2.x&amp;key=ABQIAAAAtWCOvsQE3_q3_qb8k
Y-VG3o8b6wZyHyJA" type="text/javascript"></script>

<script src="http://www.google.com/uds/api?file=uds.js&amp;v=1.0" type="text/javascri

<script type="text/javascript">
//<![CDATA[

      // Our global state
      var gLocalSearch;
      var gMap;
      var gSelectedResults = [];
      var gCurrentResults = [];
      var gSearchForm;

      // Create our "tiny" marker icon
      var gSmallIcon = new GIcon();
      gSmallIcon.image = "http://labs.google.com/ridefinder/images/mm_20_yellow.png";
      gSmallIcon.shadow = "http://labs.google.com/ridefinder/images/mm_20_shadow.png"
      gSmallIcon.iconSize = new GSize(12, 20);
      gSmallIcon.shadowSize = new GSize(22, 20);
      gSmallIcon.iconAnchor = new GPoint(6, 20);
      gSmallIcon.infoWindowAnchor = new GPoint(5, 1);

      // Set up the map and the local searcher.
      function OnLoad() {

            // Initialize the map
            gMap = new GMap(document.getElementById("map"));
            gMap.addControl(new GSmallMapControl());
            gMap.addControl(new GMapTypeControl());
            gMap.setCenter(new GLatLng(-37.91170058826019,145.1345443725586), 14);
            geocoder = new GClientGeocoder();

            // Initialize the local searcher
            gLocalSearch = new GlocalSearch();
            gLocalSearch.setSearchCompleteCallback(null, OnLocalSearch);

            // Execute the initial search
      }

      function doSearch() {
            var zip = document.getElementById("zipInput").value;
            var cat = document.getElementById("catInput").value;

            gLocalSearch.setCenterPoint(zip);

            gMap.clearOverlays();

            if (geocoder) {
```

```
                geocoder.getLatLng(zip, function(point) {
                      if (!point) {
                            alert(zip + " not found");
                      } else {
                            var marker = new GMarker(point);
                            gMap.addOverlay(marker);
                            marker.openInfoWindowHtml(zip);
                      }
                });
          }

          gLocalSearch.execute(cat);
    }

    // Called when Local Search results are returned, we clear the old
    // results and load the new ones.
    function OnLocalSearch() {
          if (!gLocalSearch.results) return;
          var searchWell = document.getElementById("searchwell");

          // Clear the map and the old search well
          searchWell.innerHTML = "";
          for (var i = 0; i < gCurrentResults.length; i++) {
                if (!gCurrentResults[i].selected()) {
                      gMap.removeOverlay(gCurrentResults[i].marker());
                }
          }

          gCurrentResults = [];
          for (var i = 0; i < gLocalSearch.results.length; i++) {
                gCurrentResults.push(new LocalResult(gLocalSearch.results[i]));
          }

          var attribution = gLocalSearch.getAttribution();
          if (attribution) {
                document.getElementById("searchwell").appendChild(attribution);
          }

          // move the map to the first result
          var first = gLocalSearch.results[0];
          gMap.recenterOrPanToLatLng(new GPoint(parseFloat(first.lng), parseFloat(f
    }

    // Cancel the form submission, executing an AJAX Search API search.
    function CaptureForm(searchForm) {
          gLocalSearch.execute(searchForm.input.value);
          return false;
    }

    // A class representing a single Local Search result returned by the
    // Google AJAX Search API.
    function LocalResult(result) {
          this.result_ = result;
```

```
        this.resultNode_ = this.unselectedHtml();
        document.getElementById("searchwell").appendChild(this.resultNode_);
        gMap.addOverlay(this.marker(gSmallIcon));
}

// Returns the GMap marker for this result, creating it with the given
// icon if it has not already been created.
LocalResult.prototype.marker = function(opt_icon) {
        if (this.marker_) return this.marker_;
        var marker = new GMarker(new GLatLng(parseFloat(this.result_.lat), parseF
        GEvent.bind(marker, "click", this, function() {
                marker.openInfoWindow(this.selected() ? this.selectedHtml() : this.
        });
        this.marker_ = marker;
        return marker;
}

