Master en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital
UNIVERSIDAD POLITÉCNICA DE VALENCIA

TÉCNICAS AVANZADAS PARA VISUALIZACIÓN Y USO DE SIMBOLOGÍA EN UN GLOBO VIRTUAL

- Trabajo de Final de Master -

Autor:
Jordi Torres Fabra
jtorres@ai2.upv.es

Director:
Javier Lluch Crespo
jlluch@ai2.upv.es

Índice general

1

Introducción

Durante los últimos años, el mundo ha vivido una revolución en la manera en que los ciudadanos hacen uso de las tecnologías de información geográfica y 3D. Los satélites de observación de la Tierra, Internet, los dispositivos móviles y las tecnologías 3D y de código abierto han universalizado el acceso a esta información, rompiendo las fronteras entre el mundo físico y el virtual.

En particular, la cartografía digital está experimentando grandes avances debido a la popularización de los sistemas de información geográfica (SIG), cada vez más utilizados por las administraciones públicas y con fines más variados de los que se había venido utilizando tradicionalmente. El campo de aplicación de los SIG se ha ampliado tanto que se pueden utilizar en cualquier ámbito donde esté presente la información geográfica y la gestión, como por ejemplo: catastro, cálculo de rutas, información de carreteras, control de flotas y transporte, diseño y gestión de redes (aguas, eléctricas, gas, telecomunicaciones y cualquier tipo de conducciones), gestión de recursos hidrológicos, geomorfología, marketing, epidemiología, y otras muchas más.

Por otro lado, los avances en el campo de visualización tridimensional, tanto en hardware como en software, han permitido que las imágenes, obtenidas mediante síntesis digital, tengan cada vez un mayor realismo, así como el desarrollo de nuevas técnicas como la realidad virtual, la realidad aumentada y visualización en dispositivos móviles, entre otras, las cuales podrían ser aplicadas en los SIG para obtener

sistemas mucho más versátiles que los actuales.

Sin embargo, estos avances en informática gráfica no se ven plasmados en los SIG. La mayor parte de las veces estos sistemas presentan ortofotos superpuestas sobre un modelo digital del terreno, cuyos resultados no son muy realistas. La mayor parte de las herramientas GIS incorporan algún tipo de visualización 3D, pero generalmente muy básicas. Normalmente, existe la posibilidad de realizar alguna imagen fija con un modelo de cámara perspectiva, posibilidad de efectos de niebla y animación de sobrevuelos de modelos digitales del terreno. Todas estas funciones se ofrecen en los SIG de forma accesoria ya que la funcionalidad principal de la aplicación es el análisis geográfico.

Para subsanar esta falta de funcionalidades se desarrolló gvSIG 3D, un proyecto que daba a los usuarios la capacidad de manejar información georreferencida en tres dimensiones sobre el sofware de código libre gvSIG. Sin embargo las restricciones impuestas para que el software funcionara en la mayoría de equipos disponibles en las administraciones no permitían hacer uso de los últimos avances en informática gráfica, ya que muchos de estos equipos no disponían del hardware necesario para manejar con soltura escenarios 3D complejos.

En este contexto se ha desarrollado el proyecto CENIT España Virtual, con el objetivo de aumentar la capacidad de análisis, gestión y visualización de información geográfica a través del 3D utilizando tecnología puntera. En este proyecto han participado importantes empresas y centros de investigación así como administraciones públicas como el Centro Nacional de Información Geográfica (IGN/CNIG).

En este sentido el Instituto de Automática e Informática Industrial (ai2) de la Universidad Politécnica de Valencia ha sido el organismo público de investigación (OPI) de la empresa Prodevelop, que ha participado en este proyecto. El trabajo de fin de master que se presenta ha sido desarrollado gracias a los fondos obtenidos a través de esta relación.

A continuación se desgranarán los objetivos del proyecto y se presentará la estructura del presente documento.

1.1. Objetivos

El objetivo principal del trabajo es dar un herramienta para que los grupos de investigación y empresas involucrados en el proyecto España Virtual pudieran visualizar los resultados de sus investigaciones, así como mejorar en la medida de lo

posible la funcionalidad de gvSIG 3D.

La representación de datos vectoriales tridimensionales haciendo uso de simbología 3D avanzada era una de las partes más demandadas por los miembros del proyecto, por tanto uno de los objetivos más importantes a llevar a cabo. También se ha demandado la posibilidad de mostrar edificios y núcleos urbanos, a ser posible utilizando estándares.

Por otro lado se pretendía avanzar en el campo de la realidad virtual y maximizar la experiencia inmersiva de los usuarios en este tipo de ambientes. Por tanto otro de los objetivos parciales ha sido dotar a gvSIG 3D de capacidades estereoscópicas y comprobar qué sistemas son los que mayor sensación de inmersión provocan en los usuarios.

