

A 3D Web Application for Weather Forecast Based on WebGL

Alejandro Graciano¹, Antonio J. Rueda^{*1}, Francisco R. Feito¹ y Francisco Martínez¹

¹Departamento de Informática, Escuela Politécnica Superior. Universidad de Jaén, Spain
{graciano, ajrueda, ffeito, fmartin}@ujaen.es

Abstract

Management, interpretation and visualization of geoscientific and atmospheric information is a major problem for the geoscientist and for a correct diffusion, due to its high volume, complexity, spatiotemporal dimension and three-dimensional nature. As a result, development of software tools for integration, exploration and graphical analysis of the growing amount of available geodata is of great importance. In the last decade a trend towards the implementation of these tools as web applications has consolidated because of its benefits: universal access, permanent availability and ease of collaborative use. However until recently Web development technologies have been ill-equipped for implementing interactive 2D and 3D applications dealing with large amounts of data as required by many geoscience applications. The support of WebGL by most modern browsers has changed this situation and promises to revolutionize the Web in the next few years with a new generation of visually appealing, high performance interactive applications. In this work we describe the technologies for 3D Web and their use in the context of atmospheric data using a 3D weather information application as case study.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications—

1. Introducción

En los últimos años la World Wide Web se ha convertido en un ecosistema natural para muchas aplicaciones geocientíficas. Podemos encontrar ejemplos en campos como la Hidrología [BJC*09, AHC*12, DJS14], Sismología [Div05, WWk], Climatología [Nat, EKR11] o en prevención de riesgos [JSST14]. Los servicios y aplicaciones Web se han convertido en la plataforma ideal para la difusión de la información geocientífica entre investigadores, políticos en puestos relacionados con la planificación de recursos, o el público en general.

La complejidad de los procesos que se estudian en las ciencias de la Tierra y los grandes volúmenes de información que se manejan, en muchos casos de naturaleza multidimensional, hacen especialmente difícil la tarea de diseñar herramientas con interfaces y representaciones visuales apropiadas que ayuden a observar e interpretar estos procesos. Si esta dificultad es importante en el caso del desarrollo

para entornos de escritorio tradicionales, todavía lo es más si se pretende implementar una herramienta con interfaz web:

- Los tiempos de visualización y respuesta son altos debido a la latencia de las redes de comunicaciones.
- Las posibilidades de interacción son menores en comparación con las aplicaciones de escritorio.
- Las capacidades de visualización son limitadas. Aunque HTML5 [HFL*14] ha mejorado mucho este aspecto, la fluidez de los gráficos no es comparable todavía a la de una aplicación de escritorio y la posibilidad de usar 3D está disponible desde hace relativamente poco tiempo.

En este artículo mostraremos cómo la tecnología WebGL [Mar13] puede ser usada para la implementación de aplicaciones web con una calidad de interacción y visualización 3D similares a los de una aplicación de escritorio. Nuestro caso de uso es un mashup que utiliza la información proporcionada por un servicio web gratuito para mostrar las condiciones atmosféricas en un mapa 3D interactivo de España usando iconos animados para cada uno de los fenómenos meteorológicos. Éste ejemplo sirve para ilustrar las posibilidades que tiene WebGL para el desarrollo de una nueva

* Autor principal

generación de aplicaciones web con un unas posibilidades visuales impensables hace unos años.

El resto del artículo se organiza como sigue: la Sección 2 describe los fundamentos relacionados con los servicios y aplicaciones web geocientíficas; la Sección 3 presenta una revisión sobre los servicios web meteorológicos y se indica el servicio web usado; la Sección 4 describe la tecnología WebGL y la librería Three.js; la Sección 5 describe la arquitectura cliente-servidor de nuestro sistema; la Sección 6 describe el diseño de los objetos meteorológicos usados; y finalmente la Sección 7 concluye y resume el presente trabajo.

2. Antecedentes

La evolución de las tecnologías de monitorización y teledetección de la Tierra ha generado un caudal ingente de información de naturaleza heterogénea que requiere de análisis e interpretación por parte de los investigadores (meteorólogos, hidrólogos, sismológicos, etc.). Para este fin se han desarrollado multitud de herramientas que son capaces de mostrar los distintos procesos implicados usando interfaces visuales avanzadas 3D/4D [HHS07, MS10, C JL*11, WYMS12].

