Programación SIG en entornos web

Albert Gavarró Rodríguez

PID_00174757

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

En el módulo anterior, hemos aprendido a extender un SIG de escritorio (gvSIG) para adaptarlo a nuestras necesidades. También hemos aprendido a agregar nuevas capas por código y elementos georeferenciados en ellas.

En este bloque aprenderemos a hacer prácticamente lo mismo pero en entornos web. Estudiaremos cómo construir una página web y cómo incorporarle un mapa. Aprenderemos también a agregar elementos georeferenciados al mapa que nos servirán para marcar puntos, trazar caminos o delimitar áreas.

Aunque hay varios servicios de mapas que operan en Internet, como Yahoo Maps o Microsoft Virtual Earth, este módulo se centrará en el estudio de Google Maps. Sin embargo, las técnicas que se explicarán serán fácilmente trasladables a los otros servicios.

Objetivos

Al final de este módulo, deberíamos ser capaces de:

1. Crear páginas web simples en las que se muestre un mapa.
2. Mostrar un tipo de mapa u otro dependiendo de nuestras necesidades.
3. Agregar marcadores georreferenciados a un mapa para señalar lugares.
4. Dibujar líneas y polígonos georreferenciados en un mapa para mostrar rutas o delimitar áreas.
5. Convertir las coordenadas de un mapa a coordenadas geográficas y viceversa.

1. El lenguaje HTML

1.1. La Web y el HTML

La World Wide Web, que se abrevia WWW y se conoce también como la Web, es un sistema de páginas enlazadas mediante el concepto de hipertexto. El hipertexto es un texto que refiere a otro texto al que el lector puede tener acceso inmediato. El lenguaje tradicional y predominante sobre el que se sustenta la Web y que tiene capacidades de hipertexto es el HTML (*HyperText Markup Language*, 'lenguaje de marcas de hipertexto').

Hasta hace relativamente poco era necesario conocer el lenguaje HTML para poder publicar una página web en Internet. Si bien el HTML es un lenguaje muy directo, en el que buena parte del código es el contenido propiamente dicho del documento, como contrapartida es engorroso de escribir. Así, por ejemplo, para mostrar una palabra en negrita, es necesario intercalar en el texto más de media docena de caracteres extras.

Hoy en día han aparecido muchísimas herramientas que permiten generar páginas web de la misma forma que editamos cualquier otro documento. De hecho, casi todos los procesadores de textos modernos ofrecen la posibilidad de guardar sus documentos en formato HTML. Además, hay herramientas altamente especializadas que generan páginas de altísima calidad en poco tiempo y con el mínimo esfuerzo, como por ejemplo Amaya*, Bluefish** o Dreamweaver***.

* http://www.w3.org/Amaya

** http://bluefish.openoffice.nl/

*** http://www.adobe.com/products/dreamweaver

Dada la existencia de estas herramientas de edición y puesto que el objetivo de estos materiales no es profundizar en el conocimiento del HTML, sólo daremos unas pocas pinceladas sobre este lenguaje, las necesarias para escribir páginas relativamente sencillas y para comprender los ejemplos que se trabajarán en el resto del módulo. Sin embargo, la UOC dispone de materiales abiertos que profundizan en el lenguaje HTML*.

* http://mosaic.uoc.edu/2009/11/20/introduccion-a-la-creacion-de-paginas-web/

1.2. Elementos y etiquetas

El lenguaje HTML se utiliza para describir la estructura y el contenido de un documento, así como para complementar el texto con objetos (por ejemplo, mapas) e imágenes.

Un típico código HTML está formado por elementos que configuran las distintas partes del documento. Generalmente, estos elementos están delimitados por una etiqueta de inicio y otra de fin, que se escriben siempre entre corchetes angulares ("<" y ">").

Por ejemplo, la línea siguiente:

```
<p>La WWW nació en el CERN.</p>
```

define un elemento (párrafo) cuyo contenido es "La WWW nació en el CERN.". Como podemos ver, las etiquetas de inicio y fin (y respectivamente) delimitan el contenido del párrafo. Además, la etiqueta de fin es igual que la de inicio pero precedida por una barra ("/").

Hay elementos que sólo tienen una etiqueta de inicio, pues carecen de contenido textual. Es el caso, por ejemplo, de las imágenes. Las líneas siguientes*:

```
El primer logotipo de la WWW:
<img src="http://upload.wikimedia.org/wikipedia/ ↵
commons/thumb/b/b2/WWW_logo_by_Robert_Cailliau.svg/ ↵
120px-WWW_logo_by_Robert_Cailliau.svg.png">
```

*** El símbolo ↵ indica la ruptura de una línea de texto por motivos de formato. En consecuencia, las líneas separadas por este símbolo deberían considerarse como una de sola.**

muestran la imagen que se encuentra en la dirección: http://upload.wikimedia.org/wikipedia/commons/thumb/b/b2/WWW_logo_by_Robert_Cailliau.svg/120px-WWW_logo_by_Robert_Cailliau.svg.png. El resultado será el siguiente:

El primer logotipo de la WWW: 

Observad la estructura que aparece dentro de la etiqueta de inicio:

```
src="http://upload.wikimedia.org/wikipedia/commons ↵
thumb/b/b2/WWW_logo_by_Robert_Cailliau.svg/120px ↵
WWW_logo_by_Robert_Cailliau.svg.png"
```

Todos los elementos HTML tienen atributos que varían de un elemento a otro según sus características. Dichos atributos permiten variar la forma en la que se presenta el elemento. En el caso de las imágenes, tienen un atributo "src" (de *source*, 'fuente') que indica la dirección web en la que se encuentra la imagen. Como se puede apreciar en el ejemplo, el valor de los atributos siempre especifíca en la etiqueta de inicio.

Una última característica importante de los elementos HTML es que se pueden imbricar. Es decir, en su contenido un elemento puede contener otros elementos que, a su vez, pueden contener otros, y así sucesivamente, formando una jerarquía de elementos.

El código siguiente:

```
<p>WWW significa <em>World Wide Web</em>.</p>
```

define un párrafo que contiene una texto enfatizado (*World Wide Web*). Es importante observar que los elementos se cierran en orden inverso a aquel en el que se han abierto. El resultado será el siguiente:

WWW significa *World Wide Web*.