1.2. Estructura de la memoria

El documento está estructurado de la siguiente manera. Primero, en el capítulo de antecedentes se explica el concepto de grafo de escena poniendo especial atención a OpenSceneGraph. En este mismo capítulo se introducen los Sistemas de Información Geográfica, estudiando sus características y funcionalidades. En especial se introduce osgVirtualPlanets y gvSIG 3D. La última parte del capítulo de antecedentes va dedicada al proyecto CENIT España Virtual.

En los siguientes capítulos se detallan las funcionalidades desarrolladas, comentando aquellas partes más importantes tanto del análisis como de la implementación. Después se muestran los resultados, dando especial importancia a aquellas funcionalidades que han sido utilizadas por otros grupos de investigación. En siguiente lugar se comenta el trabajo de difusión que se ha llevado a cabo durante la ejecución del proyecto. Por último en el capítulo de conclusiones se diserta sobre los beneficios obtenidos a partir del trabajo llevado a cabo.

Se debe tener en cuenta que a medida que transcurre la memoria se van mostrando resultados parciales de la implementación, porque los verdaderos resultados son aquellos que cumplen con el objetivo principal, es decir servir como plataforma de visualización a otros grupos de investigación. Esto es lo que se presenta en la sección de resultados.

2

Antecedentes

En este apartado se explican las tecnologías implicadas en el desarrollo de este trabajo de fin de máster así como los proyectos relacionados con él. En primer lugar se hace una descripción del concepto de grafo de escena, pasando a comentar los detalles sobre el grafo que se ha utilizado en la implementación: OpenSceneGraph. Después se muestra una breve introducción al mundo de los sistemas de información geográfica, para dar paso a los proyectos y librerías que se han visto implicados en la realización del trabajo: osgVirtual Planets y gvSIG 3D. Por último se describe el proyecto CENIT España Virtual, marco del trabajo de fin de máster.

2.1. Grafos de Escena

Las limitaciones hardware han acompañado siempre a la informática gráfica debido al ingente número de operaciones en coma flotante que se deben realizar. En un principio estos cálculos se hacían en el procesador principal, aunque hoy en día casi todos los equipos informáticos incorporan una tarjeta gráfica con un procesador especializado que libera al procesador principal de esta sobrecarga.

Aún así el cuello de botella de un sistema gráfico suele ser el número de operaciones por segundo que puede realizar una tarjeta gráfica.

Para reducir el número de cálculos se debe determinar la visibilidad en la escena

que se desea representar, osea diferenciar lo que es visible de lo que no. Así evitaremos dibujar geometría redundante. A priori hay dos técnicas para conseguirlo:

- Determinar que objetos quedan fuera de la pirámide de visualización (view-frustrum culling).
- Determinar que objetos están ocultos por otros (occlusion culling) y eliminar las caras traseras de las geometrías (back-face culling) .

Aunque existen algoritmos que resuelven sendos problemas, éstos tienen un coste lineal O(n) con el número de primitivas, por lo que determinar la visibilidad en escenas complejas puede ser un proceso muy costoso.

La idea básica para incrementar la eficiencia de estos algoritmos es eliminar rápidamente porciones grandes de la escena que no serán visibles en la imagen final. No se pretende una solución exacta, como se obtiene con el Z-buffer y el clipping, sino una manera rápida de realizar una estimación conservativa de las primitivas que serán visibles. Esta estimación recibe el nombre de conjunto potencialmente visible (potentially visible set).

La organización jerárquica de escenas nos permite conseguir un coste logarítmico O(log n) en el mejor de los casos. Se organiza la geometría en dos o tres dimensiones en torno a una estructura de datos espaciales.

A parte de la determinación de la visibilidad estas estructuras actúan en diversos ámbitos de la aplicación como el cálculo de intersecciones, detección de colisiones, aceleración de la iluminación global, etc.

El concepto de grafo de escena parte de un grafo acíclico dirigido (DAG). Comienza con un nodo raíz, que contiene todo el mundo virtual, ya sea 2D o 3D. El mundo se distribuye en una jerarquía de nodos que representan las agrupaciones espaciales, los ajustes de la posición, las animaciones de los objetos o las definiciones de las relaciones lógicas entre ellos. Las hojas del grafo representan los objetos físicos, es decir las geometrías dibujables en caso de una escena tridimensional y sus características materiales. Un nodo del grafo puede tener varios padres.

El grafo de escena no es una completa especificación de un motor gráfico. El hecho de que el grafo no integre toda la aplicación, permite la interoperabilidad con otros componentes, siendo una potente herramienta para distintas áreas. Este factor permite que este sistema sea capaz de implementar la mayoría de los algoritmos conocidos de aceleración de la visibilidad.

2.1.1. OpenSceneGraph

OpenSceneGraph es un conjunto de librerías que proveen de organización de escenas y optimización del renderizado a aplicaciones tridimensionales. Esta escrita en ANSI C++ y hace uso del estándar de la industria OpenGL como API de bajo nivel. La mayor parte de OSG opera independientemente del sistema nativo de ventanas. Como resultado OSG es multiplataforma, pudiendo ejecutarse en la mayoría de sistemas operativos.