La Web ha sido la gran revolución de las últimas dos décadas y rápidamente se ha convertido en una plataforma indispensable para el análisis, visualización e intercambio de información relacionada con la Tierra en la comunidad geocientífica. Sus ventajas son claras:

- Disponibilidad permanente.
- Acceso universal usando cualquier plataforma.
- Facilidad para el trabajo colaborativo.
- Flexibilidad a la hora de extender el servicio añadiendo nuevas funcionalidades o conjuntos de datos.
- Escalabilidad dependiendo de la carga de trabajo.

Existen abundantes ejemplos de herramientas web para visualización de la información atmosférica [EKR11, AH15], monitorización de la polución [WMS*13] o registro sísmico [WWWk]. Estas aplicaciones usan en general interfaces de usuario sencillas en 2D. En los últimos años los "virtual globes" han atraído mucho la atención ya que proporcionan una forma natural de visualización de los procesos naturales [CB11]. Los primeros "virtual globes" como Google Earth eran aplicaciones de escritorio pero los más recientes, basados en WebGL como OpenWebGlobe [CNL12] y Cesium [WWWc], están permitiendo la implementación de una nueva generación de aplicaciones web 3D con una calidad de visualización e interactividad sin precedentes.

Si nos centramos en la predicción meteorológica destinada al público en general es notable la evolución desde la aparición de la televisión en los años 50 cuando los presentadores, armados con lápices de cera, dibujaban mientras hablaban símbolos de los distintos meteoros sobre un

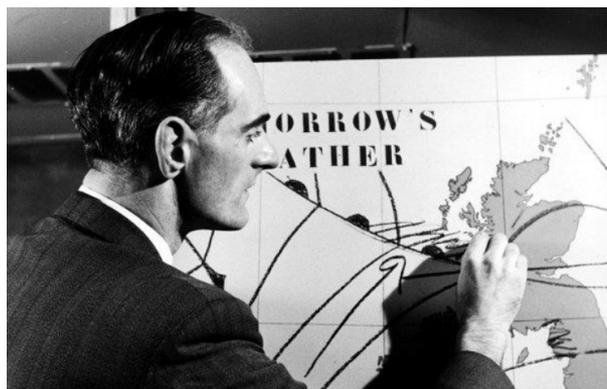


Figura 1: Hombre del tiempo en los años 60.

mapa, como ilustra la Figura 1. Hoy en día los informativos siguen mostrando la información del tiempo pero usando mapas digitales en 2D y 3D con iconos y gráficos generados por ordenador. Pero la Web se ha convertido probablemente en el medio principal de consulta de la predicción meteorológica. Los sitios como weather.com [WWWl], accuweather.com [WWWa] o eltiempo.es [WWWd] reciben millones de visitas todos los días. Desde el punto de vista visual estos sitios siguen siendo bastante conservadores, usando uno o varios mapas 2D y los correspondientes iconos o zonas de color para indicar temperaturas, probabilidad de precipitación, etc. Aunque sin duda esta metáfora visual sigue siendo efectiva, nuestro propósito en este trabajo es ir más allá explorando nuevas formas de mostrar la información atmosférica, presentando un sitio web de predicción atmosférica con una visualización 3D atractiva que incluye animaciones, efectos y una respuesta suave a la interacción con el usuario.

3. Servicios Web de Información Meteorológica

Para la obtención de la información atmosférica necesaria para nuestra aplicación, se han analizado los principales servicios web disponibles. Todos estos servicios incluyen una interfaz REST [Fie00] y usan XML o JSON [ECM] para la codificación de los mensajes. La Tabla 1, ubicada en el anexo, recoge las principales características de estos servicios.

El servicio proporcionado por weather.com [WWWn] ofrece un gran número de variables atmosféricas pero impone una restricción en el número de consultas por minuto en su servicio gratuito. myweather2.com [WWWg] también incluye una restricción en las consultas, aunque es más permisivo y podría ser usado en nuestro prototipo descrito en el artículo. Sin embargo, su descripción de las condiciones atmosféricas es limitada: no provee la cobertura de nubosidad o la precipitación esperada. Finalmente hemos elegido

openweathermap.com [WWWh] ya que permite un alto número de peticiones incluso en su versión no comercial además de ofrecer uno de los pronósticos más completos.