1.3. Estructura mínima de un documento HTML

Un documento HTML bien formado debe presentar siempre la siguiente estructura mínima:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="es">
<head>
<title>Título del documento</title>
</head>
<body>
Contenido del documento.
</body>
</html>
```

Como podemos observar, la estructura mínima de un documento HTML está formada por cinco elementos:

- **DOCTYPE.** Especifica la versión del lenguaje HTML que obedece el documento (en el ejemplo, la 4.01).

- **HTML.** Es el elemento raíz, el que define el documento HTML en sí. Contiene obligatoriamente los elementos *head* y *body*. Como puede apreciarse en el ejemplo, el elemento *html* puede contener el atributo *lang*, que define el idioma del documento (“es” de español).

- **HEAD.** Define la cabecera del documento, cuyo cometido es alojar información sobre el mismo. El título, las palabras clave y otros datos que no se consideran parte del contenido del documento irán dentro de este elemento.

- **TITLE.** Define el título del documento.

- **BODY.** Define el contenido del documento. Este elemento contendrá otros que definirán el contenido del documento.

1.4. Definición del contenido

El contenido de un documento HTML puede definirse mediante tres tipos de elementos:

- **Estructurales.** Son aquellos que describen la estructura del texto. En HTML tenemos elementos estructurales para definir títulos, párrafos, tablas, enumeraciones, etc. Normalmente, los navegadores aplican diferentes estilos de presentación a cada elemento.

- **De presentación.** Son aquellos que describen el aspecto del texto, independientemente de su función. Los elementos que definen negritas, cursivas, texto enfatizado, etc. son elementos de presentación.

- **De hipertexto.** Son aquellos que permiten enlazar con otros documentos mediante la creación de hipervínculos. Son los más importantes dada la naturaleza hipertextual de la Web.

A continuación, se mostrarán algunos de los elementos más significativos de cada tipo. Además, ilustraremos su uso con ejemplos.

1.4.1. Elementos estructurales

Como ya se ha mencionado, los elementos estructurales son aquellos que describen el propósito del texto. Entre ellos podemos distinguir los siguientes:

- ***P,*** de *paragraph* (párrafo). Define un párrafo de texto.

Las líneas siguientes:

```
<p>La WWW es un sistema de páginas enlazadas mediante el
concepto de hipertexto.</p>
<p>La WWW nació en 1989 en el CERN, de la mano de Tim
Berners-Lee.</p>
```

definen un par de párrafos. El resultado será el siguiente:

La WWW es un sistema de páginas enlazadas mediante el concepto de hipertexto.

La WWW nació en 1989 en el CERN, de la mano de Tim Berners-Lee.

Como podemos observar, al final de cada párrafo se agrega automáticamente un salto de línea.

- ***H1 ... H6,*** de *heading* (título). Definen hasta seis niveles de títulos (desde el más importante —*h1*— hasta el menos importante —*h6*. Por ejemplo, las líneas siguientes:

```
<h1>Introducción</h1>
<p>La WWW es un sistema de páginas enlazadas mediante el
concepto de hipertexto.</p>
<h2>Historia</h2>
<p>La WWW nació en 1989 en el CERN, de la mano de Tim
Berners-Lee.</p>
```

definen un título de primer nivel (“Introducción”) y otro de segundo nivel (“La Web y el HTML”). Después de cada título, aparece algo de texto en forma de párrafo. El resultado será el siguiente:

Introducción

La WWW es un sistema de páginas enlazadas mediante el concepto de hipertexto.

La Web y el HTML

La WWW nació en 1989 en el CERN, de la mano de Tim Berners-Lee.

- ***DIV,*** de *divider* (divisor). Este elemento define un bloque que puede contener otros elementos. Es muy parecido al párrafo, aunque está vacío de significado. El elemento *div* es muy importante porque nos permite agrupar elementos sin caracterizarlos. Las líneas siguientes:

```
<div align="center">
<p><img src="http://upload.wikimedia.org/wikipedia/
commons/thumb/b/b2/WWW_logo_by_Robert_Cailliau.svg/
120px-WWW_logo_by_Robert_Cailliau.svg.png"></p>
<p>El primer logotipo del WWW</p>
</div>
```

crean un bloque que contiene dos párrafos y lo centran en medio del documento mediante el atributo "align" (de *alignment* —alineación—). Consecuentemente, los párrafos también quedarán alineados en el centro. El resultado será el siguiente:

El primer logotipo de la WWW

- ***BR,*** de *line break* (salto de línea). Permite emplazar un salto de línea, por ejemplo, en medio de un párrafo. Este elemento no tiene etiqueta de fin. El código siguiente:

```
<p>En la página <br>http://www.w3c.org<br> se puede
obtener información sobre el WWW Consortium.</p>
```

escribirá:

En la página:
http://www.w3c.org
se puede obtener información sobre el WWW Consortium.

Observad que, a diferencia de lo que sucede en el párrafo, el salto de línea no modifica el espaciado entre líneas.

1.4.2. Elementos de presentación

Como ya hemos mencionado, los elementos de presentación son aquellos que describen el aspecto del texto, independientemente de su función. Entre ellos destacan los siguientes:

- ***EM***, de *emphasis* (énfasis). Este elemento enfatiza el texto. Normalmente, el texto se mostrará en cursiva. Las líneas siguientes:

```
<p>La <em>World Wide Web</em> es un sistema de páginas
enlazadas mediante el concepto de hipertexto.</p>
```

se traducirán en:

> La *World Wide Web* es un sistema de páginas enlazadas mediante el concepto de hipertexto.

- ***STRONG***, de *strong emphasis* (énfasis fuerte). Este elemento es parecido al anterior, pero pone un énfasis aún mayor. Normalmente, el texto se mostrará en negrita. Las líneas siguientes:

```
<p>La <strong>World Wide Web</strong> es un sistema de
páginas enlazadas mediante el concepto de hipertexto.</p>
```

se traducirán en:

> La **World Wide Web** es un sistema de páginas enlazadas mediante el concepto de hipertexto.

1.4.3. Elementos de hipertexto

El elemento principal que permite navegar de un documento a otro es ***A***, de *anchor* (ancla). Dicho elemento permite definir un texto que, cuando se activa, nos lleva a otro documento. Por ejemplo, las líneas siguientes:

