Virtual Archaeology Review

Avances en Sistemas de Información Espacial 3D. Aplicaciones en patrimonio y arqueología virtual

Advances in 3D Spatial Information Systems. Applications in cultural heritage and virtual archeology

María Dolores Robles Ortega, Lidia Ortega Alvarado y Francisco Ramón Feito Higueruela

Departamento de Informática, Universidad de Jaén

Resumen

En arqueología son frecuentes las grandes nubes de puntos obtenidas mediante herramientas como radares o escáneres tridimensionales. El excesivo tamaño de estos modelos ocasiona que, en la mayoría de los casos, no puedan ser integrados y manejados de forma adecuada y precisa con otros programas como, por ejemplo, los de gestión del patrimonio. Por ello, es necesario encontrar mecanismos que faciliten el manejo de los datos y optimicen su tratamiento. Por ello, en este trabajo se plantea la integración de OpenVDB y GRASS en un módulo implementado en C++, de forma que se combine la potencia y amplia funcionalidad del sistema de información geográfica GRASS con la eficiencia en el manejo de modelos 3D proporcionada por OpenVDB. En concreto, se propone la aplicación directa del mecanismo implementado para la combinación de la información topográfica de la ciudad con los modelos 3D de los edificios más significativos. Esta aplicación podría resultar de utilidad tanto para ciudades actuales como para la reconstrucción virtual de poblaciones existentes en la antigüedad y actualmente desaparecidas.

Palabras Clave: *VISUALIZACIÓN Y GESTIÓN EFICIENTE, ARQUEOLOGÍA, GRANDES MODELOS, URBANO.*

Abstract

Large point clouds from radars and three-dimensional scanners are commonly used in Archaeology. However, in most cases these models cannot be properly integrated and used in software such as heritage management due to its large size. Therefore, some tools to make this management easier and optimize the processing are needed. In this work, we propose the integration between OpenVDB and GRASS in a C++ module to combine the widen functionality of GRASS GIS with the 3D models management efficiency of OpenVDB. Specifically, this application is used to combine the topographic information of a city with the 3D models of the most significant buildings. This application can be useful for both current cities as well as for virtual reconstruction of existing villages in the olden days and currently disappeared.

Key words: *EFFICIENT VISUALIZATION AND MANAGEMENT, ARCHAEOLOGY, LARGE MODELS, URBAN.*

1. INTRODUCCIÓN

Los Sistemas de Información Geográfica (SIG) están actualmente en un proceso de desarrollo y continua evolución que incrementa las funcionalidades y posibilidades que ofrecen. Así, cada día se desarrollan nuevos módulos que facilitan el manejo de los datos espaciales, incluyendo características adicionales que resultan de utilidad para nuevas aplicaciones y ámbitos de actuación.

La visualización tridimensional es una de las mejoras más destacadas que se están incluyendo en muchos SIG tanto comerciales como de código abierto. En términos geométricos, los métodos existentes para el modelado tridimensional en este tipo de sistemas pueden clasificarse en tres categorías: vectoriales, volumétricos o híbridos (SHEN et al., 2006). En general, la mayoría de los SIG 3D suelen centrarse exclusivamente en el modelo vectorial (WHANG, 2006). Sin embargo, para muchas aplicaciones es necesario también manejar modelos basados en *voxels* que se puedan obtener directamente de la fuente de datos y que permitan representar el volumen de los modelos de una forma más eficiente (Fisher-Gewirtzman et al. 2013). Por ello, en la actualidad algunos autores están empezando a considerar el concepto de modelos híbridos de integración para el manejo conjunto de información *raster* y vectorial (BECKER et al. 2012).

En cualquier caso, un sistema híbrido no debe modificar o cambiar la naturaleza de cada dato sino que debe integrar ambas representaciones bajo un modelo capaz de manejar ambos al mismo tiempo, proporcionando resultados de distinta naturaleza según sea necesario.

Tal y como se ha comentado anteriormente, la gestión de datos vectoriales en los sistemas de información geográfica está ampliamente estudiada y la mayoría de los programas disponibles actualmente ofrecen métodos y herramientas eficientes para el manejo de este tipo de información. No obstante, en el caso de los datos volumétricos, las aplicaciones suelen estar más limitadas. Generalmente, la mayoría de los SIG no manejan de forma eficiente ficheros de modelos tridimensionales con un gran número de vértices y caras debido al gran tamaño de los mismos. Además, no suelen ofrecer una funcionalidad muy extensa, ni tampoco una integración completa de información volumétrica junto con sistemas vectoriales. Por tanto, resulta necesario algún sistema que permita mejorar la eficiencia en la gestión de este tipo de datos en los sistemas de información geográfica, tanto a nivel de visualización como de gestión de información.

Esta aplicación resultaría especialmente útil para estudios arqueológicos puesto que muchos de los dispositivos utilizados en la actualidad obtienen modelos de datos volumétricos de gran tamaño. Así, por ejemplo, los escáneres tridimensionales suelen proporcionar nubes de puntos que, tras ser procesadas, generan modelos 3D con un gran número de vértices y caras. Otras herramientas que también obtienen modelos con una gran cantidad de datos son los escáneres LIDAR o los georradares. Para los casos en los que no fuese posible un manejo interactivo de los ficheros debido a su excesivo tamaño, sería necesario implementar algún algoritmo de simplificación de mallas que reduzca el número de puntos del modelo 3D (CIGNONI, 1997).

La eficiencia en este tipo de aplicaciones es fundamental, ya que normalmente, no se disponen de dispositivos con una alta capacidad de cómputo para realizar el trabajo de campo en los yacimientos arqueológicos. Por ello, es esencial tener en cuenta este requerimiento durante el proceso de desarrollo del software. Otro aspecto importante es la inclusión del programa creado como un módulo de un sistema de información geográfico ya existente. Esto permitirá ampliar las funcionalidades que ofrece el propio SIG con las características específicas de la nueva aplicación. De esta forma, se pueden aprovechar y mejorar funciones ya existentes para implementar el programa, lo que resulta más conveniente que si se desarrollase la aplicación de modo independiente. Una ventaja adicional del uso de la interfaz del SIG es la familiaridad de la

herramienta para los expertos que están acostumbrados a utilizarla. Además, se facilita el proceso de fusión de los datos antiguos con los nuevos, por lo que la utilización de la nueva herramienta no supondría la pérdida de los anteriores ni necesitaría de ningún proceso de adaptación. En este artículo proponemos el diseño y la implementación de un módulo para visualización tridimensional de modelos volumétricos de gran tamaño en el sistema de información espacial de código abierto GRASS junto con capas de información vectorial. En concreto, utilizamos la librería OpenVDB para gestionar la información volumétrica y C++ como lenguaje de desarrollo para integrar dicha librería en un módulo propio de GRASS. Se propone asimismo un ejemplo de aplicación para la visualización del modelo tridimensional de la ciudad de Jaén considerando datos vectoriales como calles y manzanas junto con un modelo volumétrico de la fachada de la catedral.