4. Tecnologías para la Web 3D: WebGL y Three.js

WebGL [Mar13] es una API JavaScript para el renderizado de gráficos 3D en los navegadores web. Es una especificación relativamente reciente desarrollada por el Grupo Khronos. Las principales características de esta tecnología se explican a continuación:

- API JavaScript sencilla implementada por la mayoría de los navegadores modernos. El renderizado se puede realizar en el elemento canvas de HTML5 en combinación con elementos web tradicionales.
- Alto rendimiento. Los navegadores web implementan la API WebGL usando llamadas al hardware gráfico subyacente.
- Facilidad para la portabilidad a diferentes plataformas y dispositivos ya que está basado en OpenGL ES 2.0.
- Soporte de OpenGL Shading Language (GLSL) para añadir efectos visuales o para procesamiento general de la GPU.
- Especificación abierta y libre.

WebGL es una tecnología cada vez más extendida, implementada en todos los navegadores de entornos de escritorio modernos y un número creciente de navegadores en plataformas móviles. Sin embargo, desde el punto de vista del desarrollador, WebGL es una API de bajo nivel que hace del renderizado de una escena simple una ardua tarea. Por ejemplo, mostrar un rectángulo centrado en el canvas requiere (sin el código HTML asociado) alrededor de 75 líneas de código con operaciones como la obtención del contexto de renderizado, inicialización de la ventana de visión, creación de uno o más buffers de vértices, etc. Debido a esto, han surgido numerosas librerías para simplificar la programación de las aplicaciones gráficas en entornos web: Three.js [Dir13], OSG.js [WWWi], Scene.js [WWWj] o SpiderGL [DBPGS10]. Entre estas, probablemente Three.js es la más popular. Sus principales características incluyen:

- Abstracción de la API WebGL a través de una interfaz de alto nivel y un grafo de escena sencillo.
- Luces y materiales predefinidos además de soporte completo de GLSL para la implementación de modelos de sombreado avanzados y efectos.
- Incluye objetos como cubos, esferas, toros, texto 3D, etc.
- Librería sencilla de objetos auxiliares como vectores, matrices, quaternions, etc.
- Soporte de interacción con el usuario con ratón y teclado (WebGL no proporciona ningún método de interacción).
- Aunque tiene su propio formato de fichero para imágenes, objetos y escenas, la librería incluye también algunos importadores y exportadores para otros formatos muy comunes (Blender, OBJ, etc.).

- Soporta renderizado en un canvas 2D si el navegador web no implementa WebGL, aunque el rendimiento se ve seriamente afectado.

El listado 1 muestra el código Three.js requerido para renderizar un cubo.

Listado 1: Escena simple en Three.js.

```

1
2  initScene = function(){
3      var scene = new THREE.Scene();
4      var camera = new THREE.PerspectiveCamera( 75,
5          window.innerWidth/window.innerHeight,
6          0.1, 1000 );
7
8      var renderer = new THREE.WebGLRenderer();
9      renderer.setSize( window.innerWidth,
10         window.innerHeight );
11
12     document.body.appendChild( renderer.domElement );
13
14     var geometry = new THREE.BoxGeometry( 1, 1, 1 );
15     var material = new THREE.MeshBasicMaterial(
16         { color: 0x00ff00 } );
17     var cube = new THREE.Mesh( geometry, material );
18
19     scene.add( cube );
20 }

```

5. Arquitectura de la aplicación

Nuestro sistema tiene una arquitectura cliente-servidor donde una aplicación de servidor Java almacena una lista de municipios y obtiene datos atmosféricos usando el servicio web openweathermap.com. Un cliente HTML muestra la condición atmosférica actual o la predicción en un mapa 3D de España usando WebGL/Three.js.

5.1. Diseño e implementación del servidor

El servidor proporciona al cliente las condiciones atmosféricas (actuales o futuras) para los municipios mostrados en la ventana de visión o para la petición del usuario. La información atmosférica se obtiene consultando a openweathermap.com. Como esta consulta puede tomar algo de tiempo, hemos usado una estrategia de cacheado almacenando en el servidor las coordenadas de la ventana de la porción de mapa que es visualizada en ese momento. Luego, el servidor comprueba si los datos atmosféricos para los municipios mostrados en ese área están en la caché y los actualiza (estos datos expiran después de un determinado tiempo). Si se encuentra en el servidor, la información se envía inmediatamente al cliente; si no, se consulta la previsión o condiciones atmosféricas al servicio web y se guardan en la caché para posteriores consultas.

Otra posible estrategia, actualmente en desarrollo, es la basada en mantener una base de datos local con la información atmosférica para todos los municipios reconocidos por el sistema. Las consultas se resolverían siempre usando la información disponible localmente. Un proceso en segundo plano se ejecutaría continuamente siendo responsable de

mantener la información de la caché actualizada consultando al servicio web cuando fuese requerido. Esta estrategia tiene una respuesta mucho mejor para el cliente a costa de sobrecargar el servicio web. La Figura 2 representa ambas alternativas para la arquitectura del servidor.