OSG es de código abierto y está licenciado bajo la licencia GPL, esto reporta muchos beneficios. De hecho OpenSceneGraph es uno de los grafos de escena disponibles de mayor rendimiento y el estándar para grafos de escena de OpenGL.

Una de las características de OSG es su arquitectura modular que permite añadir elementos y generar nuevos módulos y plugins para ser empleados en el grafo de escena permitiendo al programador extender la librería según sus necesidades. Si estudiamos su estructura, OSG está formada por una serie de pequeñas librerías y plugins. Ambos extienden la funcionalidad, bien añadiendo efectos y patrones gráficos que se pueden incluir directamente en el grafo de escena, o bien añadiendo la posibilidad de trabajar con tipos de archivos. Generalmente estos plugins requieren de librerías externas independientes de OSG que se enlazan dinámicamente con el programa en ejecución.

OSG además se ha diseñado para cargar las librerías y plugins bajo demanda. Esto supone que un programa que emplee OSG, únicamente cargará las librerías básicas y, dependiendo de las necesidades, el resto de librerías y plugins serán enlazadas dinámicamente cuando el usuario lo requiera. Esta característica permite ahorrar memoria durante la ejecución de un programa al no tener que cargar los módulos que no se utilicen.

2.2. Sistemas de Información Geográfica

Un Sistema de Información Geográfica (SIG o GIS, en su acrónimo inglés) es una integración organizada de hardware, software, datos geográficos y procedimientos diseñados para soportar la captura, administración, manipulación, análisis, modelamiento y representación gráfica de datos u objetos referenciados espacialmente con el fin de resolver problemas complejos de planificación y gestión.

También podríamos definirlo como un modelo de una parte de la realidad referido

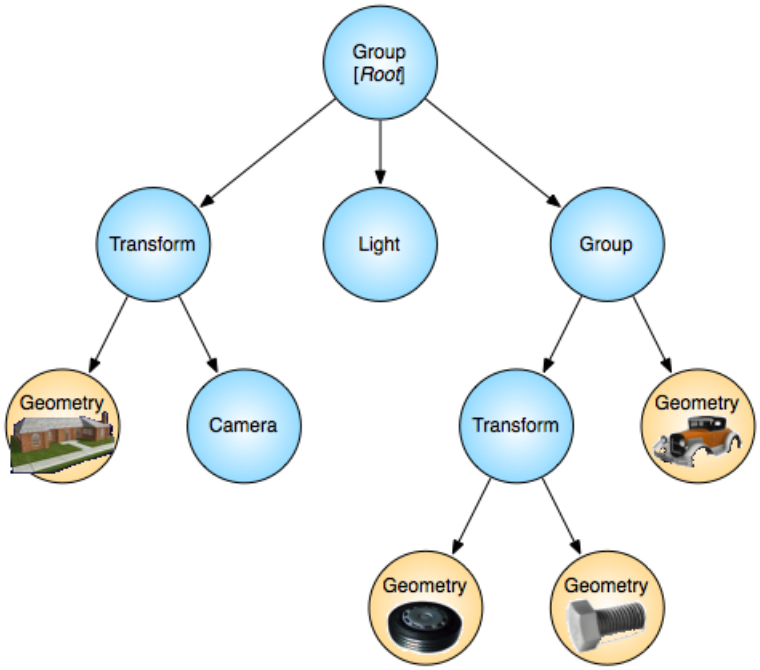

Figura 2.1: Ejemplo de organización jerárquica en un grafo de escena

a un sistema de coordenadas terrestre y construido para satisfacer unas necesidades concretas de información.

El SIG funciona como una base de datos con información geográfica (datos alfanuméricos) que se encuentra asociada por un identificador común a los objetos gráficos de un mapa digital. De esta forma, señalando un objeto se conocen sus atributos e, inversamente, preguntando por un registro de la base de datos se puede saber su localización en la cartografía.

El SIG separa la información en diferentes capas temáticas y las almacena independientemente, permitiendo trabajar con ellas de manera rápida y sencilla, y facilitando al profesional la posibilidad de relacionar la información existente a través de la topología de los objetos, con el fin de generar otra nueva que no podríamos obtener de otra forma. Esto nos permite analizar patrones, relaciones y tendencias en la información de modo que podamos tomar mejores decisiones respecto a un problema concreto.

Existen diversas formas de modelizar estas relaciones entre los objetos geográficos. Dependiendo de la forma en que ello se lleve a cabo se diferencian dos grupos principales: formato raster(imágenes) y formato vectorial.

2.2.1. osgVirtualPlanets

Con la finalidad de permitir incorporar una vista tridimensional a los clientes SIG, se desarrolló una librería que se denomina osgVirtualPlanets (osgVP) basada en el grafo de escena OSG e implementada en C++. Esta librería incorpora una API en Java utilizando la tecnología *Java Native Interface* (JNI) que permite ser integrada con la mayoría de aplicaciones SIG, que están implementadas en este lenguaje. Esta librería ha sido implementada por el Instituto Universitario ai2.

