Ejercicio: Las antípodas

Introducción

Las antípodas representan aquello que se encuentra en una posición radicalmente opuesta. Desde un punto de vista geográfico representaría un punto justo en el lado opuesto de la Tierra, es decir, dado un punto, significaría el punto más alejado posible o visto de otro modo, como si un túnel atravesara la Tierra y este túnel fuera lo más largo posible.

Vamos a elaborar una aplicación sencilla que represente dos mapas de forma que al mover uno de ellos automáticamente el segundo mapa se centre en la antípoda. Es posible encontrar una aplicación de este tipo en internet, por ejemplo, aquí:
http://www.findlatitudeandlongitude.com/antipode-map

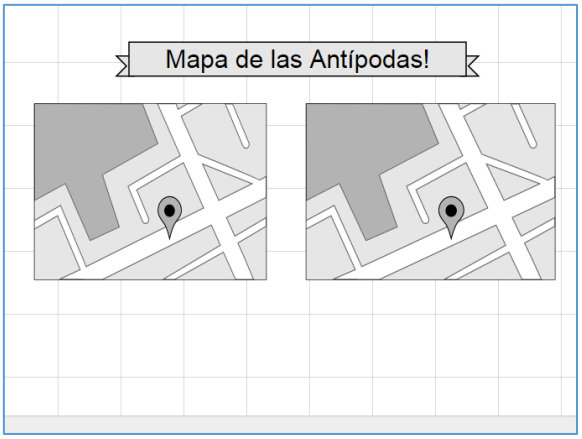
Con el API de Esri para JavaScript es realmente sencillo elaborar este tipo de aplicación si tenemos en cuenta algunos conceptos GIS, como el paso de coordenadas entre diferentes sistemas de referencia, y de desarrollo, como la gestión de eventos del mapa y las diferentes propiedades o atributos de este.

HTML y el DOM

El DOM de una página web representa los elementos HTML que la componen. En este caso vamos a tratar de elaborar una página web muy sencilla que conste de dos elementos de tipo "div" y que cada uno de ellos albergue cada uno de los mapas que estarán sincronizados.

No solo necesitamos que haya dos elementos "div", además, queremos ver los dos en la misma línea flotante (float) por lo que será necesario aplicar algunos estilos de tipo CSS.

Así, nuestra aplicación, dentro del "body" contendrá tan solo dos "divs" y, opcionalmente, un título:



Empezando… ¿desde cero?

No es recomendable comenzar una aplicación usando un editor y una página en blanco. Se recomienda basar el desarrollo en algún ejemplo que ya exista y que sea sencillo. Por ejemplo: http://developers.arcgis.com/javascript/sandbox/sandbox.html?sample=map_simple

Descarga el código fuente, guárdalo en tu servidor IIS y comienza a editarlo.

Dos mapas, dos divs

Como ya hemos dicho, nuestros usuarios necesitan dos mapas que estén sincronizados y para ello necesitamos dos "divs". La aplicación descargada ya contiene uno, crea un segundo cuyo "id" sea "map2". Añade a ambos el código CSS necesario para que "floten" en la misma línea de visión.

Una vez tengas los dos "divs" flotantes, será necesario "cargarlos" con un mapa. El primero de ellos ya lo tiene en el ejemplo descargado. Crea una nueva variable llamada map2 y crea para ella un nuevo objeto Map pasándole como parámetro el "id" del segundo "div".

Eventos

Sincronizar los mapas significa que cuando el usuario realice un pan o un zoom del mapa1, el mapa2 se actualice para centrarse en la antípoda. Por lo tanto, debemos ser capaces de saber cuándo el usuario realiza alguna de estas operaciones.

Desde el punto de vista de desarrollo, cuando un usuario interactúa con una aplicación, sucede lo que llamamos un evento. Por ejemplo, cuando pinchamos sobre un botón, sucede el evento "click" sobre ese botón.

Todos los objetos que creamos tienen ciertos eventos asociados. Por ejemplo, el objeto Map del API de JavaScript tiene asociados diferentes eventos, estos son algunos:

- **basemap-change**: sucede cuando el mapabase se cambia, como sucede en ArcGIS Online cuando seleccionamos un mapabase diferente.
- **click**: el usuario hace click con el ratón sobre el mapa.
- **layers-add-result**: sabemos que un mapa tiene siempre diferentes capas. Este evento sucede cuando todas ellas ya se han cargado en el mapa.
- **extent-change**: este es el evento que buscamos, el usuario ha hecho zoom o pan y por lo tanto la extensión del mapa, ha cambiado. Llamamos extensión a los valores x e y tanto máximos como mínimos que definen la parte del mapa que se está visualizando.

Comprueba en la página de ayuda del API de JavaScript cómo es este evento del objeto mapa: https://developers.arcgis.com/javascript/jsapi/map-amd.html#event-extent-change

Un objeto mapa siempre tiene todos estos eventos asociados pero para poder utilizarlos tenemos que "suscribirnos" a ellos. Esto se debe a que cuando programamos no necesitamos

utilizar todos los eventos disponibles pero habrá casos en los cuales nos interese alguno de ellos. En este caso nos interesa el evento “extent-change” para saber cuándo el usuario se mueve por el mapa.