// "Saves" this result if it has not already been saved
LocalResult.prototype.select = function() {
        if (!this.selected()) {
                this.selected_ = true;

                // Remove the old marker and add the new marker
                gMap.removeOverlay(this.marker());
                this.marker_ = null;
                gMap.addOverlay(this.marker(G_DEFAULT_ICON));

                // Add our result to the saved set
                document.getElementById("selected").appendChild(this.selectedHtml()

                // Remove the old search result from the search well
                this.resultNode_.parentNode.removeChild(this.resultNode_);
        }
}

// Returns the HTML we display for a result before it has been "saved"
LocalResult.prototype.unselectedHtml = function() {
        var container = document.createElement("div");
        container.className = "unselected";
        container.appendChild(this.result_.html.cloneNode(true));
        var saveDiv = document.createElement("div");
        saveDiv.className = "select";
        saveDiv.innerHTML = "Save this location";
        GEvent.bindDom(saveDiv, "click", this, function() {
                gMap.closeInfoWindow();
                this.select();
                gSelectedResults.push(this);
        });
        container.appendChild(saveDiv);
        return container;
}

// Returns the HTML we display for a result after it has been "saved"
```

```
      LocalResult.prototype.selectedHtml = function() {
            return this.result_.html.cloneNode(true);
      }

      // Returns true if this result is currently "saved"
      LocalResult.prototype.selected = function() {
            return this.selected_;
      }

      GSearch.setOnLoadCallback(OnLoad);

//]]>

</script>

</head>

<body onunload="GUnload()">

<h1>My Favorite Places</h1>

<p>Search for locations on the map below and save them to your list of favorite place