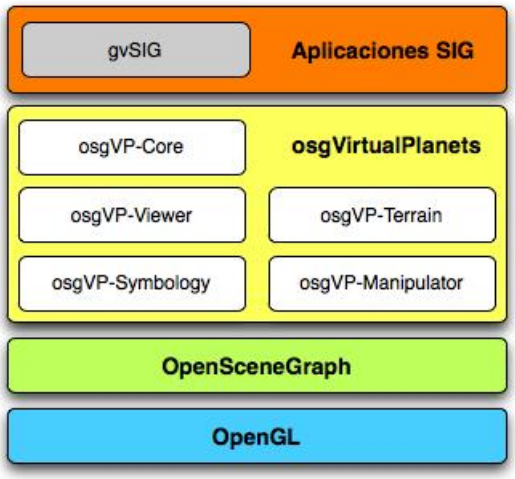

Figura 2.2: Arquitectura de cliente SIG sobre *osgVirtualPlanets*

Los componentes de osgVirtualPlanets son:

- **osgVP-Viewer**: permite añadir un contexto 3D a cualquier aplicación para renderizar escenas creadas mediante la librería o utilizando los nodos de OSG.
- **osgVP-Terrain**: genera y maneja modelos de terreno a partir de datos de elevación e imágenes raster en formato SIG.
- **osgVP-Symbology**: ofrece los mecanismos necesarios para visualizar datos vectoriales y símbolos asociados, así como para cambiar sus propiedades.
- **osgVP-Manipulator**: permite manipular objetos de la escena como por ejemplo, datos vectoriales.
- **osgVP-Core**: es el componente principal de la librería.

La implementación de osgVP ha sufrido varias iteraciones en su desarrollo, tanto en su arquitectura interna como en el sistema de building. De hecho para llevar a cabo los objetivos planteados ha sido necesario rediseñar gran parte de la librería, en especial la parte de simbología.

2.2.2. gvSIG 3D

gvSIG 3D es una extensión del software GIS gvSIG, financiado por la Generalitat Valenciana en sus inicios pero que ha crecido exponencialmente. Tanto es así que hoy en día es un de los proyectos GIS de código libre con mayor importancia a nivel internacional. La extensión 3D para gvSIG fue implementada por la empresa Iver T.I. en colaboración con el Instituto Universitario ai2 de la UPV.

gvSIG 3D consta de dos partes bien diferenciadas:

- El núcleo o motor gráfico de gvSIG 3D que es la librería osgVP.
- La extensión que sirve como puente entre gvSIG y osgVP.

Como una de las partes principales de este trabajo ha sido la creación de un framework para simbología tridimensional, se procede a comentar cómo está organizada esta simbología en la vista 2D para después poder extrapolar este diseño al 3D en la medida de lo posible.

Representación de la información vectorial en gvSIG

Las datos con información geográfica de tipo vectorial constan de parte alfanumérica y geometrías. Ambas hacen referencia a los datos pero no a su representación. Esta representación está definida en capas más altas pero no forma parte de los propios datos y dependerá del contexto en el que sea usada la información que queramos representarla de una u otra forma.

Como tipos de objetos a representar se eligen una agrupación por tipo de representación. Esto es: geometrías y etiquetas. Las geometrías pueden ser de tres tipos, puntos, lineas y polígonos. Una vez agrupadas se ha tenido que definir el método de representación para cada una.

Las etiquetas pueden ser los datos alfanuméricos contenidos en la tabla asociada o textos definidos por un usuario. Se ha tenido que considerar para su representación

la posibilidad de seleccionar el campo de la tabla y las características del texto (altura, tamaño, color, unidades, rotación, etc...), así como otras posibilidades de ubicación o visualización dependiendo de la escala a la que se encuentre el mapa. Como capacidades añadidas se ha considerado útil la posibilidad de aplicar condiciones por las cuales una etiqueta se mostrará o se ocultará en un momento dado. En estas condiciones puede intervenir los valores de los campos alfanuméricos. Por ejemplo, podríamos de esta forma etiquetar en un mapa de poblaciones de la península solo aquellas con más de 100000 habitantes.