```
<p>La WWW nació en 1989 en el CERN, de la mano de
<a href="http://es.wikipedia.org/wiki/
Tim_Berners-Lee">Tim Berners-Lee</a>.</p>
```

crean un enlace entre el nombre del inventor de la WWW y el artículo correspondiente de la Wikipedia. Como se puede observar, el elemento *ancla* tiene un atributo *href* (de *hypertext reference*, 'referencia de hipertexto'), que establece el documento con el que está enlazado el texto. El resultado será el siguiente:

La WWW nació en 1989 en el CERN, de la mano de Tim Berners-Lee.

Al activar el enlace, ya sea haciendo clic sobre él o por otros medios, iremos a la página http://es.wikipedia.org/wiki/Tim_Berners-Lee.

2. Google Maps

Google Maps es la herramienta SIG en línea de Google. Permite explorar mapas en 2D de casi cualquier parte del mundo. La popularidad de la que goza Google Maps en la actualidad se debe principalmente a su buen rendimiento, su vasta cartografía y la posibilidad de incorporar toda esta tecnología a cualquier sitio web mediante el uso de la API* (de *Application Programming Interface*, 'interfaz de programa de aplicación') de Google Maps.

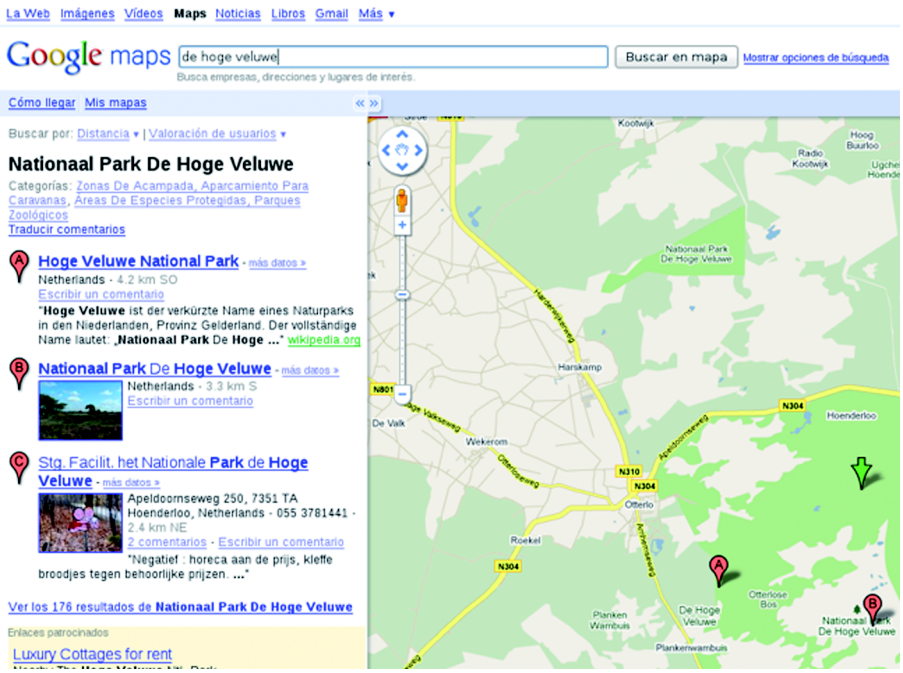
*** La documentación completa de la API se puede encontrar en:**
http://code.google.com/intl/ca/apis/maps/documentation/reference.html

En los apartados siguientes, veremos cómo incorporar un mapa a nuestra página y cómo hacer uso de la API para trabajar sobre él.

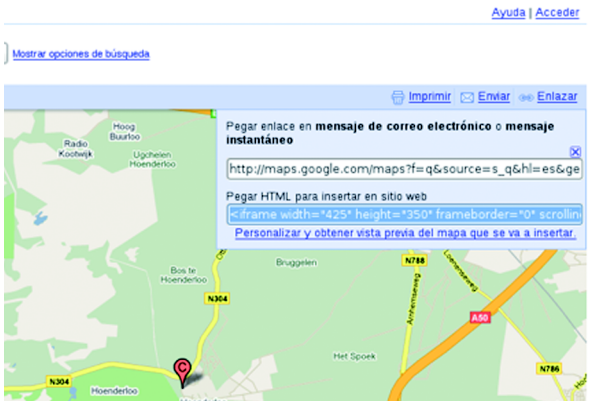
2.1. Incorporar un mapa en una página

La forma más sencilla de incorporar un mapa en nuestra página es utilizando el web de Google Maps y, una vez seleccionada la ubicación que deseamos mostrar, incorporar el código HTML que nos proporciona Google a nuestra página.

Veámoslo paso a paso. En primer lugar, debemos seleccionar una ubicación en maps.google.es. Esto se puede hacer mediante la caja de búsqueda o navegando por el mapa.



En segundo lugar, hemos de seleccionar la opción *Enlazar* y copiar el código HTML que aparece en la caja "Pegar HTML para insertar en sitio web".

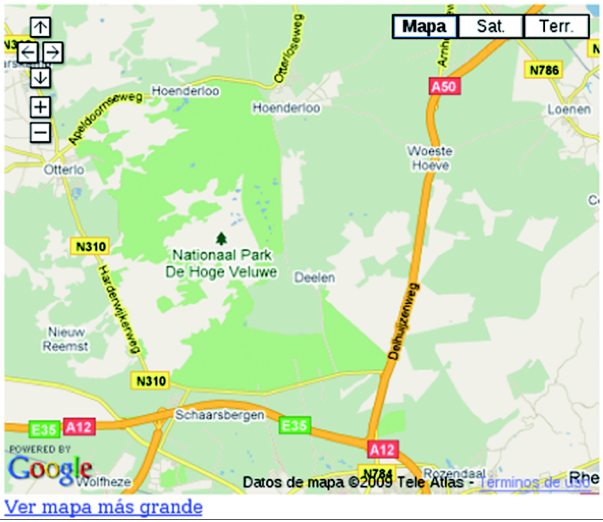


El tercer y último paso será incorporar este código en nuestra página HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Nationaal Park De Hoge Veluwe</title>
</head>
<body>
<p>El <em>Nationaal Park De Hoge Veluwe</em> (el Parque
Nacional <em>De Hoge Veluwe</em> es un parque nacional
neerlandés situado en la provincia de Gelderland, cerca
de las ciudades de Ede, Arnhem y Apeldoorn.</p>
<iframe width="425" height="350" frameborder="0"
scrolling="no" marginheight="0" marginwidth="0"
src="http://maps.google.com/?hl=es&amp;ie=UTF8 ↵
&amp;ll=52.075286,5.889359&amp;spn=0.261235,0.698318 ↵
&amp;z=11&amp;output=embed">
</iframe><br />
<small><a href="http://maps.google.com/?hl=es ↵
&amp;ie=UTF8&amp;ll=52.075286,5.889359 ↵
&amp;spn=0.261235,0.698318&amp;z=11&amp;source=embed"
style="color:#0000FF;text-align:left">Ver mapa más
grande</a></small>
</body>
</html>
```

El resultado será el siguiente:

El *Nationaal Park De Hoge Veluwe* (el Parque Nacional *De Hoge Veluwe* es un parque nacional neerlandés situado en la provincia de Gelderland, cerca de las ciudades de Ede, Arnhem y Apeldoorn.