<div id="placelist">
      <div id="search">
            <div id="searchform">
                  <input type="text" id="zipInput" />
                  <select id="catInput">
                        <option value="restaurants">Restaurants</option>
                        <option value="bars">Bars</option>
                  </select>
                  <input type="button" value="Find" onclick="doSearch()"/>
            </div>
      </div>
      <div id="results">
            <div id="searchwell"></div>
      </div>
      <div id="map"></div>
      <div id="selected"></div>
</div>
</body>
</html>
```

class GIcon

An icon specifies the images used to display a GMarker on the map. For browser compatibility reasons, specifying an icon is actually quite complex. Note that you can use the default Maps icon G_DEFAULT_ICON if you don't want to specify your own.

Constructor: GIcon(copy?:GIcon, image?:String)

Description: Creates a new icon object. If another icon is given in the optional copy argument, its properties are copied, otherwise they are left empty. The optional argument image sets the value of the image property.

Example: var gSmallIcon = new GIcon();

Property: image

Type: String

Description: The foreground image URL of the icon.

Example: gSmallIcon.image = "http://labs.google.com/ridefinder/images/mm_20_yellow.png";

Property: shadow

Type: String

Description: The shadow image URL of the icon.

Example: gSmallIcon.shadow = "http://labs.google.com/ridefinder/images/mm_20_shadow.png";

Property: iconSize

Type: GSize

Description: The pixel size of the foreground image of the icon.

Example: gSmallIcon.iconSize = new GSize(12, 20);

Property: shadowSize

Type: GSize

Description: The pixel size of the shadow image.

Example: gSmallIcon.shadowSize = new GSize(22, 20);

Property: iconAnchor

Type: GPoint

Description: The pixel coordinate relative to the top left corner of the icon image at which this icon is anchored to the map.

Example: gSmallIcon.iconAnchor = new GPoint(6, 20);

Property: infoWindowAnchor

Type: GPoint

Description: The pixel coordinate relative to the top left corner of the icon image at which the info window is anchored to this icon.

Example: gSmallIcon.infoWindowAnchor = new GPoint(5, 1);

---

class GMap2

Method: addOverlay(overlay:GOverlay)

Return value: none

Description: Adds an overlay to the map and fires the addoverlay event.

Example: gMap.addOverlay(this.marker(G_DEFAULT_ICON));

Method: removeOverlay(overlay:GOverlay)

Return value: none

Description: Removes the overlay from the map. It is an error to try to remove an overlay that is not on the map. If the call is successful, it fires the removeoverlay event.

Example: gMap.removeOverlay(this.marker());

---

class GMarker

A GMarker marks a position on the map. It implements the GOverlay interface and thus is added to the map using the GMap2.addOverlay() method.

A marker object has a latlng, which is the geographical position where the marker is anchored on the map, and an icon. If the icon is not set in the constructor, the default icon G_DEFAULT_ICON is used.

After it is added to a map, the info window of that map can be opened through the marker. The marker object will fire mouse events and infowindow events.

Constructor: GMarker(latlng:GLatLng, opts?:GMarkerOptions)

Description: Creates a marker at the latlng with options specified in GMarkerOptions. By default markers are clickable & have the default icon G_DEFAULT_ICON.

Example: var marker = new GMarker(point);

class GMap2

Method: addOverlay(overlay:GOverlay)

Return value: none

Description: Adds an overlay to the map and fires the addoverlay event.

Example: gMap.addOverlay(marker);

interface GOverlay

This interface is implemented by the GMarker, GPolyline, GTileLayerOverlay and GInfoWindow classes in the maps API library. You can implement it if you want to display custom types of overlay objects on the map. An instance of GOverlay can be put on the map with the method GMap2.addOverlay(). The map will then call the method GOverlay.initialize() on the overlay instance to display itself on the map initially. Whenever the map display changes, the map will call GOverlay.redraw() so that the overlay can reposition itself if necessary. The overlay instance can use the method GMap2.getPane() to get hold of one or more DOM container elements to attach itself to.

Constructor: GOverlay()

Description: This constructor creates dummy implementations for the methods. Still, when inheriting from this class, your derived class constructor should call this constructor for completeness.

---

class GMarker

Method: openInfoWindowHtml(content:String, opts?:GInfoWindowOptions)

Return value: none

Description: Opens the map info window over the icon of the marker. The content of the info window is given as a DOM node. Only option GInfoWindowOptions.maxWidth is applicable.

Example:

```
marker.openInfoWindowHtml(zip);
```

3.10. EXAMPLE 10: ADDING MARKERS

- URL: http://www.geocities.com/quocthai0512/example10.html
- Description:
  - Users are enabled to select two types of marker and drag and drop them onto the map.
  - These markers are draggable.
  - To remove a marker, users can simple drag it off the map.
  - Users can click the “Clear All” button to remove all the markers at once.
- Screenshots:

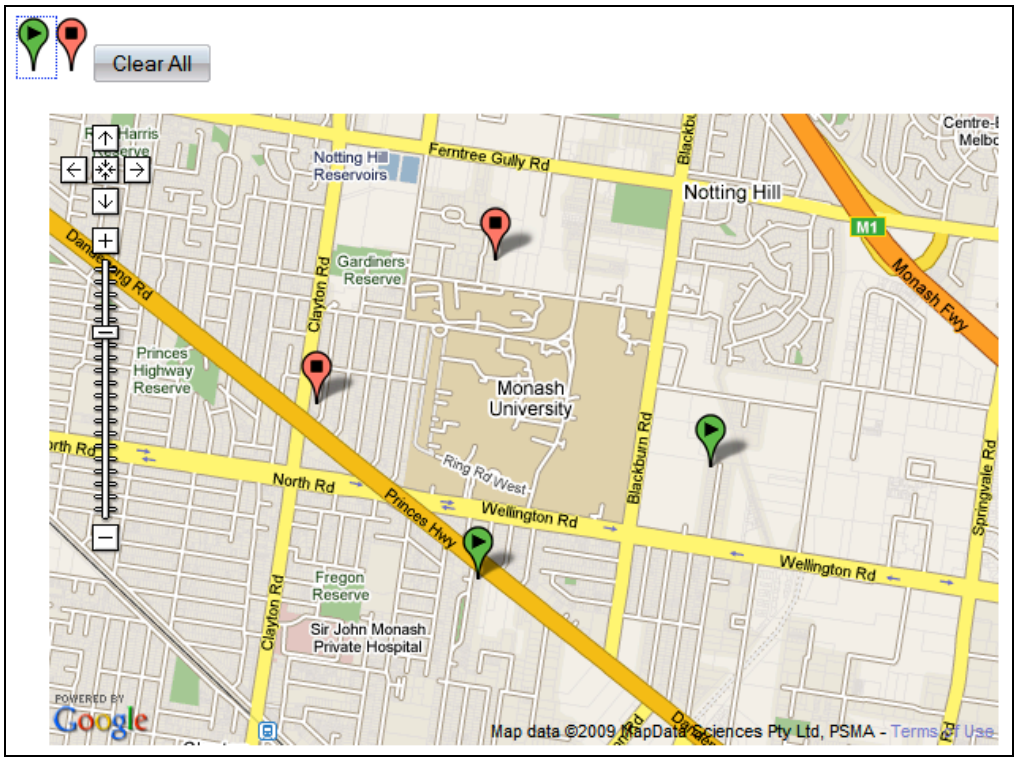


Figure 10 – Adding Markers Example

EXAMPLE10.HTML