El resto del artículo se estructura como sigue. En la siguiente sección se detallan los trabajos previos más significativos relacionados con el propósito del artículo. Seguidamente se describe la estructura general de la aplicación y se justifica la utilización del software GRASS, de OpenVDB y de C++ como herramientas de desarrollo. A continuación se describe el procedimiento de integración de dichas tecnologías para generar el módulo de visualización conjunta de información volumétrica y vectorial. Se detalla asimismo una aplicación concreta para el módulo desarrollado: la visualización de datos reales de manzanas y calles de la ciudad de Jaén cercanas a la catedral, que se visualiza incluyendo un modelo 3D de su fachada. Finalmente, se exponen los resultados obtenidos, así como las principales conclusiones y las posibles mejoras que podrían llevarse a cabo en trabajos futuros.

2. TRABAJOS PREVIOS

Los sistemas de información espacial han sido ampliamente utilizados en arqueología virtual con diferentes propósitos. En esta sección expondremos algunos de los trabajos más significativos.

En (LÓPEZ FRAILE et al. 2014) se describe la utilización de un SIG en los estudios microespaciales de yacimientos paleolíticos, consiguiendo una base de datos topográfica y arqueológica asociada. Se usan asimismo modelos 3D previamente escaneados y generados en el formato PDF 3D que son accesibles mediante el SIG a través de un conjunto de fichas. La principal diferencia con nuestra propuesta es que el modelo tridimensional es independiente del SIG puesto que éste simplemente enlaza un fichero con la escena correspondiente. En nuestro trabajo, en cambio, se propone la creación de un módulo propio del SIG que permitirá un control directo de los modelos con los datos asociados y almacenados en el sistema de información geográfica. De esta forma, será posible implementar mecanismos de optimización para la visualización de grandes volúmenes de datos, así como procedimientos de comunicación bidireccional entre el SIG y el módulo de manera que cualquier cambio producido en cualquiera de los dos programas pueda ser transmitido directamente al otro sin necesidad de ningún procedimiento adicional por parte del usuario.

Además del procedimiento de escaneado tridimensional, se han utilizado otras técnicas para generar las escenas tridimensionales asociadas a sistemas de información geográfica como, por ejemplo, la reconstrucción de escenas panorámicas a través de imágenes estéreo (LIN, T. et al. 2008).

Aunque la mayor parte de los trabajos relacionados con SIG aplicados a arqueología proponen sistemas basados en escritorio, existen también otros programas y sistemas de información geográfica orientados a web como PRAGIS (MCCOOL, 2014), que permite el acceso a bases de datos con algunas funcionalidades básicas. En (FABRIZIO et al. 2012) se describe otro ejemplo de este tipo de aplicaciones orientadas a sistemas web. En concreto, se describen los procedimientos y las técnicas digitales utilizadas para crear una infraestructura digital que permite reconstruir, clasificar, gestionar y visualizar los hallazgos arqueológicos

en el ámbito de un repositorio web 3D en el área arqueológica de Pompeya.

Evidentemente, el desarrollo de aplicaciones SIG en páginas web es un campo interesante de investigación para las aplicaciones arqueológicas. Sin embargo, el principal propósito del trabajo que se presenta en este artículo es la visualización eficiente de modelos de gran tamaño, cuya transmisión a través de Internet requeriría un gran ancho de banda. Por ello, nuestro trabajo se orienta a dispositivos de escritorio para evitar el cuello de botella que supondría el envío de los archivos tridimensionales vía web.

Finalmente, otros usos de los sistemas SIG han sido la localización de zonas potenciales de interés para la investigación (CLARKSON et al. 2014) o los relacionados con el modelado espacial de SIG 3D para ciudades (WANG, 2006).

Una vez expuestos los trabajos más significativos relacionados, a continuación describimos la estructura general de la aplicación que proponemos en este artículo para gestionar modelos de gran tamaño en un SIG 3D para su uso en el área de arqueología y visualización de zonas urbanas.

3. ESTRUCTURA GENERAL DE LA APLICACIÓN

En esta sección se expone la estructura general del módulo desarrollado, indicando el procedimiento utilizado para la gestión del flujo de datos en toda la aplicación. Se describen además las posibles alternativas para la implementación tanto para sistemas de información geográfica como para la creación del módulo y la visualización tridimensional.

El esquema general de la aplicación se muestra en la Figura 1. Tal y como se puede observar, el elemento central en el proceso de desarrollo es el módulo propio que se crea para extender la funcionalidad del sistema de información geográfica. Este algoritmo se encargará del manejo de información volumétrica y ficheros ply para posteriormente realizar la visualización tridimensional de los mismos.

El resto de información de entrada (datos vectoriales) será manejada directamente por el SIG. La doble comunicación establecida entre el sistema de información geográfica y el módulo viene determinada por la necesidad de intercambio de datos entre ambas herramientas. Así, el módulo utilizará funciones propias del SIG para manejar las tablas que almacenen los datos vectoriales mientras que el SIG usará los procedimientos creados para la visualización 3D y el manejo de información volumétrica.

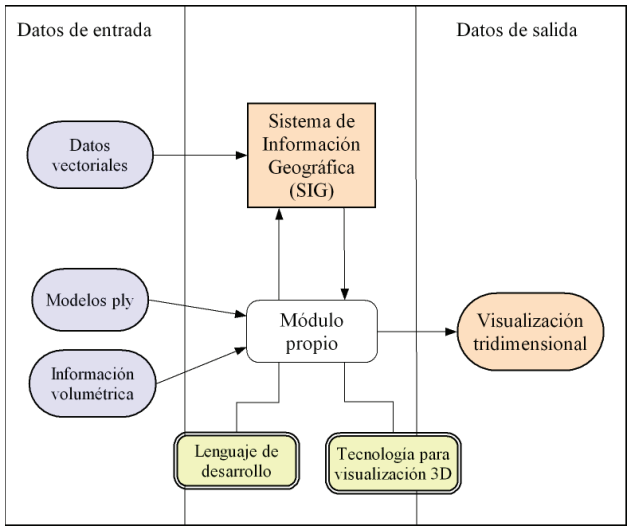

Figura 1. Estructura general de la aplicación

Para implementar esta estructura es necesario determinar previamente qué sistema de información geográfica se va a utilizar, lo que influirá en la posterior elección del lenguaje de desarrollo y la tecnología utilizada para la visualización 3D. A continuación se describen los SIG más usados que podrían servir de base para la aplicación, indicando sus principales ventajas e inconvenientes. Posteriormente, una vez elegido el SIG, se detallarán los posibles lenguajes de desarrollo y de visualización, justificando la elección final de los mismos.