Ver mapa más grande

Si bien el mapa que obtenemos es interactivo (podemos hacer *zoom* y desplazarnos por él) y algo personalizable (podemos cambiar mínimamente el aspecto de nuestro mapa mediante la opción *Personalizar y obtener vista previa del mapa que se va a insertar*), las posibilidades de interacción que nos ofrece son muy limitadas. Por ejemplo, no se pueden añadir o quitar capas, y no se puede asociar información a determinados puntos del mapa. Sin embargo, si nuestra única intención es situar un lugar en el mapa, ésta es una solución fácil y rápida.

2.2. La API de Google Maps

Afortunadamente, Google Maps nos ofrece muchas más posibilidades que las que hemos visto hasta ahora. Sin embargo, éstas se obtienen a costa de la facilidad de uso, pues se requieren unos mínimos conocimientos de programación para poder sacarles partido.

A esta funcionalidad adicional se accede por medio de la API de Google Maps, mediante un conjunto de tecnologías agrupadas bajo la denominación AJAX (*Asynchronous JAvascript and XML*, 'JavaScript asíncrono y XML'). Como su nombre indica, AJAX está compuesto por dos tecnologías:

- **JavaScript.** Es un lenguaje muy parecido a Java que se usa para alterar documentos HTML de forma dinámica. Es importante remarcar que no es Java; sólo está basado en él.
- **XML.** Es un lenguaje usado para el intercambio de datos (sigla de *eXtensible Markup Language*, 'lenguaje de marcas extensible'). Es una forma genérica de escribir documentos maximizando su simplicidad, su generalidad y su usabilidad.

Ambas tecnologías, integradas en un documento HTML, permiten obtener mapas y datos de los servidores de Google Maps y mostrarlos en la página.

2.2.1. Crear un mapa

La función más elemental que nos ofrece la API de Google Maps es crear un mapa. El mapa se mostrará de forma muy parecida a cuando se copia el código HTML del sitio de Google Maps: centrado en un punto y con un nivel de *zoom* determinado. Además, se mostrarán una serie de controles mínimos que nos permitirán desplazarnos por el mapa, variar su ampliación y seleccionar el tipo de mapa mostrado.

El código siguiente muestra un mapa centrado en el rectorado de la UOC:

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type"
content="text/html; charset=utf-8"/>
<title>Universitat Oberta de Catalunya</title>

<script src="http://maps.google.com/maps?file=api ↵
&amp;v=2&amp;key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ↵
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
type="text/javascript"></script>

<script type="text/javascript">
function InicializarMapa() {
  var mapa;
  var centro;

  if (GBrowserIsCompatible()) {
    mapa = new GMap2(
      document.getElementById("elemento_mapa"));
    centro = new GLatLng(41.414829, 2.133003);
    mapa.setCenter(centro, 17);
    mapa.setUIToDefault();
  }
}
</script>

</head>
<body onload="InicializarMapa()" onunload="GUnload()">
<div id="elemento_mapa"
style="width: 500px; height: 300px"></div>
</body>
</html>
```

Como podemos observar, el código es una mezcla entre HTML y JavaScript. El código JavaScript (que hemos resaltado convenientemente mediante recua-

dros) se incluye mediante el elemento *script* y siempre va situado en la cabecera del documento (*head*). Observad el uso del atributo "type" ('tipo') del elemento *script*, que identifica el código como JavaScript.

También es importante observar las similitudes entre Java y JavaScript. Quizá la diferencia más notable sea la ausencia de tipos. Todas las variables son del mismo tipo, aunque como ya veremos, internamente puedan tener atributos distintos. Consecuentemente, las variables se declaran usando la fórmula ***var*** *nombre_variable* y las funciones, ***function*** *nombre_función*.

Veamos los pasos que hay que seguir para mostrar un mapa en nuestra página: en primer lugar, es necesario crear un elemento HTML que albergue el mapa. Aunque en la práctica hay diversos elementos que pueden desempeñar esta función, es recomendable utilizar *div*, porque es el elemento más neutro. Al elemento utilizado debemos darle un nombre mediante el atributo "id" (de *identifier*, 'identificador') y, adicionalmente, un tamaño, que especificaremos mediante los atributos estilísticos *width* (ancho) y *height* (alto), y que será el tamaño del mapa medido en píxeles. En el ejemplo, se ha creado el elemento mediante el código siguiente:

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

Como se puede observar, al elemento se le ha llamado "elemento_mapa", y se le han atribuido las dimensiones iniciales de 500 por 300 píxeles.

Una vez creado el elemento que albergará el mapa, debemos escribir el código JavaScript que se encargará de inicializarlo y mostrarlo. Para llevar a cabo esta tarea, hemos de realizar tres acciones:

- incluir el código de la API de Google Maps,
- escribir una función que se encargue de inicializar el mapa, y
- llamar a la función de inicialización cuando el navegador termine de dibujar la página.

Veámoslas una a una:

Incluir el código de la API de Google Maps

Mediante la primera acción, damos acceso a nuestro código a la API de Google Maps. Esta acción se realiza en el primer bloque de código JavaScript:

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

Por regla general, este código siempre será el mismo. Su única peculiaridad es el valor "key" (que en el ejemplo ha sido reemplazado por varias equis), que variará de un lugar web a otro y que se debe obtener de Google mediante un proceso de registro gratuito*. Una vez hecho esto, el código JavaScript de nuestra página podrá acceder a las funciones de la API de Google Maps.

*** http://code.google.com/intl/ca/apis/maps/signup.html**
No es necesario obtener una clave para las páginas almacenadas localmente en nuestro ordenador.

Escribir una función que se encargue de inicializar el mapa

La segunda acción crea el mapa sobre el elemento HTML seleccionado y lo inicializa, estableciendo el centro y el nivel de *zoom* iniciales. En el ejemplo, esta acción la lleva a cabo la función *InicializarMapa* en el segundo bloque de código JavaScript:

```
<script type="text/javascript">
function InicializarMapa() {
  var mapa;
  var centro;

  if (GBrowserIsCompatible()) {
    mapa = new GMap2(
      document.getElementById("elemento_mapa"));
    centro = new GLatLng(41.414829, 2.133003);
    mapa.setCenter(centro, 17);
    mapa.setUIToDefault();
  }
}
</script>
```

Como se puede observar, en primer lugar instanciamos un objeto "GMap2", que es el mapa en sí, ubicándolo en el elemento HTML "elemento_mapa", que es el que hemos creado al principio con este objetivo. Después, establecemos el centro y el nivel de *zoom* iniciales del mapa, mediante la función *setCenter* de la clase "GMap2". Como primer argumento, esta función requiere un objeto "GLatLng", que no es más que una coordenada geográfica expresada en función de su latitud y su longitud (según el sistema de coordenadas WGS84). El segundo elemento es un entero, que puede variar entre 0 y 19, por los niveles de *zoom* mínimo y máximo respectivamente.

La llamada a la función *GBrowserIsCompatible* evita la ejecución del código en caso de que se detecte que el navegador utilizado no es compatible con Google Maps.

Llamar a la función de inicialización cuando el navegador termine de dibujar la página

Finalmente, sólo queda indicar al navegador cuándo se debe inicializar el mapa o, lo que es lo mismo, cuándo debe ejecutarse la función *InicializarMapa*.

Mientras un navegador carga una página HTML y la construye, se obtienen, a la vez, imágenes externas y otros *scripts* que puedan formar parte de la página. Durante este proceso de carga, el entorno puede no ser lo bastante estable para Google Maps, y ello podría conducir a un comportamiento errático de la aplicación. Google recomienda inicializar el mapa sólo cuando la página haya sido cargada completamente.

Para ello, utilizaremos el atributo "onload" del elemento HTML *body*. En él podemos introducir código JavaScript que se ejecutará cuando la página haya sido cargada completamente. En nuestro caso, pondremos una llamada a la función *InicializarMapa*:

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

De la misma forma, cuando se descarga la página, cosa que sucede cuando se cambia a otra, llamaremos a la función *GUnload* de la API de Google Maps para liberar correctamente los recursos que haya consumido el mapa.

El resultado será el siguiente:

2.2.2. Tipos de mapas

De forma predeterminada, la API de Google Maps muestra el mapa de calles y carreteras. Aunque el tipo de mapa puede cambiarse mediante los controles situados en su parte superior derecha, a veces es deseable mostrar uno u otro tipo desde buen comienzo.

Podemos cambiar el tipo de mapa mediante la función *setMapType* de la clase "GMap2". La función espera un solo parámetro, el tipo de mapa, que ha de ser uno de los siguientes:

- **G_NORMAL_MAP**, para el mapa de calles y carreteras,
- **G_SATELLITE_MAP**, para el mapa fotográfico,

- **G_HYBRID_MAP**, para el mapa híbrido (mapa fotográfico con las calles y las carreteras superpuestas), y
- **G_PHYSICAL_MAP**, para el mapa de relieve.

Siguiendo con el ejemplo anterior, si agregáramos la línea siguiente al final de la función *InicializarMapa*:

```
mapa.setMapType(G_HYBRID_MAP);
```

obtendríamos el mapa híbrido. El código completo de la función quedaría así:

```
<script type="text/javascript">
function InicializarMapa() {
  var mapa;
  var centro;

  if (GBrowserIsCompatible()) {
    mapa = new GMap2(
      document.getElementById("elemento_mapa"));
    centro = new GLatLng(41.414829, 2.133003);
    mapa.setCenter(centro, 17);
    mapa.setUIToDefault();
    mapa.setMapType(G_HYBRID_MAP);
  }
}
</script>
```

y el resultado que obtendríamos sería el siguiente:

2.2.3. Transformación de coordenadas

Una funcionalidad importante que nos ofrecen las herramientas SIG es la de transformar las coordenadas de la pantalla en coordenadas geográficas y viceversa. Así, podemos determinar las coordenadas geográficas de un punto del mapa o situar en el mapa un punto geográfico determinado. Veamos cómo se manejan ambos casos con Google Maps.

Supongamos que queremos marcar una coordenada geográfica sobre el mapa. Para este propósito, Google Maps nos proporciona marcadores ("GMarker") que se pueden emplazar en una posición determinada. El constructor de la clase "GMarker" espera un objeto "GLatLng" que indique la coordenada geográfica que deberá señalar.

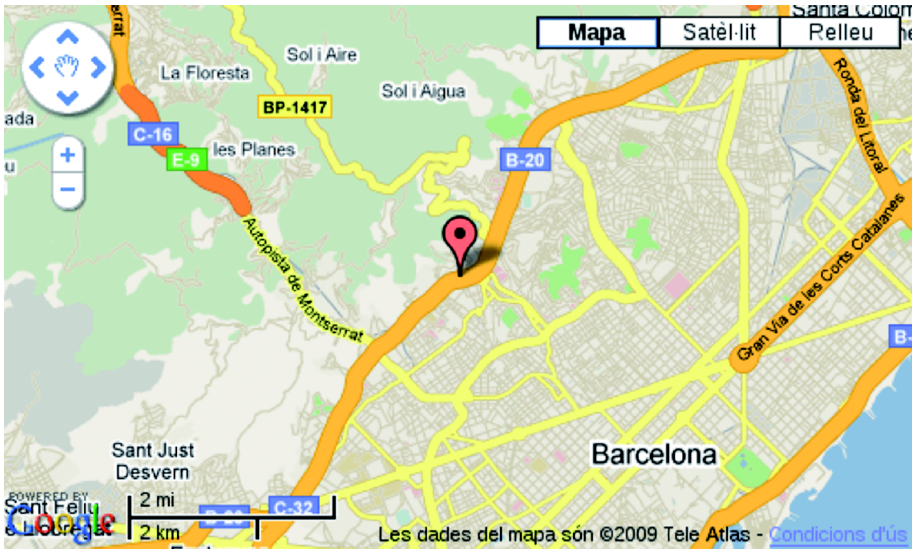
El ejemplo siguiente destaca la situación del rectorado de la UOC en medio de Barcelona mediante el uso de un marcador:

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type"
content="text/html; charset=utf-8"/>
<title>
Rectorado de la Universitat Oberta de Catalunya
</title>
<script src="http://maps.google.com/maps?file=api ↵
&amp;v=2&amp;key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ↵
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
type="text/javascript"></script>
<script type="text/javascript">
function InicializarMapa() {
 var mapa;
 var posicion_rectorado;
 var marca_rectorado;

 if (GBrowserIsCompatible()) {
  posicion_rectorado = new GLatLng(41.414829, 2.133003);

  // crea el mapa y lo inicializa
  mapa = new GMap2(
    document.getElementById("elemento_mapa"));
  mapa.setCenter(posicion_rectorado, 12);
  mapa.setUIToDefault();

  // crea un marcador sobre el rectorado de la UOC
  marca_rectorado = new GMarker(posicion_rectorado);
```

```
    mapa.addOverlay(marca_rectorado);
  }
}
</script>
</head>
<body onload="InicializarMapa()" onunload="GUnload()">
<div id="elemento_mapa"
style="width: 500px; height: 300px"></div>
</body>
</html>
```

Como se puede observar, en primer lugar se crea un objeto "GLatLng" con la posición del rectorado de la UOC: (41,414829N, 2,133003E). Este objeto es utilizado a posteriori para establecer el centro del mapa (función *setCenter*) y para crear el marcador. Finalmente, la función *addOverlay* (agregar capa) agrega el marcador al mapa.