Hemos añadido a nuestro servidor las coordenadas y nombres de los 8119 municipios de España, usando un quad-tree [Sam05] como índice espacial para acelerar las consultas de la ventana enviadas por el cliente.

Como detalle final de implementación, se ha definido una interfaz para cualquier servicio atmosférico y una definición común de datos atmosféricos. Hemos implementado un mecanismo de extensión basado en el patrón SPI (Service Provider Interface) [Tha09] que permite detectar y cargar cualquier plugin que proporcione una implementación de *IWeatherProvider* (Listados 2 y 3) para conectar un nuevo servicio atmosférico. Se ha implementado únicamente un proveedor para *openweathermap.com* pero este mecanismo asegura una integración transparente y simple de cualquier servicio atmosférico futuro.

Listado 2: Datos comunes proveídos por los servicios web.

```

22
23 public class WeatherData {
24     float temperature;
25     int percentageClouds;
26     float precipitation;
27     boolean snow;
28     boolean storm;
29 }

```

Listado 3: Interfaz Java para cualquier servicios web atmosférico.

```

31
32 public interface IWeatherProvider {
33     public String providerDescription();
34     public WeatherData current(String location, String
35         country);
36     public WeatherData current(double latitude, double
37         longitude);
38     public WeatherData forecast(String location, String
39         country, Date date);
40     public WeatherData forecast(double latitude, double
41         longitude, Date date);

```

5.2. Diseño e implementación del cliente

El cliente es una aplicación gráfica implementada usando HTML5, JavaScript y la librería Three.js. En este cliente se muestra un mapa 3D del territorio Español texturizado con una imagen satélite.

Además se ha añadido un modelo de elevación para obtener más realismo y atractivo visual. La Figura 3 muestra la apariencia general de la aplicación. Los iconos en el mapa representan los diferentes fenómenos atmosféricos. Usando el ratón el usuario puede desplazarse por el mapa e incrementar el nivel de detalle (Figura 4) pero estos movimientos

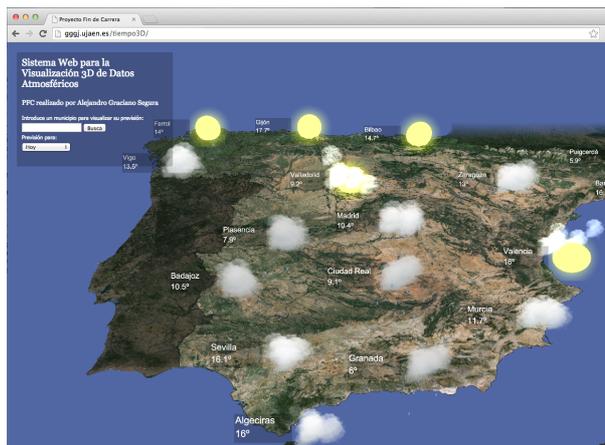


Figura 3: Captura de pantalla del cliente web.

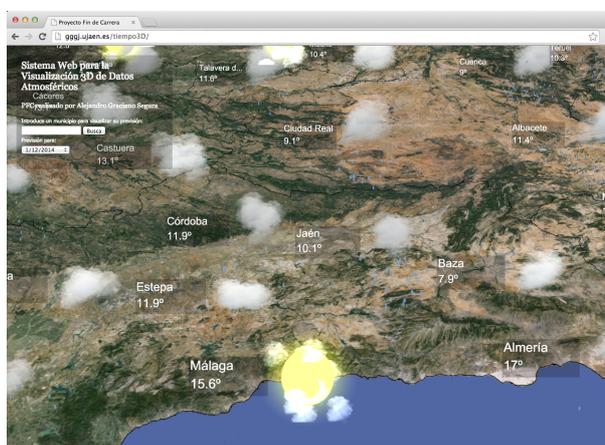


Figura 4: Zoom del este de Andalucía.

están limitados para evitar la visualización de áreas muy pequeñas o un desplazamiento hacia fuera del mapa. Alternativamente, si el usuario introduce el nombre de un municipio, el mapa se centrará en su ubicación con un nivel de detalle apropiado y mostrando las condiciones atmosféricas. Finalmente el usuario puede seleccionar en una lista la opción de mostrar las condiciones meteorológicas actuales o una previsión para el resto del día o para uno de los tres próximos días.