3.1 Sistemas de Información Geográfica

Según la organización NCGIA (*National Center for Geographic Information and Análisis – USA)* un sistema de Información Geográfica (SIG) puede definirse como "un sistema hardware, software y procedimientos elaborado para facilitar la

obtención, gestión, manipulación, análisis, modelado y representación de datos espacialmente referenciados y para la resolución de problemas complejos que impliquen la manipulación y gestión de dichos datos".

En la actualidad existen multitud de programas SIG disponibles, tanto de código abierto como propietarios y que ofrecen una amplia variedad de funciones. En esta sección destacamos aquellos que se han tenido en cuenta para la realización del presente trabajo, exponiendo las ventajas e inconvenientes de cada uno de ellos. Finalmente, se justifica la elección del software que, a nuestro juicio, resulta más adecuado en esta implementación.

Entre los programas SIG existentes, se han valorado principalmente cuatro: MapInfo y ArcGIS (ambos software propietario) y los programas de código abierto Quantum GIS y GRASS. Seguidamente se exponen las características principales de cada uno de ellos:

- MapInfo (http://www.mapinfo.com)

Se trata del software proporcionado por la empresa Pitney Bowes. Soporta un amplio rango de formato de datos espaciales, base de datos relacionales y permite también incluir imágenes aéreas, de satélite y escaneadas en los mapas. Además de la versión original, es posible instalar diferentes programas complementarios que incrementan la funcionalidad como, por ejemplo, Engage 3D que proporciona herramientas analíticas 2D y 3D o Vertical Mapper, que permite mostrar, gestionar e interpretar información espacial continua basada en grid.

Como los datos vectoriales iniciales de la aplicación (información de los cruces y manzanas de la ciudad) se encontraban en formato MapInfo, fue el primer programa considerado. Sin embargo, se descartó su utilización puesto que su lenguaje de desarrollo, MapBasic, es propio y está limitado en la comunicación con otras herramientas como las de visualización. Además, no es multiplataforma y únicamente está disponible para sistemas Windows.

- ArcGIS (http://www.esri.es/es/productos/arcgis)

Al igual que MapInfo, es un software propietario, en este caso proporcionado por ESRI. Se trata de una herramienta que permite almacenar, crear y difundir datos y modelos tridimensionales. Es accesible desde clientes en versión escritorio, navegadores web y terminales móviles.

Dispone de diferentes APIs para desarrolladores para crear aplicaciones web basadas en JavaScript™ y HTML5 para la visualización, edición y análisis de datos vectoriales y *raster* dinámicos.

- GRASS (http://grass.osgeo.org/)

Desarrollado como proyecto oficial de software abierto por *Open Source Geospatial Foundation* (FEITO, 2009), es una de las herramientas SIG más utilizadas. Existen versiones disponibles para Mac OSX, Windows y Linux.

Este programa permite la manipulación de datos geográficos proporcionando un conjunto amplio de operaciones para gestionarlos, bien directamente a través de la propia interfaz de la aplicación o mediante las librerías implementadas en C++ o Python. Gracias a estas últimas, es posible crear un módulo adaptado a las necesidades de cada aplicación que puede utilizar toda la funcionalidad y potencia del SIG, pudiendo además integrarse en el programa original.

El software GRASS incluye alguna funcionalidad básica para el manejo de modelos 3D. Sin embargo, el tratamiento de grandes volúmenes de datos necesario en el área de arqueología virtual puede ocasionar graves problemas de rendimiento, por lo que resulta conveniente utilizar algún otro mecanismo para manejar los datos tridimensionales de una forma más eficiente.

- Quantum GIS (http://www.qgis.org/es/site/)

El software Quantum GIS (QGIS) es un sistema de información geográfica de código abierto, disponible en las siguientes plataformas: Linux, Unix, Mac, OSX, Windows y Android. Soporta numerosos formatos y funcionalidades para datos vectoriales, *raster* y bases de datos. Dispone de una interfaz amigable que facilita su uso tanto para usuarios noveles como para expertos.

En cuanto al lenguaje de desarrollo, proporciona principalmente dos APIs: Python y C++. Incluye asimismo un plugin de GRASS que se utiliza, entre otras cosas, para manejar la información vectorial o en el procedimiento de conversión de un vector a datos *raster*.

Teniendo en cuenta el requerimiento de que la aplicación desarrollada sea multiplataforma, la utilización de MapInfo se descarta inicialmente. A pesar de que ArcGIS es una opción interesante para el desarrollo web, para este trabajo se ha optado por utilizar programas de código abierto.

Por tanto, las dos alternativas posibles son QGIS o GRASS. Aunque el manejo de Quantum GIS es más sencillo, GRASS tiene una mayor potencia y ofrece una funcionalidad más amplia. Además, las funciones de manejo de datos vectoriales de QGIS se obtienen a través de un plugin de GRASS. Por todas estas razones y por ser un programa ampliamente utilizado por la comunidad científica, finalmente se decidió usar GRASS para el desarrollo e implementación del trabajo que se está describiendo en este artículo.

Una vez determinado el sistema de información geográfica que se va a utilizar, el siguiente paso es elegir el lenguaje de desarrollo del módulo así como las librerías para la visualización tridimensional de los modelos.

3.2 Lenguaje de desarrollo

Tal y como se ha comentado anteriormente, es posible crear módulos propios en GRASS mediante dos alternativas: C++ o Python. A continuación se describen los aspectos fundamentales de ambas opciones:

- C++

El lenguaje C++ ha sido utilizado tradicionalmente para optimizar el rendimiento de las aplicaciones, puesto que permite controlar a bajo nivel muchas de las características esenciales para mejorar la eficiencia de un programa. Por ello, podría resultar una opción interesante para el trabajo que se está describiendo.

En el caso de GRASS, es posible acceder desde código C++ a las funciones del SIG mediante las librerías incluidas en la propia distribución. Así, se pueden usar procedimientos internos como si se estuviesen ejecutando en la consola del programa, pero con la diferencia de que los datos obtenidos pueden ser almacenados en variables que podrán ser modificadas posteriormente. Además, también es posible acceder a la estructura interna de las tablas almacenadas tanto para consulta como para actualización, lo que permite la doble comunicación necesaria para nuestra aplicación especificada en la Figura 1.