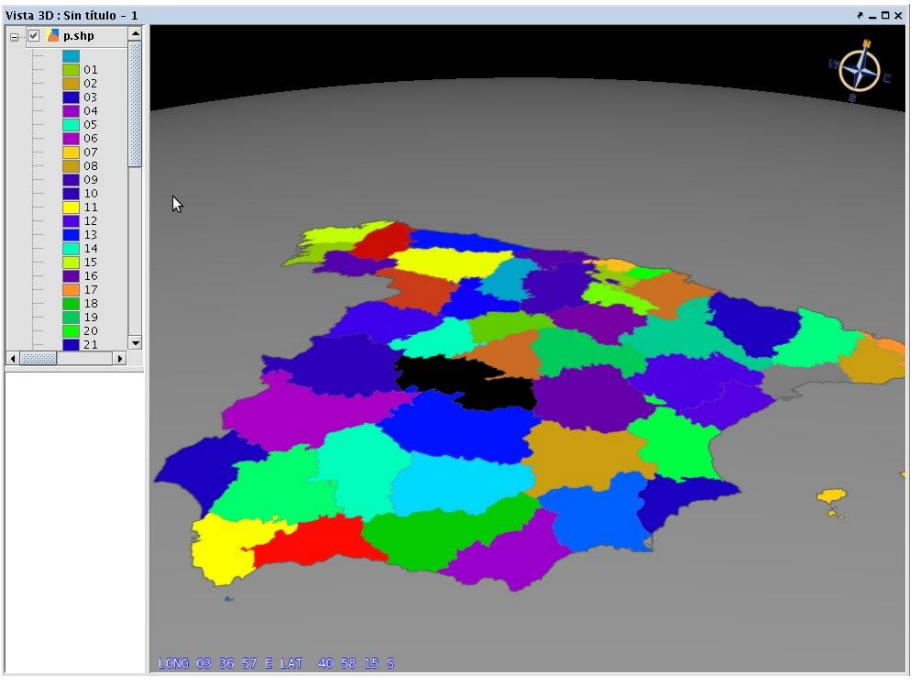

Figura 2.3: Simbologia vectorial aplicada a una capa en gvSIG 3D

Para representación de geometrías se aplicarán leyendas mediante un método de asignación cómodo para cualquier escenario en que nos encontremos. En este caso se usan una serie de categorías.

Categorías de leyendas

- **Leyendas de Valores Únicos:** Es la implementación concreta e inmediata del tipo Leyenda Vectorial de Valores Únicos (*IVectorialUniqueValueLegend*). Representa una clasificación de símbolos de acuerdo a un valor. El símbolo aplicado a una

Feature es el que está asociado al valor del atributo de clasificación de dicha *Feature*. Así, esta leyenda contiene tantos símbolos como valores se definan en la tabla de atributos para un determinado atributo.

- **Leyenda de Símbolos proporcionales:** Es la leyenda en donde, en base al valor de un campo, el sistema es capaz de generar símbolos lineales en capas de líneas, y puntuales en capas de puntos y polígonos (para las cuales se toma como punto el centroide del polígono). Es un único símbolo que se representa con un tamaño definido por un atributo de la *feature*. El tamaño del símbolo es proporcional al valor de dicho campo. Esta leyenda contiene un único tipo de símbolo.

- **Leyendas de Símbolos Graduados:** Es la leyenda en donde, en base al valor de un campo, el sistema es capaz de generar símbolos lineales en capas de líneas, y puntuales en capas de puntos y polígonos (para las cuales se toma como punto el centroide del polígono). Es un único símbolo en donde con un tamaño definido para cada intervalo. El tamaño del símbolo es proporcional al valor de dicho campo. Esta leyenda contiene un tipo de símbolo para cada intervalo y uno o más intervalos definidos.

- **Leyendas de Intervalos:** Es la implementación concreta e inmediata del tipo Leyenda Vectorial de Intervalos (*IVectorialIntervalLegend*). Representa una clasificación de símbolos de acuerdo a un valor. El símbolo aplicado a una Feature es el que está asociado al intervalo que contiene el valor del atributo de clasificación de la Feature. Así, esta leyenda contiene un conjunto de símbolos con tantos símbolos como intervalos se hayan definido.

- **Leyenda multivariable:** Se presenta como una leyenda que a su vez es una composición de una leyendas de valores únicos y/o por intervalos. Una por cada variable.

- **Leyenda de gráficos:** Permite componer una leyenda multivariable a través de una única leyenda que muestra símbolos multivariable como los símbolos de tipo *chart*. En este caso, la leyenda requiere del soporte que dan este tipo de símbolos.

- **Leyenda de densidad de puntos:** Es una especialización de una leyenda de valores únicos. La diferencia se encuentra en que se apoya en un símbolo de densidad de puntos combinado con un símbolo de relleno simple a través de un *MultiLayerFillSymbol* para crear la leyenda.

La leyenda es la encargada de situar en el mapa cada uno de los símbolos que lo componen. Una leyenda gestiona uno o más símbolos, según la necesidad para la cual se haya diseñado, no necesariamente iguales. Se pueden usar símbolos de imagen junto con caracteres en la misma leyenda.

En síntesis, una leyenda contiene uno o más *ISymbols* y implementa unas reglas. Toma *Features* desde un origen de datos y según sus reglas decidirá qué símbolo aplica a cada *feature*. Una vez se sabe qué símbolo aplica a dicha feature la leyenda hace que el símbolo la represente gráficamente.