El mapa se ha dividido en cuadrantes y las consultas al servidor no se hacen por municipios individuales sino por uno o varios de estos cuadrantes que incluyen múltiples municipios. La razón de esto es simplemente la coherencia espacial: es muy probable que un usuario interesado en el tiempo de un municipio pueda estar también interesado en los

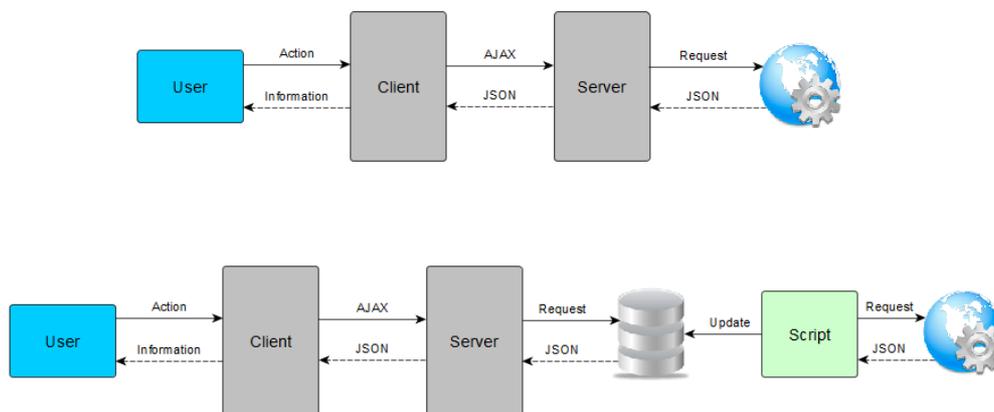


Figura 2: Arquitectura del servidor actual (arriba) y alternativa (abajo).

municipios cercanos. Este enfoque mejora el rendimiento del sistema evitando nuevas consultas.

La aplicación cliente JavaScript espera a las acciones del usuario, selecciona los cuadrantes del mapa visualizados actualmente en la ventana de visión y realiza la petición al servidor de los datos atmosféricos para estos cuadrantes a través de AJAX. No se requieren nuevas consultas mientras el usuario no se desplace por el mapa o incremente el nivel de detalle hacia un nuevo cuadrante. La respuesta del servidor es una lista de los municipios incluidos en el cuadrante con sus datos atmosféricos (Listado 2) codificados en JSON [ECM]. Finalmente para cada municipio, sus coordenadas geográficas se transforman a coordenadas de escena y su posición y nombre son mostrados en el mapa con un icono representando las condiciones meteorológicas.

Para dotar de relieve al mapa de España hemos descargado el correspondiente modelo de elevación (DEM) de la fuente de datos ASTER GDEM v2 [WWWb]. El DEM resultante tras la composición de los tiles con el software Global Mapper [WWWc] fue exportado como un fichero binario usándose de entrada para Three.js

El uso de una malla sin preprocesamiento no es adecuado para una aplicación gráfica web debido a su gran tamaño y al consiguiente tiempo de descarga. Para solventar este problema, la malla fue simplificada usando el método "Edge Collapse" implementado en la aplicación de modelado Blender [WWWf]. Con esta utilidad se eliminaron el 75% de los triángulos del DEM inicial (9525 triángulos finales). El resultado es mostrado en la Figura 5.

6. Diseño de los objetos meteorológicos

El cliente usa seis objetos gráficos diferentes para representar las condiciones meteorológicas: soleado, nuboso, par-

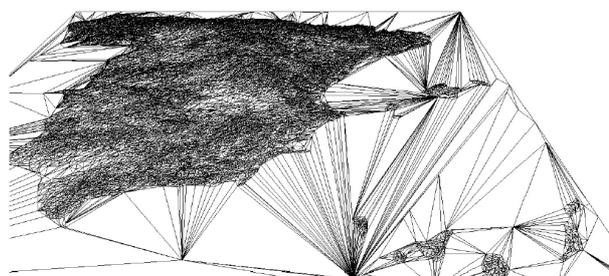


Figura 5: Resultado de la simplificación del DEM de España.

cialmente nuboso, lluvia, nieve y tormenta (Figura 7). Las animaciones de densidad de las nubes y la intensidad de la lluvia y nieve dependen de los parámetros recibidos desde el servidor (porcentaje de nubes, volumen de precipitación de lluvia o nieve). Para la implementación de estos objetos hemos usado técnicas estándares en Informática Gráfica como sprites o sistemas de partículas.