- Python

Python es un lenguaje de programación interpretado, es decir, está diseñado para ser ejecutado por medio de un intérprete. En la actualidad es uno de los lenguajes con un mayor auge y se está comenzando a utilizar en muchas aplicaciones y desarrollos científicos.

Para poder utilizarlo en GRASS, es necesario instalar previamente las extensiones de Python proporcionadas por el propio SIG. Una vez hecho esto, es posible acceder a las interfaces de las librerías disponibles y a las funciones de las mismas. Se han diseñado asimismo APIs como PyGRASS (ZAMBELLI

et al, 2013) que proporcionan una interfaz para integrar GRASS y Python.

Como los dos lenguajes propuestos permiten el acceso a las funciones del sistema de información geográfica, la decisión de una u otra alternativa dependerá de la posibilidad de incluir tecnologías para la visualización tridimensional de los distintos modelos que manejará la aplicación. Por ello, en la siguiente sección se estudian las principales tecnologías disponibles para realizar este tipo de visualización y su posible integración con estas herramientas y, por tanto, con GRASS.

3.3 Tecnologías para la visualización 3D

Además de una visualización 3D correcta, para esta aplicación es importante que la tecnología que se utilice sea capaz de manejar de forma eficiente una gran cantidad de datos. Así, tal y como se ha comentado anteriormente, en arqueología son frecuentes las grandes nubes de puntos obtenidas mediante herramientas como radares o escáneres tridimensionales. Por tanto, la librería que se utilice deberá optimizar el tratamiento de toda la información, proporcionando una respuesta interactiva al usuario durante el manejo de la aplicación.

Seguidamente se incluyen cuatro lenguajes o librerías que podrían utilizarse para realizar la tarea de visualización en nuestra aplicación:

- X3D (Extensible 3D, http://www.web3d.org/x3d/)

  Desarrollado por el Consorcio Web3D, es el sucesor de VRML. Permite generar contenidos 3D interactivos, tanto estáticos como dinámicos. Está basado en XML y puede utilizarse conjuntamente con tecnologías como Ajax y PHP para el acceso a bases de datos.

  A pesar de ser un estándar, los diferentes visores disponibles pueden mostrar de forma diferente la misma escena debido a diferentes implementaciones del lenguaje original.

- WebGL (http://www.khronos.org/webgl/)

  Permite incluir modelos 3D en páginas web a través de HTML5 sin necesidad de instalar ningún plugin adicional. Se prevé que en un futuro todos los navegadores lo soporten.

  Es una API escrita en JavaScript que permite usar la implementación nativa de OpenGL ES 2.0. No obstante, para facilitar la implementación existen toolkits de desarrollo basados en WebGL como, por ejemplo, X3DOM (http://www.x3dom.org/) o Three.js (http://threejs.org/).

- OpenGL (http://www.opengl.org/)

  OpenGL (*Open Graphics Library*) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que incluyan gráficos 2D y 3D. Fue desarrollada por Silicon Graphics en 1992.

  Muchos de los procedimientos proporcionados por OpenGL efectúan operaciones a bajo nivel, lo que permite controlar y mejorar la eficiencia en las aplicaciones que se desarrollan. Ofrece asimismo una amplia variedad de funciones para mejorar la visualización gráfica de las escenas.

- OpenVDB (http://www.openvdb.org/)

  La librería de código abierto OpenVDB (MUSETH, 2014) proporciona una estructura de datos jerárquica y un conjunto de herramientas para una manipulación eficiente de datos volumétricos discretizados en grids tridimensionales. Permite un almacenamiento compacto con un acceso rápido y eficiente a los datos. Incluye asimismo un conjunto de algoritmos específicamente optimizados para tareas como visualización, filtrado y voxelización.

  Implementada en C++, incluye un visor propio basado en OpenGL para visualizar los modelos gestionados.

Las dos primeras opciones, X3D y WebGL están orientadas específicamente a sistemas web. Por ello, no resultan adecuadas para este trabajo y se descarta su utilización. En cambio, OpenGL y OpenVDB sí que pueden integrarse en aplicaciones de escritorio y podrían, por tanto, ser útiles para nuestro módulo.

Entre las dos alternativas, finalmente se ha elegido OpenVDB puesto que proporciona un almacenamiento eficiente de grandes cantidades de datos volumétricos, siendo éste un aspecto fundamental en la aplicación. Además, al estar basado en OpenGL, será posible también utilizar funciones de esta API en el programa.

Evidentemente, como OpenVDB está implementado en C++, su integración en un programa desarrollado con este lenguaje es inmediata. Existe un módulo OpenVDB para Python que está en proceso de desarrollo y que actualmente ofrece una funcionalidad reducida. Por ello, se ha decidido utilizar el lenguaje C++ junto a OpenVDB y GRASS para desarrollar nuestro módulo para el manejo eficiente de modelos 3D.

4. IMPLEMENTACIÓN DE LA APLICACIÓN

La Figura 2 muestra la estructura general de la aplicación incluyendo las tecnologías y programas elegidos según los criterios descritos previamente. Así, GRASS se utiliza como sistema de información espacial, mientras que OpenVDB se usa para gestionar los datos volumétricos de los ficheros ply obtenidos tras un procedimiento de escaneado tridimensional.

La combinación de estas dos herramientas tiene un amplio rango de aplicaciones en el área de arqueología virtual y patrimonio como, por ejemplo, gestión de yacimientos incluyendo información tridimensional de las piezas encontradas y georreferenciación, asociación de datos geográficos con modelos 3D de interés arqueológico o visualización tridimensional en sistemas de información espacial aplicados a arqueología, entre otras.

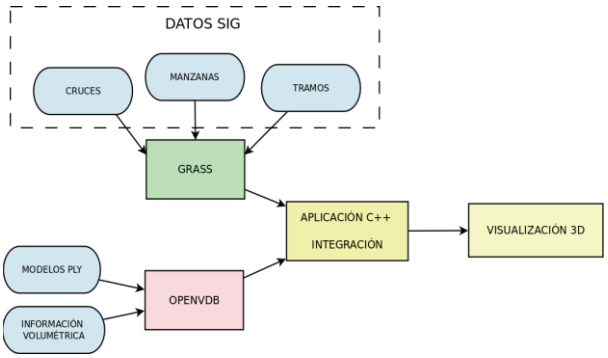

Figura 2. Estructura de la aplicación para visualización urbana

En este trabajo se propone utilizar Grass y OpenVDB para realizar la visualización de una escena urbana (concretamente de la ciudad de Jaén) incluyendo modelos arqueológicos de gran tamaño. Para ello, se ha implementado un módulo en C++ que, usando funciones de ambas herramientas, combina los datos bidimensionales de las calles y manzanas junto con los modelos tridimensionales de los edificios más significativos.