El resultado será el siguiente:



Supongamos ahora que deseamos hacer el proceso inverso: determinar la posición geográfica de un punto del mapa. Será necesario recurrir a esta transformación cuando, por ejemplo, deseemos conocer las coordenadas geográficas de una posición del mapa sobre la que hayamos hecho clic.

En el ejemplo siguiente se muestran las coordenadas geográficas (expresadas en grados norte y este) del punto del mapa sobre el que se hace clic:

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<meta http-equiv="content-type"
content="text/html; charset=utf-8"/>
<title>Buscar coordenadas</title>
<script src="http://maps.google.com/maps?file=api ↵
&amp;v=2&amp;key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ↵
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
type="text/javascript"></script>
<script type="text/javascript">
function CuandoClic(capa, latlng, capaLatlng) {
  this.openInfoWindowHtml(latlng,
    latlng.lat() + "N<br>" + latlng.lng() + "E");
}
function InicializarMapa() {
  var centro_mapa;
  var mapa;

  if (GBrowserIsCompatible()) {
    centro_mapa = new GLatLng(41.414829, 2.133003);

    // crea el mapa y lo inicializa
    mapa = new GMap2(
      document.getElementById("elemento_mapa"));
    mapa.setCenter(centro_mapa, 15);
    mapa.setUIToDefault();

    // asocia los clics sobre el mapa con la función
    // "CuandoClic"
    GEvent.addListener(mapa, "click", CuandoClic);
  }
}
</script>
</head>
<body onload="InicializarMapa()" onunload="GUnload()">
<div id="elemento_mapa"
style="width: 500px; height: 300px"></div>
</body>
</html>
```

Para lograrlo, hemos asociado la función *CuandoClic* a los clics del ratón mediante la función *addListener* de la clase "GEvent":

```
GEvent.addListener(mapa, "click", CuandoClic);
```

El primer parámetro es el mapa sobre el que se escucharán los clics; el segundo, el evento que se ha de capturar (en este caso el "click", por el clic del ratón), y el último, la función a la que se llamará cada vez que se desate el evento.

Como hemos visto, cada vez que hagamos clic sobre el mapa se llamará a la función *CuandoClic*, que recibirá, por medio del parámetro "latlng" (de la clase "GLatLng"), las coordenadas geográficas sobre las que se ha apuntado.

La tarea de mostrar las coordenadas sobre el mapa la realiza la función *openInfoWindowHtml* del objeto "mapa":

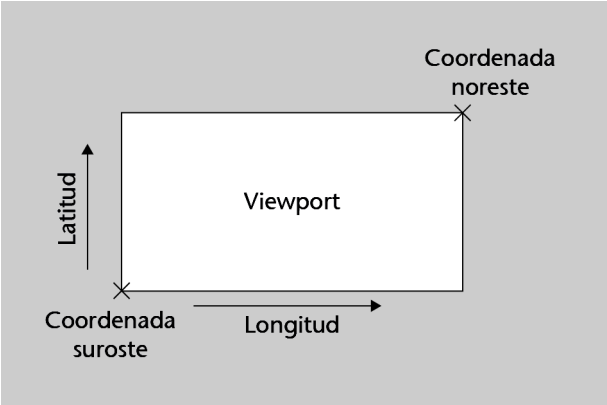
```
this.openInfoWindowHtml(latlng,
  latlng.lat() + "N<br>" + latlng.lng() + "E");
```

El primer parámetro son las coordenadas geográficas sobre las que apuntará el mensaje, y el segundo, el mensaje que se mostrará. El *this* se utiliza para hacer referencia al mapa actual.

Después de cada clic, obtendremos un resultado parecido al siguiente:

2.2.4. El *viewport*

Como ya hemos visto en el módulo anterior, el recuadro que delimita la parte visible del mapa recibe el nombre de *viewport*. También hemos visto que el *viewport* se caracteriza por las coordenadas geográficas de sus esquinas inferior izquierda (suroeste) y superior derecha (noreste), creciendo latitud y longitud desde la coordenada suroeste hasta la coordenada noreste.



Google Maps nos permite obtener los límites del *viewport* mediante la función *getBounds* de la clase "GMap2". Esta función devuelve un objeto de la clase "GLatLngBounds" que contiene las coordenadas geográficas de las esquinas suroeste y noreste del *viewport*. A su vez, dichas coordenadas las leeremos mediante las funciones *getLat* y *getLng*.

El ejemplo siguiente hace uso de la función *getBounds* para situar cinco marcas escogidas al azar dentro del *viewport*:

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type"
content="text/html; charset=utf-8"/>
<title>Marcas aleatorias</title>
<script src="http://maps.google.com/maps?file=api ↵
&amp;v=2&amp;key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ↵
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
type="text/javascript"></script>
<script type="text/javascript">
function InicializarMapa() {
  var centro_mapa;
  var mapa;
  var limites;
  var noreste;
  var suroeste;
  var incrementoLatitud;
  var incrementoLongitud;
  var i;
  var marca;

  if (GBrowserIsCompatible()) {
    centro_mapa = new GLatLng(41.414829, 2.133003);
```

```
    // crea el mapa y lo inicializa
    mapa = new GMap2(
      document.getElementById("elemento_mapa"));
    mapa.setCenter(centro_mapa, 15);
    mapa.setUIToDefault();

    // obtiene los bordes noreste y sureste del 'viewport'
    limites = mapa.getBounds();

    noreste = limites.getNorthEast();
    suroeste = limites.getSouthWest();

    // obtiene la diferencia de latitud y longitud entre
    // ambos puntos
    incrementoLatitud = noreste.lat() - suroeste.lat();
    incrementoLongitud = noreste.lng() - suroeste.lng();

    // crea los cinco puntos, multiplicando las diferencias
    // por un número entre cero y uno (Math.random())
    i = 0;
    while (i < 5) {
      punto = new GLatLng(suroeste.lat() +
        incrementoLatitud * Math.random(),
        suroeste.lng() +
        incrementoLongitud * Math.random());
      marca = new GMarker(punto);
      mapa.addOverlay(marca);
      i++;
    }

  }
}
</script>
</head>
<body onload="InicializarMapa()" onunload="GUnload()">
<div id="elemento_mapa"
style="width: 500px; height: 300px"></div>
</body>
</html>
```