La implementación del icono que representa una previsión soleada se ha realizado a partir de una técnica de sprite. Este sprite tiene un comportamiento de billboard por lo que está continuamente orientado hacia la cámara. En combinación con una esfera, el sprite (que incluye una textura con nivel de transparencia de forma degradada) creará un efecto de luminosidad debido a un blending aditivo que unirá esta textura a las texturas de los elementos que se encuentren detrás del icono. En la figura 6 se puede observar el resultado al eliminar el blending aditivo al sprite.

Para la visualización del icono de nubosidad se ha optado por utilizar un sistema de partículas. En este sistema de partí-



Figura 6: Desactivación del *blending* aditivo asociado al *sprite*.

culas cada elemento será nuevamente un *sprite* y cada uno de ellos será colocado aleatoriamente dentro de un rango vertical y horizontal (quedando así combinados en un cuboide). A mayor densidad de nubes proporcionada por el servicio web, mayor será la densidad en el icono.

En cuanto a los iconos de lluvia y nieve se han realizado combinando al anterior icono un nuevo sistema de partículas clásico en el que cada partícula, ya sea una gota de lluvia o copo de nieve, cae hasta un umbral. Al igual que con el icono de nubosidad, la densidad de partículas dependerá de la densidad real de la precipitación.

Análogamente al icono de previsión soleada, se ha usado un *sprite* con comportamiento de *billboarding* para el icono de tormenta. En este caso un conjunto de *sprites* se van alternando en una animación. Este icono se ha formado también en combinación con el de nubosidad.

Finalmente se ha creado un icono que indica claros y nubes que simplemente es la unión del icono de soleado con el de nubosidad con la salvedad de que para este último icono se ha bajado tanto la densidad de *sprites* como su tamaño.

El servicio web devuelve como parámetro de nubosidad un porcentaje de cobertura de nubes. El criterio usado para mostrar un icono u otro dependiendo de este factor ha sido el siguiente: para una cobertura menor del 5% se mostrará el icono de soleado; para una cobertura mayor del 5% y menor del 50% se utiliza el icono de claros y nubes, y finalmente, para el resto de porcentajes se muestra un icono de nubes con una densidad proporcional a su porcentaje.

7. Conclusiones y trabajo futuro

En este artículo se ha descrito el uso de las nuevas tecnologías web 3D como WebGL y Three.js para el desarrollo de

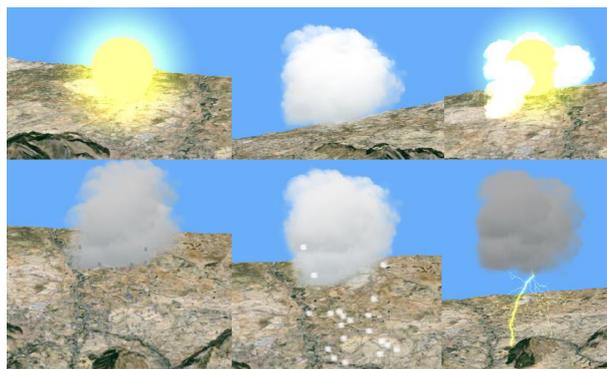


Figura 7: Catálogo de iconos 3D para la representación de las condiciones atmosféricas.

una aplicación de información atmosférica con una calidad visual y una interacción más avanzadas que las web tradicionales de este tipo. La aplicación web descrita incluye una interfaz gráfica con un mapa 3D, iconos animados realistas y una respuesta rápida y suave. Es de esperar que en los próximos años aparezcan numerosos geoportales usando estas tecnologías, ya sea directamente o indirectamente a través de frameworks para implementación de "virtual globes".

Se han realizado pruebas de rendimiento en tres tarjetas gráficas distintas. Los frames por segundo obtenidos de media para la primera tarjeta gráfica, una Nvidia GeForce GTX 280 con 1GB de memoria, han sido 54 fps. La segunda tarjeta gráfica, una Nvidia Tesla C2050 con 3GB de memoria dedicada, ha ofrecido una media de 58 fps, mientras que la tercera, una Intel HD 4000 con 1 GB de memoria, ha ofrecido un rendimiento medio de 43 fps.