Para la implementación de dicho módulo, se han considerado cuatro tareas o fases fundamentales:

1. Generación de los modelos tridimensionales de manzanas y calles.
2. Cambios en el sistema de referencia geodésico.
3. Importación de los datos de las tablas de GRASS en el programa C++.
4. Generación de los modelos tridimensionales utilizando OpenVDB y visualización de los mismos.

En los siguientes apartados se describen cada una de estas fases, indicando los aspectos más significativos e importantes de las mismas.

4.1 Generación de los modelos tridimensionales de manzanas y calles

Es frecuente disponer para cualquier localidad de información proveniente de un sistema de información geográfica bidimensional o de datos del catastro relativa a manzanas y calles. Estos

datos podrían transformarse en modelos 3D usando el método propuesto en (ROBLES-ORTEGA, 2013). En dicho trabajo se describe el proceso de creación de modelos 2.5D para los edificios a partir del polígono que representa su planta. De esta forma, los modelos tridimensionales de calles y manzanas se obtienen a través de un procedimiento rápido y eficiente.

Sin embargo, el proceso indicado anteriormente está implementado en MapBasic, el lenguaje de programación propio de MapInfo. Por ello, una vez obtenidos los modelos 3D en dicho SIG es necesario realizar un procedimiento de importación de las tablas generadas al sistema de información geográfica GRASS.

Como el formato de MapInfo es compatible con GRASS, la importación se realiza directamente mediante la introducción del comando *v.in.ogr* directamente en la consola de la aplicación. Dicha función convierte capas vectoriales OGR en mapas de vectores de GRASS (NETELER et al., 2008). Por lo tanto, una vez ejecutada con las tablas correspondientes de las manzanas, cruces y calles (tramos), ya se dispone en el software GRASS de los datos almacenados en el formato adecuado.

No obstante, existe un procedimiento adicional que es necesario realizar antes de que el módulo C++ importe los datos de las manzanas, calles y cruces desde GRASS para generar la escena tridimensional: se debe llevar a cabo un cambio en el sistema de referencia geodésico para que coincida con el de los modelos tridimensionales escaneados. Esto es debido a que las manzanas y calles están expresadas en el sistema ED50, mientras que los ficheros escaneados utilizan ETRS89. Entre los dos sistemas, se ha adoptado ETRS89 puesto que actualmente es el sistema oficial.

A continuación se describen brevemente los cambios realizados en el sistema de referencia geodésico oficial en España, indicándose asimismo cómo se ha realizado el proceso del cambio de ED50 a ETRS89 para los ficheros de las manzanas, calles y cruces.

4.2 Cambios en el sistema de referencia geodésico

La figura *natural* de la tierra, excluyendo la topografía o forma externa, se asemeja a la definición de geoide definida como una superficie de nivel equipotencial del campo gravitatorio terrestre. Como la definición matemática del geoide presenta gran complejidad, así como su definición, la superficie de la tierra puede representarse con mucha aproximación mediante un elipsoide de revolución, definiéndose este sistema con una superficie de referencia (sobre la que se definen la latitud y longitud geográficas) y un conjunto de ejes.

El elipsoide de revolución que mejor se adapta al geoide en la zona con un punto donde ambos coinciden o bien la normal a ambos es la solución adoptada, constituyendo el concepto de Sistema Geodésico de Referencia. A lo largo de la historia se han utilizado diversos elipsoides para definir el sistema de referencia de cada país, de tal forma que se define aquel que mejor se ajusta al geoide.

En España se adoptó en 1970 el sistema ED50 (*European Datum 1950*) como sistema oficial, por lo que a partir de dicha fecha se utilizó como referencia para indicar las coordenadas geográficas. No obstante, en 1990, la Subcomisión de la Asociación Internacional de Geodesia (IAG) para el marco de referencia europeo (EUREF), recomendó ETRS89 (*European Terrestrial Reference System*) como sistema de referencia terrestre para Europa, que finalmente se estableció oficialmente en el año 2007 (PÉREZ NAVARRO et al. 2011).

Como consecuencia de estos cambios de referencia, en muchas aplicaciones se trabajan con datos con coordenadas de ambos sistemas. En concreto, en el presente trabajo, los datos de las calles, cruces y manzanas utilizados en GRASS utilizan el sistema ED50 mientras que los datos volumétricos del fichero ply de la fachada de la catedral están expresados en ETRS89. Por ello, es necesario un procedimiento que permita una transformación de las coordenadas geográficas en un mismo

sistema de forma que puedan usarse conjuntamente ambos tipos de datos.

A pesar de que en una primera aproximación se pudiera considerar que dos elipsoides de diferente tamaño y forma situados en dos puntos del espacio distintos se relacionan exactamente mediante expresiones matemáticas (traslaciones, rotaciones y escalados), en realidad no existe una relación perfecta entre los sistemas ED50 y ETRS89. Esto es debido a que la realización del sistema de referencia depende de varios factores: técnicas de observación, método de compensación, equipo humano e instrumental utilizado, etc. En consecuencia, la realización de un *datum* presenta heterogeneidades, más aún en un *datum* clásico como ED50 basado en medidas terrestres junto con algunas espaciales.

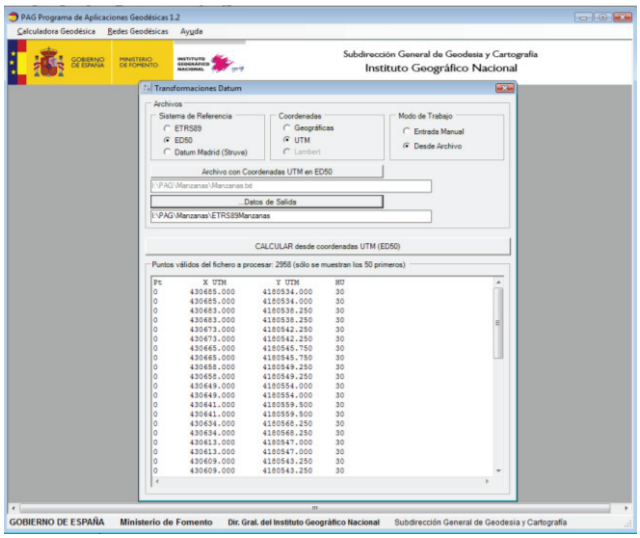

Figura 3. Calculadora geodésica del programa PAG

Por todo lo anterior, era imprescindible determinar un procedimiento de cambio en el sistema de referencia antes de realizar la integración de los datos de entrada disponibles en el módulo C++. Tras realizar varias pruebas con distintas herramientas sin obtener un resultado adecuado, finalmente se optó por utilizar el programa de Aplicaciones Geodésicas (PAG) del Instituto Geográfico Nacional del Ministerio de Fomento, accesible en http://www.ign.es/ign/layoutIn/herramientas.do. Tal y como se puede observar en la Figura 3, dicho programa dispone de una calculadora geodésica que permite transformar puntos o ficheros de puntos entre ED50, ETRS89 y *Datum Madrid (Struve)*. También es posible indicar las coordenadas geográficas o las UTM.