La leyenda es, en sí, la regla que decide qué símbolo se usa. De esta manera, para cada modalidad de mapa que requiera una representación basada en unas reglas concretas, existirá una implementación de *Legend* encargada de plasmar dichas reglas. La aplicación de las reglas que implementa cada tipo de leyenda sobre todas las features de la capa sobre la vista da como resultado un mapa renderizado.

Clasificación de leyendas

Las leyenda se pueden clasificar según dos criterios.

- Según su origen de datos:

  Si la capa a la que pertenece la leyenda tiene un origen de datos vectorial. Si una capa es vectorial entonces sus features son formas geométricas asociadas a un registro de una tabla que contiene sus propiedades. Las propiedades pueden ser de cualquier tipo de datos soportado por el origen de datos.

  Si la capa a la que pertenece la leyenda tiene un origen de datos ráster. Si una capa es ráster entonces sus *features* son celdas de una cuadrícula uniforme que cubre todo el área de la capa. Estas celdas representan propiedades del área que cubren. Las celdas contienen uno o varios valores numéricos, uno para cada propiedad. Las propiedades de una capa ráster se conocen también como bandas del ráster.

- Según si está clasificada:

  No clasificada: Una leyenda no está clasificada si su único objetivo es mostrar las features en pantalla.

  Clasificada: Una leyenda está clasificada si sus propiedades están representadas de acuerdo con unas reglas.

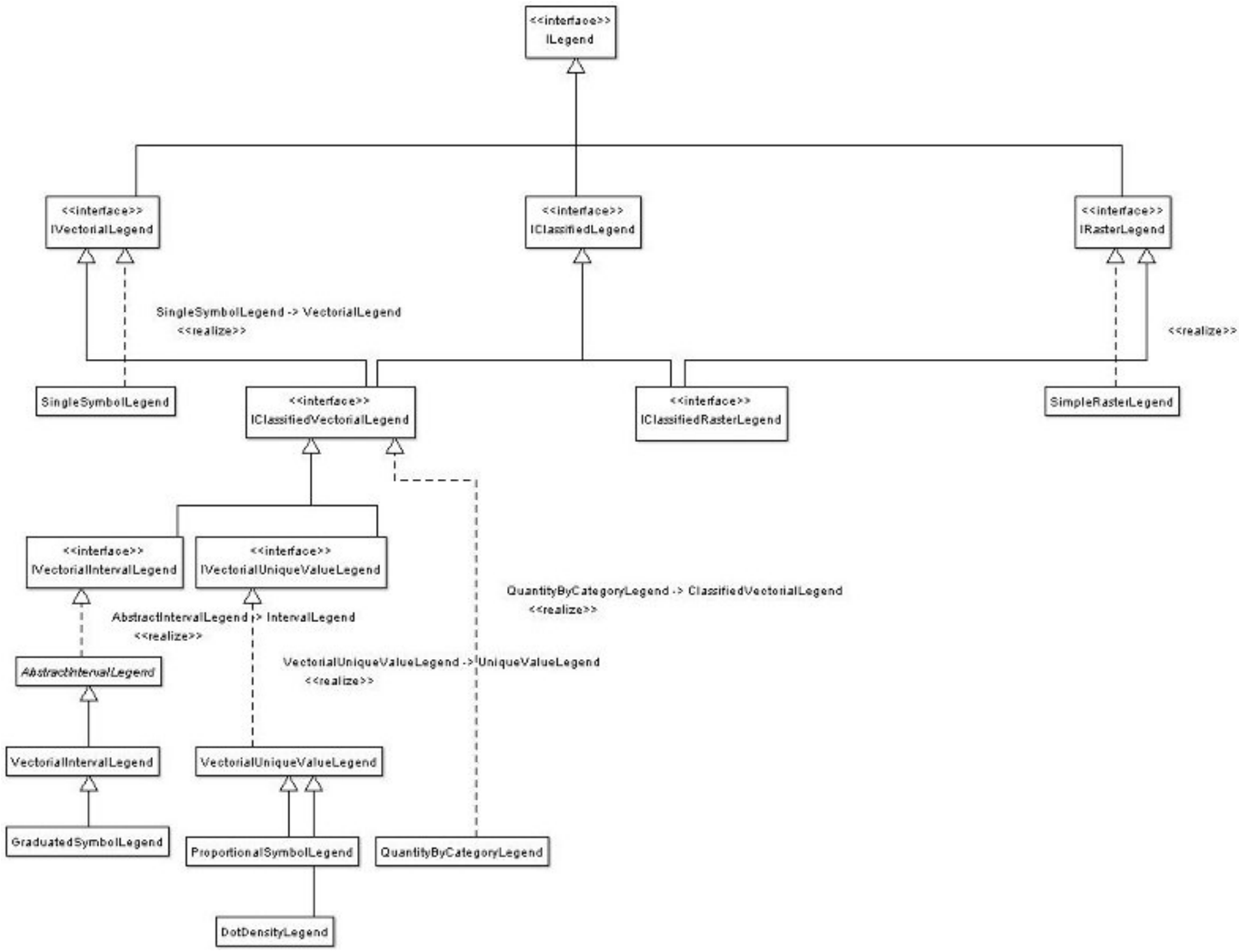

Figura 2.4: Diagrama de clases de las leyendas vectoriales

2.3. España Virtual

España Virtual es un proyecto CENIT, subvencionado por el CDTI dentro del programa Ingenio 2010, orientado a crear un puente entre el mundo geográfico y las tecnologías de Internet.