La aplicación puede ser mejorada de diferentes maneras. En la Sección 5.1 describimos una alternativa para la implementación del servidor que, como se ha dicho anteriormente, está actualmente en fase de desarrollo. Ésta y otras alternativas se pueden implementar y probar para conseguir una mejora en el rendimiento en situaciones de alta carga de trabajo pero siendo razonable en el uso del servicio web que proporciona la información del tiempo. Otra posible mejora sería la inclusión de un nivel de detalle para el renderizado del terreno, dependiendo de los requisitos hardware de la máquina cliente, además de analizar resultados comparando la usabilidad sobre diferentes plataformas. Muchos datos atmosféricos como la temperatura o la dirección y velocidad del viento no tienen aún una metáfora visual y por tanto podría ser interesante su estudio. En un estado más desarrollado del sistema sería interesante realizar algún test a usuarios para comprobar la usabilidad del mismo. Finalmente, aunque los iconos de nubes son usados para representar condiciones locales, también se podrían utilizar para mostrar un mapa de nubes del país de una región en particular.

La aplicación descrita en este artículo se puede consultar en: <http://gggj.ujaen.es/tiempo3D>.

8. Agradecimientos

Este artículo ha sido parcialmente subvencionado por la Universidad de Jaén bajo el proyecto de investigación UJA2015/08/10 y por el Fondo Europeo de Desarrollo Regional (FEDER) a través del proyecto TIN2014/58/218/R.

Referencias

- [AH15] ALDER J., HOSTETLER S.: Web based visualization of large climate data sets. *Environmental Modelling & Software* 68, 0 (2015), 175 – 180. 2
- [AHC*12] AMES D. P., HORSBURGH J. S., CAO Y., KADLEC J., WHITEAKER T., VALENTINE D.: HydroDesktop: Web services-based software for hydrologic data discovery, download, visualization, and analysis. *Environ. Model. Softw.* 37 (Nov. 2012), 146–156. 1
- [BJC*09] BERAN B., JONATHAN S., COX D., VALENTINE D., ZASLAVSKY I., MCGEE J.: Web services solutions for hydrologic data access and cross-domain interoperability. *International Journal on Advances in Intelligent Systems* 2, 2&3 (2009), 317 – 324. 1
- [CB11] CHEN A., BAILEY J. (Eds.): *Computers & Geosciences. Special issue: Virtual Globes in Science* (2011), vol. 37. 2
- [CJL*11] CASTRILLÓN M., JORGE P., LÓPEZ I., MACÍAS A., MARTÍN D., NEBOT R., SABBAGH I., QUINTANA F., SÁNCHEZ J., SÁNCHEZ A., SUÁREZ J., TRUJILLO A.: Forecasting and visualization of wildfires in a 3d geographical information system. *Computers and Geosciences* 37, 3 (2011), 390–396. cited By 8. 2
- [CNL12] CHRISTEN M., NEBIKER S., LOESCH B.: Web-based large-scale 3D-geovisualisation using WebGL: The OpenWeb-Globe project. *International Journal on 3-D Information Modelling* 1, 3 (2012), 16 – 25. 2
- [DBPGS10] DI BENEDETTO M., PONCHIO F., GANOVELLI F., SCOPIGNO R.: SpiderGL: A JavaScript 3D graphics library for next-generation WWW. In *Web3D 2010. 15th Conference on 3D Web technology* (2010), pp. 165–174. note. 3
- [Dir13] DIRKSEN J.: *Learning Three.js: The JavaScript 3D Library for WebGL*. Packt Publishing, 2013. 3
- [Div05] DIVIACCO P.: An open source, Web based, simple solution for seismic data dissemination and collaborative research. *Computers & Geosciences* 31, 5 (2005), 599 – 605. 1
- [DJS14] DELIPETREV B., JONOSKI A., SOLOMATINE D. P.: Development of a Web application for water resources based on open source software. *Computers & Geosciences* 62, 0 (2014), 35 – 42. 1
- [ECM] Introducing JSON. <http://www.json.org>. 2, 5
- [EKR11] ERICKSON T. A., KOZIOL B. W., ROOD R. B.: OpenClimateGIS - A Web Service Providing Climate Model Data in Commonly Used Geospatial Formats. *AGU Fall Meeting Abstracts* (Dec. 2011), A1395. 1, 2
- [Fie00] FIELDING R. T.: *REST: Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000. 2
- [HFL*14] HICKSON I., FAULKNER S., LEITHEAD T., NAVARRA E. D., O'CONNOR E., PFEIFFER S.: *HTML5 Proposed Recommendation (10-28-2014)*. Tech. rep., W3C, 2014. 1
- [HHS07] HUFNAGEL R., HELD M., SCHRÖDER F.: Large-scale, realistic cloud visualization based on weather forecast data. *Proceedings of the 9th IASTED International Conference on Computer Graphics and Imaging, CGIM 2007* (2007), 54–59. cited By 3. 2
- [JSST14] JAEDICKE C., SYRE E., SVERDRUP-THYGESON K.: GIS-aided avalanche warning in Norway. *Computers & Geosciences* 66, 0 (2014), 31 – 39. 1
- [Mar13] MARRIN C.: *WebGL 1.0.2 specification*. Tech. rep., Khronos Group, 2013. 1, 3
- [MS10] MORRIS K., SCHWALLER M.: Data visualization and analysis tools for the global precipitation measurement (gpm) validation network. In *Geoscience and Remote Sensing Symposium (IGARSS), 2010 IEEE International* (July 2010), pp. 847–850. 2
- [Nat] NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION (USA): *Climate.gov*. 1
- [Sam05] SAMET H.: *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. 4
- [Tha09] THAKUR T.: *Service provider interface: Creating extensible java applications*, 2009. 4
- [WMS*13] WANG H., MAI J., SONG Y., WANG C., ZHANG L., TAO F., WANG Q.: A 3d visualization framework for real-time distribution and situation forecast of atmospheric chemical pollution. In *AsiaSim 2013*, Tan G., Yeo G., Turner S., Teo Y., (Eds.), vol. 402 of *Communications in Computer and Information Science*. Springer Berlin Heidelberg, 2013, pp. 415–420. 2
- [WWWa] AccuWeather.com. <http://accuweather.com>. 2
- [WWWb] Aster GDEM. <http://asterweb.jpl.nasa.gov/gdem.asp>. 5
- [WWWc] Cesium WebGL virtual globe and map engine. <http://cesiumjs.org>. 2
- [WWWd] El tiempo.es. <http://www.eltiempo.es>. 2
- [WWWe] Global mapper. <http://bluemarblegeo.com/products/global-mapper.php>. 5
- [WWWf] Home of the Blender project. <http://www.blender.org/>. 5
- [WWWg] MyWeather2 API. <http://www.myweather2.com/developer/>. 2, 8
- [WWWh] OpenWeatherMap API. <http://openweathermap.org/api>. 3, 8
- [WWWi] OSG.js. <http://osgjs.org>. 3
- [WWWj] SceneJS. <http://scenejs.org>. 3
- [WWWk] SeismicPortal. Network of European Research Infrastructures for Earthquake Risk Assessment and Mitigation (European Commission). <http://www.seismicportal.eu/>. 1, 2
- [WWWl] Weather.com. <http://weather.com>. 2
- [WWWm] WorldWeatherOnline API. <http://www.worldweatheronline.com/api/>. 8
- [WWWn] Wunderground API. <http://www.wunderground.com/weather/api>. 2, 8
- [WYMS12] WENKE W., YUMENG G., MIN X., SIKUN L.: Automatic generation of large scale 3d cloud based on weather forecast data. In *Virtual Reality and Visualization (ICVRV), 2012 International Conference on* (Sept 2012), pp. 69–73. 2