Una vez transformados los datos de las manzanas, calles y cruces al sistema ETRS89 e incluidos nuevamente en GRASS, el siguiente paso consiste en que el módulo C++ los importe para poder usarlos durante el proceso de creación del modelo tridimensional de la escena urbana.

4.3 Importación de los datos de las tablas de GRASS en el programa C++

Las tablas de los datos vectoriales almacenados en GRASS no contienen únicamente información geográfica sino que también incluyen datos alfanuméricos que nos permiten identificar y describir cada elemento de las mismas. Así, por ejemplo, en el caso de las manzanas se almacenan las coordenadas correspondientes al polígono que representa su planta en el plano junto con otros valores como la altura, el número de pisos, la referencia catastral o un código único de identificación, entre otros.

Para poder realizar de forma correcta la importación de toda la información (geográfica o no) en el módulo C++ es necesario conocer previamente la estructura utilizada por GRASS para almacenar los mapas de vectores. Así, éstos se representan utilizando una estructura denominada *arco-nodo*, que consiste en un conjunto de curvas llamadas arcos, que son una serie de pares de coordenadas (x,y,z). Los nodos, por su parte, se crean automáticamente por los extremos de un segmento de arco. Un vector tiene una serie de características (también denominadas primitivas): punto (que puede ser tanto bidimensional como tridimensional), línea (secuencia dirigida de vértices conectados con dos extremos llamados nodos), frontera, centroide, cara y núcleo.

Respecto a la información temática, cada objeto vectorial puede tener asociadas ninguna, una o varias categorías que se identifican por el número de capa. Es posible también que un objeto vectorial tenga múltiples categorías para la misma capa. Cada categoría es un

identificador utilizado para unir la geometría con los atributos que se almacenan en una o varias bases de datos externas. Por tanto, este valor se usa para poder buscar el atributo asociado al objeto del vector. El número de capa determina qué tabla debe usarse para las consultas de los atributos. Por ejemplo, una manzana podría tener asignada en la capa 1 una tabla de atributos conteniendo descripciones de usos mientras que la capa 2 podría contener atributos relativos a los dueños. Cada una de estas capas podría ser además manejada de forma independiente por distintos usuarios.

Una vez especificada la estructura asociada a los datos vectoriales, a continuación se describe el procedimiento seguido en el módulo implementado para acceder a la misma desde las funciones de C++.

Inicialmente, se debe abrir la tabla de datos. Para ello se utiliza la función *Vect_open_old* Posteriormente, se obtiene la información relativa a las bases de datos (usando *Vect_get_field),* abriendo la correspondiente conexión. Tras realizar este proceso, se lee cada tupla de la tabla (que representa un elemento individual como manzanas o cruces) mediante *Vect_read_line*. Esta última función permite obtener también los identificadores de las categorías que se usarán para realizar la consulta a la base de datos y obtener así los diferentes atributos asociados, que podrán almacenarse en variables propias del módulo implementado.

Después de realizar este proceso, ya se dispone de una copia de los datos de GRASS accesible desde el programa C++. A partir de esta información, se generarán los modelos tridimensionales utilizando la librería OpenVDB siguiendo el procedimiento que se describe en la siguiente sección, realizándose finalmente la visualización de los mismos.

4.4 Generación de los modelos tridimensionales utilizando OpenVDB y visualización de los mismos.

La última fase del módulo que estamos presentando en este trabajo consiste en realizar la visualización de los datos geográficos almacenados en GRASS y obtenidos anteriormente (manzanas, cruces y tramos) junto con modelos 3D de los edificios más significativos de la ciudad de interés turístico y arqueológico.

Tal y como se ha comentado anteriormente, el modelado de estos edificios suele realizarse mediante un proceso de escaneado tridimensional, lo que conlleva que los ficheros generados sean de un tamaño excesivo. El manejo de estos archivos por un sistema de información espacial no resulta factible en la mayoría de los casos debido al elevado número de puntos que captura el escáner y que componen el modelo. Por ello, proponemos la utilización de la librería OpenVDB para la manipulación de los datos de forma que la escena final pueda ser optimizada antes de realizar la visualización.

En OpenVDB se distinguen tres objetos esenciales para el manejo de información volumétrica:

- ***Árboles (Tree)***: estructura tridimensional similar a un B-árbol
- ***Transformaciones (Transform)***: relativizan los índices de cada *voxel* (i,j,k) a las localizaciones físicas (x,y,z) en el espacio del mundo
- ***Rejilla (Grid)***: contenedor que asocia a un árbol con sus correspondientes transformaciones y metadatos adicionales.

Por tanto, para representar cualquier objeto en OpenVDB es necesario crear un grid con un árbol y un conjunto de transformaciones asociadas. En nuestro caso, disponemos únicamente de las mallas de triángulos tanto de los modelos ply como de los datos procedentes de GRASS. Sin embargo, gracias a la función **meshToLevelSet** proporcionada por la propia librería podemos transformar dicha malla de triángulos en un volumen (en un grid). Para ello, sólo necesitamos incluir los parámetros de la función: una variable de tipo *Transform* que indique las transformaciones necesarias para establecer la correspondencia entre el mundo

virtual y el real, los vértices y las caras de la malla de triángulos y, finalmente, un valor numérico para indicar la anchura usando *voxels*.

Tras la ejecución de la función, se obtiene el grid asociado a la malla de triángulos, que se almacena para ser visualizado posteriormente junto al resto de grids de todos los elementos que componen la escena (cruces, manzanas, tramos de calle y modelos de monumentos). Para ello se utiliza el visor propio que está incluido en la distribución de la librería.

Es posible almacenar en un archivo externo el grid obtenido tras aplicar la función meshToLevelSet. Para ello, se usaría el procedimiento write de la clase io::File de la librería, que crea un fichero con la extensión vdb. Dichos archivos podrán ser visualizados utilizando el visor externo o bien mediante las instrucciones de la librería para ser manipulados en el módulo de creación propia.