```
<html>

<head>

<style type="text/css">

      #map{
            width:600px;
            height:400px;
            margin: 20;
      }

      a {
            text-decoration: none;
      }

      .pushpin{
            width:20px;
            height:34px;
            border:none;
      }

</style>

<script
src="http://maps.google.com/maps?file=api&v=2&key=ABQIAAAAtWCOvsQE3_q3_qb8kEaDNhQxMJd
Y-VG3o8b6wZyHyJA" type="text/javascript">
</script>

</head>

<body onunload="GUnload()" >

<a class="marker" href="javascript:follow(0)"><img src="http://maps.google.com/mapfil
title="click me" class="pushpin"/></a>

<a class="marker" href="javascript:follow(1)"><img src="http://maps.google.com/mapfil
me" class="pushpin"/></a>

<input type="button" onClick="clearMarkers();" value="Clear All">

<div id="map"><noscript>You should turn on JavaScript</noscript></div>

<script type="text/javascript" language="JavaScript">

      var map = new GMap2(document.getElementById("map"));
      var start = new GLatLng(-37.91170058826019,145.1345443725586);
      map.addControl(new GLargeMapControl);
      map.setCenter(start, 14);
```

```
	var images = [
		"http://maps.google.com/mapfiles/dd-start.png",
		"http://maps.google.com/mapfiles/dd-end.png"
	];

	function follow(imageInd){
		var marker;
		var dog = true;
		var noMore = false;

		var mouseMove = GEvent.addListener(map, 'mousemove', function(cursorPoint
			if(!noMore){
				marker = new GMarker(cursorPoint,{draggable:true, autoPan:fal
				map.addOverlay(marker);
				marker.setImage(images[imageInd]);
				noMore = true;
				// This function deletes the marker when dragged outside map
				GEvent.addListener(marker, 'drag', function(markerPoint){
				if(!map.getBounds().containsLatLng(markerPoint)){
					map.removeOverlay(marker);
				}
				});
			}
			if(dog){
				marker.setLatLng(cursorPoint);
			}
			});

			var mapClick = GEvent.addListener(map, 'click', function(){

				dog = false;

				// 'mousemove' event listener is deleted to save resources
				GEvent.removeListener(mouseMove);
				GEvent.removeListener(mapClick);

			});
	}

	function clearMarkers(){

		map.clearOverlays();