Servicio web	Localización	Llamadas	Cantidad de datos	Respuesta	Pronóstico	Precio (en Euros)
weather.com [WWWn]	<ul style="list-style-type: none"> ▪ Estado y ciudad (EEUU) ▪ Código Zip ▪ País y ciudad ▪ Coordenadas geográficas 	500 por día. 10 por minuto.	Alta	3 o 10 días con una única predicción. Primer día cada hora.	XML, JSON	20-600
	<ul style="list-style-type: none"> ▪ IP ▪ Código de aeropuerto ▪ Código zip 					
myweather2.com [WWWg]	<ul style="list-style-type: none"> ▪ Código postal ▪ Coordenadas geográficas ▪ País ▪ Ciudad 	500 por día.	Baja	2 días con un solo pronóstico. 7 días limitado	XML, JSON	36-120
worldweatheronline.com [WWWm]	<ul style="list-style-type: none"> ▪ Código zip ▪ Código postal ▪ Coordenadas geográficas 	500 por hora.	Medium	5 días con un solo pronóstico.	XML, JSON, CSV	Desconocido
	<ul style="list-style-type: none"> ▪ País y ciudad ▪ Código de ciudad ▪ Coordenadas geográficas 	40,000,000 por día. 30,000 por minuto	Media	5 días en intervalos de 3 horas. 14 días con un solo pronóstico.	XML, JSON, HTML	130-1450
openweathermap.com [WWWh]						

Tabla 1: Resumen de características de los principales servicios web atmosféricos disponibles.