4. RESULTADOS

En esta sección exponemos los resultados obtenidos tras la implementación del módulo descrito a lo largo del artículo. Para ello, se han utilizado los datos reales del catastro correspondientes a la ciudad de Jaén, así como el modelo escaneado de la fachada de la catedral.

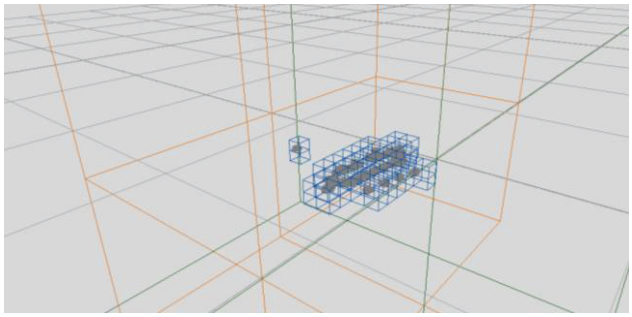

Figura 4. Visualización de las calles y cruces de la ciudad usando OpenVDB

En la Figura 4 se muestran las calles y cruces de la ciudad usando OpenVDB y en la Figura 5 las manzanas. Estas calles y manzanas se han generado mediante el procedimiento descrito en (ROBLES-ORTEGA, et al. 2013) y sus coordenadas han sido transformadas mediante la calculadora geodésica del programa PAG, tal y como se ha indicado en la sección 4.2. Tras seguir el proceso de conversión a OpenVDB expuesto anteriormente, se genera el grid correspondiente que puede ser visualizado y manipulado libremente por el usuario.

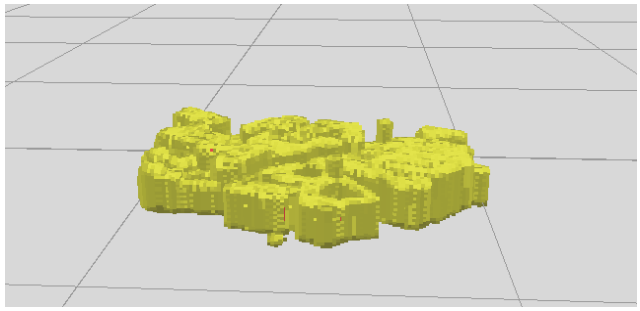

Figura 5. Visualización de las manzanas de la ciudad usando OpenVDB

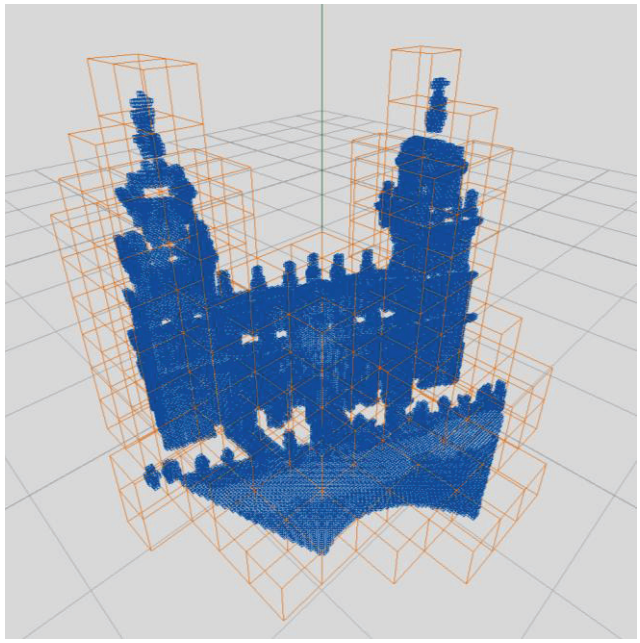

Figura 6. Visualización del fichero escaneado de la catedral de Jaén usando OpenVDB y un tamaño de voxel de 0.001

El principal inconveniente para gestionar los ficheros escaneados de los principales monumentos es su gran tamaño. Así, en el caso de la fachada de la catedral de Jaén, el modelo tiene 5.118.714 vértices y 10.171.246 caras, ocupando el archivo ply un total de 621 MB. En cambio, utilizando la librería OpenVDB dicho archivo se reduce a 35.8 MB si se usa un tamaño

de *voxel* de 0.001, y a sólo 46 KB si se establece un *voxel* de 0.01. Por tanto, la reducción de memoria obtenida es evidente frente a los modelos originales. Gracias a la posibilidad de almacenar los archivos OpenVDB en un fichero propio (de extensión vdb), el tiempo necesario para calcular los modelos voxelizados puede considerarse como preprocesamiento, por lo que no influirá en el rendimiento final de la aplicación. En la Figura 6 se visualiza la fachada de la catedral con un tamaño de *voxel* de 0.001.

Finalmente, las Figuras 7 y 8 muestran imágenes de la aplicación en las que se visualizan simultáneamente tanto los modelos de las calles, cruces y manzanas generados con GRASS como el archivo de la fachada de la catedral de Jaén. Como se puede observar, las transformaciones de las coordenadas geodésicas son correctas, puesto que todos los modelos aparecen en la posición adecuada. En este caso se ha utilizado un tamaño de *voxel* de 0.01, estando activos en la escena global un total de 217.227 *voxels*.

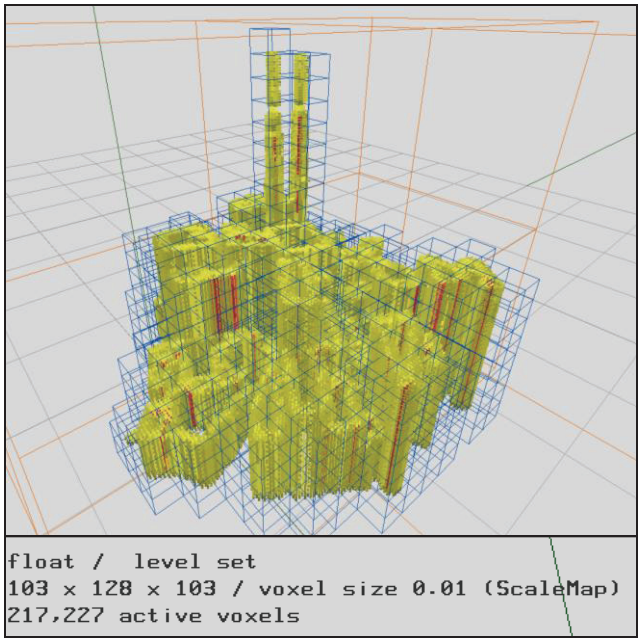

Figura 7. Visualización del fichero escaneado de la catedral de Jaén y de las manzanas y calles usando OpenVDB en el modo de voxelización