Su objetivo es la definición de la arquitectura, protocolos y estándares de la Internet Geográfica, con un foco especial en la visualización 3D, mundos virtuales e interacción entre usuarios. España Virtual incluye aspectos semánticos y tecnologías para el procesamiento masivo y almacenamiento de datos geográficos.

Con una duración de cuatro años y un presupuesto de 25 millones de euros, Elecnor Deimos ha liderado un consorcio de 20 miembros que ha tenido un importante impacto nacional e internacional. Además de Elecnor Deimos, el consorcio cuenta con la participación del Centro Nacional de Información Geográfica (IGN/CNIG), Indra Espacio, Androme Ibérica, GeoSpatiumLab, Desginit, Prodevelop, Telefónica I+D y más de una decena de prestigiosos centros de investigación y universidades nacionales, entre las que se encuentra el Instituto de Automática e Informática Industrial (ai2) de la Universidad Politécnica de Valencia.

España Virtual se ha estructurado en nueve paquetes de trabajo que se agrupan en dos áreas: Datos e Infraestructura Geográfica y Arquitectura y Tecnologías de Internet 3D. La primera de las áreas trata todos los temas relativos a la obtención y el tratamiento de los datos geográficos, incluyendo investigaciones para mejorar el procesamiento y el almacenamiento de los mismos. La segunda área trata de hacer estos datos accesibles a los usuarios finales a través de productos de visualización y servicios en Internet.

Cada una de estas áreas se compone de tres paquetes de trabajo con diversas líneas de investigación tecnológica y activos experimentales parciales y un cuarto paquete en el que realizan activos experimentales integrados que combinan varias tecnologías y resultados de dicha área.

El noveno paquete de trabajo se ha encargado de la coordinación y supervisión técnica de las líneas de investigación entre el resto de paquetes y áreas, así como del replanteamiento al inicio de cada ciclo del proyecto. Además, en este ámbito se ha coordinado la participación y colaboración del proyecto con distintas instituciones de estandarización e investigación a nivel nacional e internacional.

La investigación en las distintas líneas y la experiencia derivada de los activos experimentales integrado permiten la participación activa de los miembros del consorcio en la actualización de normas y recomendaciones, así como en la creación de nuevas propuestas.

Los trabajos llevados a cabo durante los cuatro años que ha durado el proyecto han producido una cantidad elevada de resultados, plasmados tanto en informes públicos de avance en la investigación en las diferentes temáticas del proyecto, como en la realización de demostradores y activos experimentales que, en un futuro, tienen la capacidad de evolucionar a productos comerciales.

El proyecto ha finalizado en mayo de 2012.

3

Entorno para simbología tridimensional

Uno de los elementos que dotan de realismo un entorno virtual es la simbología tridimensional. Con los Sistemas de Información Geográfica en 3D, han aparecido nuevas formas de representar datos vectoriales mucho más intuitivas y realistas para los usuarios. Por tanto se pretende obtener un *framework* de representación de objetos 3D.

Para ello se ha trabajado sobre una librería multiplataforma y de código libre, que permite visualizar estos datos mediante símbolos tridimensionales como podrían ser árboles o postes eléctricos. Además, se quiere poder editar esos objetos tridimensionales, presentando un sistema de edición interactiva mediante una librería de manipuladores para la edición de datos vectoriales y símbolos que permite ser fácilmente integrada en aplicaciones SIG.

3.1. Simbología tridimensional

En una vista tridimensional no es necesario proyectar entidades del mundo real como edificios o árboles, sino que podemos representarlos directamente con modelos sintéticos sobre el terreno. De esta forma, no sólo tenemos una representación visual

de la latitud y longitud de una entidad, sino que además podemos conocer su altitud. Al mismo tiempo, se simplifica la lectura de los mapas, haciendo a veces innecesario el uso de complejas leyendas. Sin embargo, en la actualidad no existe ningún estándar para trabajar con volúmenes sobre datos vectoriales, sino que los objetos tridimensionales se representan mediante entidades 2D con un atributo de altura o en su defecto como conjuntos de polígonos. Algunos clientes SIG implementan su propia topología para objetos 3D, el problema es que no es compatible con otras aplicaciones.

La librería con la que se ha trabajado esta basada en el grafo de escena de OpenSceneGraph, incorpora un sistema de tratamiento de simbología que permite, no sólo modificar los atributos propios de las entidades como pueden ser el color o el grosor del contorno, sino que además permite sustituir estas características por modelos sintéticos en tiempo de visualización, consiguiendo representaciones más intuitivas y realistas que con los métodos tradicionales.