Como se puede observar, en un primer paso se obtienen los límites sureste y noreste del *viewport* mediante la función "getBounds". Dichas coordenadas se almacenan en las variables "noreste" y "suroeste" respectivamente:

```
limites = mapa.getBounds();

noreste = limites.getNorthEast();
suroeste = limites.getSouthWest();
```

Hecho esto, en un segundo paso, se obtienen el ancho y el alto del *viewport*, restando por separado las latitudes y las longitudes de ambos puntos. Estas medidas, que representan el incremento máximo que pueden experimentar latitud y longitud, se almacenan en las variables "incrementoLatitud" e "incrementoLongitud" respectivamente:

```
incrementoLongitud = noreste.lng() - suroeste.lng();
incrementoLatitud = noreste.lat() - suroeste.lat();
```

Ahora ya sólo queda sumar a la coordenada suroeste del *viewport* una latitud entre 0 e "incrementoLatitud" y una longitud entre 0 e "incrementoLongitud" para obtener una nueva coordenada dentro del *viewport*. Estos valores aleatorios se pueden obtener multiplicando "incrementoLatitud" e "incrementoLongitud" por un valor aleatorio entre 0 y 1, que es lo que proporciona la función *Math.random*:

```
punto = new GLatLng(suroeste.lat() +
  incrementoLatitud * Math.random(),
  suroeste.lng()
+ incrementoLongitud * Math.random());
```

Finalmente, se agregan las marcas al mapa mediante la función *addOverlay*.

El resultado será parecido al siguiente:

2.2.5. Líneas y polígonos

Google Maps nos permite representar objetos sobre un mapa, de manera que su geometría esté ligada a unas coordenadas geográficas determinadas. Buen ejemplo de ello son las marcas que hemos visto hasta ahora: se acomodan a los desplazamientos del mapa y a los cambios de zoom para apuntar siempre sobre el mismo lugar.

Sin embargo, las opciones que nos ofrece Google Maps no se acaban en los marcadores: también podemos representar líneas y polígonos.

*** Este ejemplo sólo pretende mostrar cómo trazar líneas con un objeto "GPolyline". Google Maps tiene mecanismos propios para trazar rutas entre dos puntos automáticamente. Si nuestro objetivo es obtener una ruta por carretera para ir de una dirección a otra, deberíamos usar la clase "GDirections".**

Supongamos que queremos representar la ruta que hay que seguir para ir de Plaça de Catalunya de Barcelona al rectorado de la UOC. El código siguiente dibuja la ruta mediante un objeto "GPolyline"*:

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type"
content="text/html; charset=utf-8"/>
<title>Cómo llegar a la UOC</title>
<script src="http://maps.google.com/maps?file=api ↵
&amp;v=2&amp;key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ↵
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
type="text/javascript"></script>
<script type="text/javascript">

function InicializarMapa() {
 var centro_mapa;
 var mapa;
 var ruta;

 if (GBrowserIsCompatible()) {
  centro_mapa = new GLatLng(41.38693554508235,
   2.169950008392334);

  // crea el mapa y lo inicializa
  mapa = new GMap2(
   document.getElementById("elemento_mapa"));
  mapa.setCenter(centro_mapa, 15);
  mapa.setUIToDefault();

  // dibuja la ruta
  ruta = new GPolyline(
   [ centro_mapa,
   new GLatLng(41.38799806199135, 2.1700572967529297),
```

```
      new GLatLng(41.39651366867027, 2.1594786643981934),
      new GLatLng(41.39981336757232, 2.1555089950561523),
      new GLatLng(41.40201843881663, 2.1529340744018555),
      new GLatLng(41.403965921268934, 2.1518611907958984),
      new GLatLng(41.40649273299253, 2.1495437622070312),
      new GLatLng(41.408279145680964, 2.1474623680114746),
      new GLatLng(41.41122420530506, 2.14508056640625),
      new GLatLng(41.41177136014642, 2.140810489654541),
      new GLatLng(41.415118560018506, 2.136197090148926),
      new GLatLng(41.41589096626463, 2.1361541748046875),
      new GLatLng(41.41479672137174, 2.1332359313964844) ],
      "#ff0000", 10, 0.5);
    mapa.addOverlay(ruta);
  }
}
</script>
</head>
<body onload="InicializarMapa()" onunload="GUnload()">
<div id="elemento_mapa"
style="width: 500px; height: 300px"></div>
</body>
</html>
```

Como se puede observar, el constructor de la clase "GPolyline" espera un vector con los puntos geográficos que constituyen la línea, su color, expresado en notación hexadecimal* ("#ff0000" por rojo), y su anchura, medida en píxeles (10). El cuarto parámetro, que es opcional, indica el grado de opacidad de la línea (0: transparente; 1: opaca). Observad que en JavaScript basta con encerrar una lista de objetos entre corchetes para crear un vector.

*** El código de color se puede obtener de cualquier herramienta de dibujo o edición fotográfica. Si no, hay paletas de colores disponibles en la Web (por ejemplo, en Wikipedia).**
http://es.wikipedia.org/wiki/Colores_HTML#Tabla_de_colores.

El resultado será el siguiente:

Basta con seguir la línea roja para llegar a nuestro destino.

Es importante destacar que, de forma predeterminada, las líneas dibujadas sobre un mapa mediante la clase "GPolyline" son líneas rectas respecto a la proyección, no respecto a la realidad. Esto es así porque la Tierra es curva, y para representarla sobre un mapa debemos deformarla ligeramente. Si la línea que queremos dibujar es suficientemente larga (de varios kilómetros), lo que sobre el mapa es recto, sobre el terreno será más o menos curvo y viceversa.

Para dibujar lineas geodésicas (aquellas que representan la distancia más corta entre dos puntos de la superficie terrestre), debemos pasarle al constructor de la clase "GPolyline" la opción *geodesic: true*. El ejemplo siguiente dibuja dos líneas entre Moscú y Pekín: la más corta sobre el mapa (en rojo) y la más corta sobre el terreno (en azul):

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type"
content="text/html; charset=utf-8"/>
<title>Moscú - Pekín</title>
<script src="http://maps.google.com/maps?file=api ↵
&amp;v=2&amp;key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ↵
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
type="text/javascript"></script>
<script type="text/javascript">
function InicializarMapa() {
  var centro_mapa;
  var mapa;
  var moscu;
  var pekin;
  var recta_mapa;
  var recta_terreno;

  if (GBrowserIsCompatible()) {
    centro_mapa = new GLatLng(45.9511496866914,
      79.1015625);

    // crea el mapa y lo inicializa
    mapa = new GMap2(
      document.getElementById("elemento_mapa"));
    mapa.setCenter(centro_mapa, 2);
    mapa.setUIToDefault();

    moscu = new GLatLng(55.7516682526483,
      37.61860370635986);
    pekin = new GLatLng(39.91582588231232,
      116.39079093933105);
```

```
    // dibuja la línea recta respecto a la proyección
    recta_mapa = new GPolyline([ moscu, pekin ],
      "#ff0000", 10, 0.5);
    mapa.addOverlay(recta_mapa);
    // dibuja la línea recta respecto al terreno
    recta_terreno = new GPolyline([ moscu, pekin ],
      "#0000ff", 10, 0.5, {geodesic: true});
    mapa.addOverlay(recta_terreno);
  }
}
</script>
</head>
<body onload="InicializarMapa()" onunload="GUnload()">
<div id="elemento_mapa"
style="width: 500px; height: 300px"></div>
</body>
</html>
```