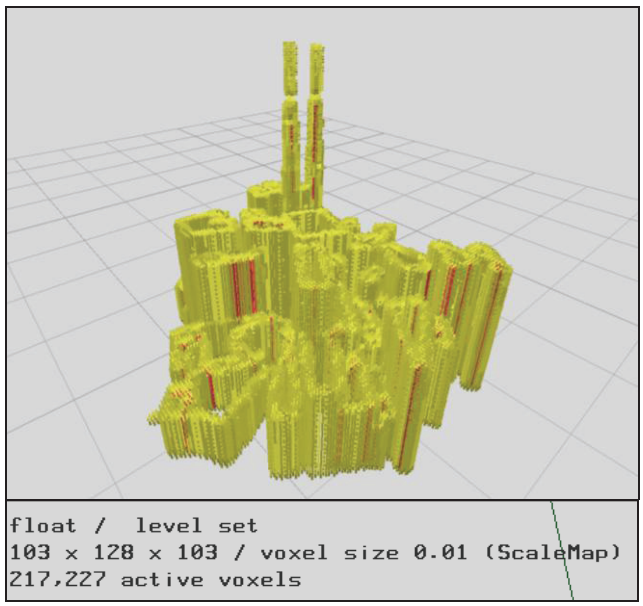

Figura 8. Visualización del fichero escaneado de la catedral de Jaén y de las manzanas y calles usando OpenVDB

La representación volumétrica, además de la reducción en el tamaño de los archivos, puede aportar otras ventajas en aplicaciones relacionadas con el ámbito de la arqueología. Así, la eficiencia en las operaciones booleanas en el manejo de modelos 3D puede mejorar el rendimiento de los programas que las utilicen. Por ejemplo, en el caso de representaciones de piezas tridimensionales en los que sea necesario realizar cortes transversales, la voxelización permitirá ejecutar dichas operaciones de una forma más eficiente que si se llevara a cabo directamente sobre el modelo 3D.

5. CONCLUSIONES Y TRABAJOS FUTUROS

En este artículo se ha descrito el trabajo desarrollado para realizar visualizaciones volumétricas en sistemas de información espaciales de una forma eficiente, así como su aplicación en trabajos arqueológicos. En concreto, se ha creado un módulo en GRASS que, a partir de capas bidimensionales de información geográfica de la ciudad (manzanas, polígonos de cruces y de calles), genera una visualización tridimensional. Además, el software generado permite la inclusión de

archivos 3D de modelos arqueológicos generados previamente mediante cualquier otro tipo de herramientas como, por ejemplo, un escáner tridimensional.

Además de la mejora de la visualización incluyendo texturas, en futuros trabajos se pretende desarrollar la inclusión de opciones adicionales del manejo de los datos con el objetivo de favorecer la interactividad con las escenas creadas. Esto permitiría enriquecer la aplicación generada incluyendo nuevas funcionalidades que podrían ser de utilidad en ámbitos arqueológicos.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente subvencionado por la Universidad de Jaén bajo el proyecto de investigación "Gestión del Subsuelo Urbano mediante SIG 3D" - Centro de Estudios Avanzados en TIC.

BIBLIOGRAFIA

BECKER, S. et al. (2012): "Integrated management of heterogeneous geodata with a hybrid 3D geoinformation system". *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*. I-2, pp. 87-92.

CIGNONI, P. et al. (1997): "A Comparison of Mesh Simplification Algorithms". *Computers & Graphics*. Volumen 22, pp. 37-54.

CLARKSON, C. et al. (2014): "Mapping stone: using GIS spatial modelling to predict lithic source zones". *Journal of Archaeological Science*. Volumen 46, pp. 324 – 333.

FABRIZIO, I.A. et al. (2012): "3D reality-based artefact models for the management of archaeological sites using 3D Gis: a framework starting from the case study of the Pompeii Archaeological area", *Journal of Archaeological Science*. Volumen 39, nº 5, pp. 1271-1287, doi: http://dx.doi.org/10.1016/j.jas.2011.12.034.

FEITO, F.R., SEGURA, R.J. (2009): "Herramientas SIG 3D". I Congreso Internacional de Arqueología e Informática Gráfica, Patrimonio e Innovación.

FISHER-GEWIRTZMAN, D. et al. (2013): "Voxel based volumetric visibility analysis of urban environments". *Survey Review*. Volumen 45, nº 333, pp. 451-461.

LIN, T. et al. (2008): Development of a virtual reality GIS using stereo vision, Computers and Electronics in Agriculture. Volumen 63, nº 1, pp. 38-48.

LÓPEZ-FRAILE, F.J. et al. (2014): "Aplicaciones SIG en la caracterización geoarqueológica del yacimiento paleolítico de Las Delicias (Madrid, España) y visualización en 3D de los resultados", en *Virtual Archaeology Review*. Volumen 5, nº 10, pp. 32-44.

McCOOL, J.P. (2014): PRAGIS: a test case for a web-based archaeological GIS. Journal of Archaeological Science. Volumen 41, pp. 133-139.

MUSETH, K. (2014): *Hierarchical Digital Differential Analyzer for Efficient Ray-Marching in OpenVDB*. ACM SIGGRAPH Talk.

NETELER, M. et al. (2008): *Open source GIS: A GRASS GIS Approach*. 3rd Edition. Springer. ISBN-13: 978-0-387-35767-6. Book Series: The International Series in Engineering and Computer Science: Volume 773.

PÉREZ NAVARRO, A et al. (2011): *Introducción a los sistemas de información geográfica y geotelemática*. Editorial UOC.

ROBLES-ORTEGA, M.D. et al. (2013): "Automatic Street Surface Modeling for Web-Based Urban Information Systems". *Journal of Urban Planning and Development*. Volumen 139, nº 1, pp. 40-48.

SHEN, DY. et al. (2006): "3D simulation of soft geo-objects". *International Journal of Geographical Information Science*. Volumen 20, nº 3, pp. 261-271.

WANG, Y. (2006): 3D GIS Spatial Modeling for City Surface and Subsurface Integration. IGARSS'06.

ZAMBELLI, P. et al. (2013): Pygrass: An Object Oriented Python Application Programming Interface (API) for Geographic Resources Analysis Support System (GRASS) Geographic Information System (GIS). *ISPRS International Journal of Geo-Information* 2, pp. 201–219.