	}
</script>
</body>
</html>
```

class GMarker

A GMarker marks a position on the map. It implements the GOverlay interface and thus is added to the map using the GMap2.addOverlay() method.

A marker object has a latlng, which is the geographical position where the marker is anchored on the map, and an icon. If the icon is not set in the constructor, the default icon G_DEFAULT_ICON is used.

After it is added to a map, the info window of that map can be opened through the marker. The marker object will fire mouse events and infowindow events.

Constructor: GMarker(latlng:GLatLng, opts?:GMarkerOptions)

Description: Creates a marker at the latlng with options specified in GMarkerOptions. By default markers are clickable & have the default icon G_DEFAULT_ICON.

---

class GLatLng

GLatLng is a point in geographical coordinates longitude and latitude.

Notice that although usual map projections associate longitude with the x-coordinate of the map, and latitude with the y-coordinate, the latitude cooridnate is always written first, followed by the longitude, as it is custom in cartography.

Notice also that you cannot modify the coordinates of a GLatLng. If you want to compute another point, you have to create a new one.

Constructor: GLatLng(lat:Number, lng:Number, unbounded?:Boolean)

Description: Notice the ordering of latitude and longitude. If the unbounded flag is true, then the numbers will be used as passed, otherwise latitude will be clamped to lie between -90 degrees and +90 degrees, and longitude will be wrapped to lie between -180 degrees and +180 degrees.

---

class GMarkerOptions

Instances of this class are used in the opts? argument to the constructor of the GMarker class. There is no constructor for this class. Instead, this class is instantiated as a javascript object literal.

Property: draggable

Type: Boolean

Description: Toggles whether or not the marker will be draggable by users. Markers set up to be dragged require more resources to set up than markers that are clickable. Any marker that is draggable is also clickable, bouncy and auto-pan enabled by default. The default value for this option is false.

Property: autoPan

Type: Boolean

Description: Auto-pan the map as you drag the marker near the edge. If the marker is draggable the default value for this option is true.

Example: marker = new GMarker(cursorPoint,{draggable:true, autoPan:false});

---

class GMap2

Method: addOverlay(overlay:GOverlay)

Return value: none

Description: Adds an overlay to the map and fires the addoverlay event.

Example: map.addOverlay(marker);

---

class GMarker

Method: setImage(url:String)

Return value: none

Description: Requests the image specified by the url to be set as the foreground image for this marker. Note that neither the print image nor the shadow image are adjusted. Therefore this method is primarily intended to implement highlighting or dimming effects, rather than drastic changes in marker's appearances.

Example: marker.setImage(images[imageInd]);

---

namespace GEvent

This namespace contains functions that you use to register event handlers, both for custom events and for DOM events, and to fire custom events. All the events defined by this API are custom events that are internally fired by GEvent.trigger().

Method: GEvent.addListener(source:Object, event:String, handler:Function)

Return value: GEventListener

Description: Registers an event handler for a custom event on the source object. Returns a handle that can be used to eventually deregister the handler. The event handler will be called with this set to the source object.

---

class GEventListener

This class is opaque. It has no methods and no constructor. Its instances are returned from GEvent.addListener() or GEvent.addDomListener() and are eventually passed back to GEvent.removeListener().

Example: GEvent.addListener(marker, 'drag', function(markerPoint){

if(!map.getBounds().containsLatLng(markerPoint)){

map.removeOverlay(marker);

}

});

---

class GMap2

Method: getBounds()

Return value: GLatLngBounds

Description: Returns the the visible rectangular region of the map view in geographical coordinates.

---

class GLatLngBounds

A GLatLngBounds instance represents a rectangle in geographical coordinates, including one that crosses the 180 degrees meridian.

Method: containsLatLng(latlng:GLatLng)

Return value: boolean

Description: Returns true iff the geographical coordinates of the point lie within this rectangle.

4. REFERENCES

[1] *Google Maps API* Retrieved February 15, 2009, from http://code.google.com/apis/maps/

[2] *Spreadsheets Map Wizard* Retrieved Februrary 15, 2009, from http://gmaps-samples.googlecode.com/svn/trunk/spreadsheetsmapwizard/makecustommap.htm

[3] *Sign Up for the Google Maps API* Retrieved Februrary 15, 2009 from http://code.google.com/apis/maps/signup.html