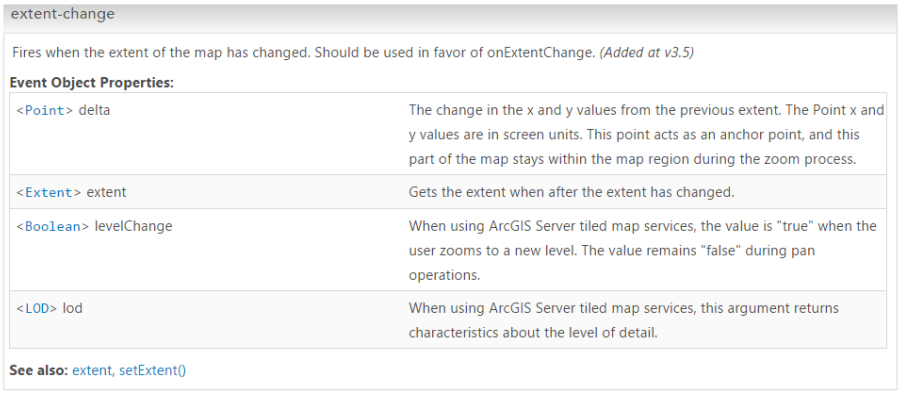
Para suscribirnos a un evento podemos utilizar la sintaxis “on” del objeto con el cual está relacionado. En este caso, nos suscribimos al evento “extent-change” del objeto map:

```
map.on("extent-change", function(change){

    // TÚ CÓDIGO IRÁ AQUÍ

}
```

Un evento suele ofrecernos información. En este caso, como hemos visto en la página de ayuda, el evento “extent-change” nos devuelve un objeto, que en la imagen superior hemos denominado “change”, que tiene todas estas propiedades, échales un vistazo:



extent-change

Fires when the extent of the map has changed. Should be used in favor of onExtentChange. *(Added at v3.5)*

Event Object Properties:

<Point> delta	The change in the x and y values from the previous extent. The Point x and y values are in screen units. This point acts as an anchor point, and this part of the map stays within the map region during the zoom process.
<Extent> extent	Gets the extent when after the extent has changed.
<Boolean> levelChange	When using ArcGIS Server tiled map services, the value is "true" when the user zooms to a new level. The value remains "false" during pan operations.
<LOD> lod	When using ArcGIS Server tiled map services, this argument returns characteristics about the level of detail.

See also: extent, setExtent()

Entre otras cosas, vemos que nos ofrece la nueva extensión a la cual se ha movido el usuario de la aplicación. Esta extensión, como sabemos, representa las x e y mínimas y máximas, podríamos decir que se trata de la esquina superior izquierda y la esquina inferior derecha del mapa que estamos visualizando.

Sin embargo, nuestros usuarios quieren que el punto central del mapa1 esté sincronizado con el punto central del mapa2. Podemos obtener este punto central usando el método “getCenter()” de la clase “Extent”. Revísalo:
https://developers.arcgis.com/javascript/jsapi/extent-amd.html

Podemos aprovechar este punto central para sincronizar ambos mapas del siguiente modo:

```
map.on("extent-change", function(change){

    var center = change.extent.getCenter();
    map2.centerAt(center);

});
```

Ahora los dos mapas se sincronizan mostrando el mismo punto central pero no es lo que queremos. Necesitamos que el mapa2 se centre en otro punto, en realidad, justo en el opuesto.

El punto antípoda

Existen varios métodos para calcular el punto antípoda de uno dado. Uno de los métodos consiste desde un punto de vista GIS en negar la latitud, es decir, cambiar su signo y modificar en 180 grados la longitud.

Esta modificación de la longitud desde un punto de vista de desarrollo consistiría literalmente en: si la longitud del punto original es menor a cero, la nueva longitud sumará 180 grados; al contrario se le restarán.

El mayor problema con el cual nos encontramos para realizar esta operación de calcular el punto antípoda es que nuestro mapa usa por defecto un sistema de referencia Web Mercator y las operaciones que sabemos realizar son sobre latitud y longitud. El API de JavaScript nos proporciona una clase muy útil para realizar transformaciones entre sistemas denominada webMercatorUtils:
https://developers.arcgis.com/javascript/jsapi/esri.geometry.webmercatorutils-amd.html

El flujo de trabajo para calcular el punto antípoda cuando un usuario se mueva por el mapa será:

1. Obtener el punto central del mapa al cual se ha movido el usuario
2. Convertir ese punto central de Web Mercator a latitud/longitud
3. Realizar el cálculo del punto antípoda sobre esa latitud y longitud como se ha visto
4. Convertir esa nueva latitud/longitud antípoda de nuevo a coordenadas Web Mercator
5. Crear con esas coordenadas Web Mercator un nuevo punto central que será un objeto de tipo Point: https://developers.arcgis.com/javascript/jsapi/point-amd.html
6. Centrar el mapa2 a este nuevo punto

Opcional: Sincronización bidireccional

Trata de sincronizar el mapa2 y el mapa1 del mismo modo que hemos sincronizado el mapa1 con el mapa2.

Opcional: Título

Añade contenido HTML y CSS para poner un título a la aplicación y una explicación de lo realizado.