El resultado será el siguiente:

De entre todas las funciones que nos ofrece la clase "GPolyline", quizás la más interesante sea la *getLength*. Esta función nos permite obtener la distancia real en metros de una línea, independientemente de si se ha representado como geodésica o no.

Aplicada a los objetos "recta_mapa" y "recta_terreno" del ejemplo anterior, en ambos casos la función "getLength" devolverá 5798434,48 m:

```
recta_mapa.getLength();     // devuelve 5798434,48
recta_terreno.getLength(); // devuelve 5798434,48
```

Dibujar polígonos

Aunque con un objeto "GPolyline" sería posible dibujar un polígono (en el caso de que el último punto fuera igual al primero), Google Maps nos ofrece una clase especializada para este propósito: "GPolygon".

Al igual que con las líneas, podemos definir el color, el ancho y la opacidad del borde, además de poder definir el color y la opacidad del relleno.

Las líneas siguientes calculan el área de la Plaça de Catalunya de Barcelona:

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type"
content="text/html; charset=utf-8"/>
<title>Área de Plaça de Catalunya</title>
<script src="http://maps.google.com/maps?file=api ↵
&amp;v=2&amp;key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ↵
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
type="text/javascript"></script>
<script type="text/javascript">
function InicializarMapa() {
 var centro_mapa;
 var mapa;
 var poligono;

 if (GBrowserIsCompatible()) {
  centro_mapa = new GLatLng(41.3870240888213,
   2.1699607372283936);

  // crea el mapa y lo inicializa
  mapa = new GMap2(
   document.getElementById("elemento_mapa"));
  mapa.setCenter(centro_mapa, 16);
  mapa.setUIToDefault();

  // dibuja el área
  poligono = new GPolygon(
   [ new GLatLng(41.38787732230797, 2.169971466064453),
   new GLatLng(41.38741850946622, 2.168748378753662),
   new GLatLng(41.38585691167221, 2.1701431274414062),
   new GLatLng(41.38691944620778, 2.1712052822113037),
   new GLatLng(41.38787732230797, 2.169971466064453) ],
    "#ff0000", 10, 0.5, "#ff0000", 0.2);
  mapa.addOverlay(poligono);
```

```
    // muestra el área del polígono
    mapa.openInfoWindowHtml(centro_mapa,
      Math.round(poligono.getArea()) + " m<sup>2</sup>");
  }
}
</script>
</head>
<body onload="InicializarMapa()" onunload="GUnload()">
<div id="elemento_mapa"
style="width: 500px; height: 300px"></div>
</body>
</html>
```

Como se puede observar, el constructor de la clase "GPolygon" es muy parecido al de la clase "GPolyline". Los tres primeros parámetros son idénticos: vector de puntos, color y opacidad del borde. Los dos parámetros que siguen son el color y la opacidad del relleno. Observad que el primer y el último de los puntos del vector son iguales. Google recomienda que sea así para evitar "comportamientos inesperados"*.

*** Cita textual de la *Maps API Developer's Guide:* http://code.google.com/intl/ca/apis/maps/documentation/javascript/v2/overlays.html#Polygons_Overview**

La última línea utiliza la función *getArea* para obtener el área, en metros cuadrados, del polígono. Observad que se utiliza la función *round* de la clase "Math" para redondear el resultado. El área obtenida se muestra por pantalla mediante la función *openInfoWindowHtml*:

```
mapa.openInfoWindowHtml(centro_mapa,
  Math.round(poligono.getArea()) + " m<sup>2</sup>");
```

El resultado será el siguiente:

Resumen

Después de aprender a programar aplicaciones SIG de escritorio, en este módulo se ha hecho lo propio con aplicaciones web. Si en el caso de las aplicaciones de escritorio nos hemos basado en un sistema ya implementado (gvSIG), con las aplicaciones web hemos hecho algo parecido: hemos utilizado la API pública de Google Maps.

Antes de abordar la estructura y las funciones de la API, hemos aprendido a crear una página web simple y a incorporarle un mapa. Después, hemos utilizado las funciones de la API para personalizar la presentación del mapa (coordenadas iniciales, zoom y tipo de mapa) y representar puntos, y hemos aprendido a transformar las coordenadas del mapa a coordenadas geográficas.

Finalmente, se han estudiado las funciones que permiten dibujar líneas y polígonos, caracterizando rutas y áreas que, más tarde, mediante funciones dedicadas, hemos podido medir.

Aunque todos los ejemplos están basados en la API de Google Maps, las técnicas explicadas en este módulo se pueden extrapolar más o menos directamente a otras API, libres o comerciales.

Bibliografia

Pere Barnola Augé, "*Introducción a la creación de páginas web*", http://mosaic.uoc.edu/2009/11/20/introduccion-a-la-creacion-de-paginas-web/ , 09/2008.

W3C, "*HTML 4.01 Specification*", http://www.w3.org/TR/1999/REC-html401-19991224, 24/12/1999.

Varios autores, "*Parque nacional Hoge Veluwe*", http://es.wikipedia.org/wiki/Parque_nacional_Hoge_Veluwe.

Google, "*Google Maps API Concepts*", http://code.google.com/intl/es/apis/maps/documentation/javascript/v2/basics.html.

Google, "*Google Maps JavaScript API V2 Reference*", http://code.google.com/intl/es/apis/maps/documentation/javascript/v2/reference.html.