A continuación, se describirán las principales técnicas para renderizar datos vectoriales en el ámbito SIG, así como las diferencias que existen entre las entidades y el símbolo que se utiliza para representarlas. Se hablará sobre la importancia de los símbolos en clientes 3D y las posibilidades que ofrece el conjunto de herramientas desarrolladas. Posteriormente, se tratará sobre la edición interactiva, tanto de datos como de símbolos, mediante manipuladores desarrollados específicamente para este tipo de aplicación. Después, se explicará como se han integrado este conjunto de herramientas en un cliente SIG real y se comentarán los resultados obtenidos.

Existen dos métodos muy extendidos para renderizar datos vectoriales en aplicaciones SIG 3D. El primero, consiste en rasterizar previamente las entidades y trabajar con la imagen generada como si se tratará de datos raster, es decir, aplicándolos como una textura sobre el terreno. Este método tiene las mismas desventajas que los datos raster, pero tiene la ventaja de que los datos siempre quedan pegados sobre la superficie del terreno, evitando que queden por encima o por debajo del terreno. Algunas implementaciones permiten rasterizar los datos bajo demanda, creando modelos simples de multiresolución como en [1]. Otra implementación utilizando *shaders* está descrita en [2].

El segundo método consiste en renderizar directamente las entidades como primitivas geométricas. Esta estrategia tiene la ventaja de facilitar los métodos de edición interactiva, así como el uso de algunas herramientas de análisis. Sin embargo, aplicar un modelo de multirresolución a estas geometrías o lograr que se sitúen sobre la superficie de un modelo de terreno con distintos niveles de detalle, son tareas mucho más complicadas.

El tratamiento de la simbología es un elemento común entre los clientes SIG 2D. Normalmente se representan los datos vectoriales por diferentes símbolos que ayudan al usuario a categorizar estas características vectoriales. Además se permite elaborar leyendas donde el color o el patrón de la geometría puede indicar el valor de alguno de los atributos asociados. Muchos desarrolladores se han limitado a exportar esta funcionalidad a sus aplicaciones 3D, sin explotar las posibilidades que ofrece este nuevo espacio de trabajo, aportando como mucho algún mecanismo de extrusión de geometría.

Por ejemplo, supongamos que se dispone de una serie de datos sobre los bosques de una región. Se pueden representar los polígonos que se definen en la información vectorial, como conjuntos de modelos sintéticos de árboles, variando su densidad en función de un atributo que indique la densidad de árboles real y su color en función de otro atributo que indica el peligro de incendio. En un espacio bidimensional no sería posible albergar tantos datos, es por ello que surge la necesidad de desarrollar una librería que permita el uso de simbología tridimensional para clientes SIG, de forma que los mapas creados tengan mayor realismo y sean capaces de proporcionar al usuario muchos más datos mediante nuevas herramientas de análisis.

La mayoría de los clientes SIG 3D proveen de los mecanismos necesarios para modificar los símbolos durante la visualización como en *ArcGIS 3D Analyst* [3] , pero no aportan ningún mecanismo para editar las entidades de forma interactiva desde la propia vista 3D. Para modificar estos datos, es necesario modificar directamente la base de datos, abrir una vista 2D o utilizar una herramienta CAD externa con la incomodidad que esto representa para el usuario final. Por tanto, es necesario aportar manipuladores de geometría intuitivos que permitan modificar los datos y ver los resultados sin necesidad de tener que cambiar de espacio de trabajo.

Dada la enorme cantidad de polígonos presentes en una vista 3D de una aplicación SIG, se ha optado por aprovechar los beneficios que aportan los grafos de escena, como *OpenSceneGraph* [1] (OSG) [4] que incorpora mecanismos para facilitar el uso de técnicas de aceleración gráfica tales como: LOD, view-frustrum culling, back-face culling o small feature culling, etc.

[1] http://www.openscenegraph.org

3.2. Desarrollo del entorno

En las siguientes secciones, nos centraremos en el componentes osgVP-Symbology que es el encargado de la visualización datos vectoriales.

El objetivo principal del componente osgVP-Symbology es ofrecer una API a los desarrolladores que de soporte para la visualización de datos vectoriales con garantía de interactividad. Una buena estrategia a la hora de definir la simbología es un factor clave para aprovechar toda la potencia del grafo de escena utilizado en la visualización.

Tras el estudio de los requerimientos de los sistemas SIG actuales, se ha decidido definir los siguientes símbolos: puntos, polilíneas, polígonos, texto, símbolos compuestos y símbolos extruídos. Cada una de estas entidades básicas se especializa en otras específicas para poder abarcar cualquier tipo de geometría deseada para un símbolo. De este modo tendremos, por ejemplo, puntos 2D y puntos 3D, que a su vez se especializan en modelos geométricos más concretos.

En nuestro modelo, un símbolo puede contener muchas entidades, que serán representadas del mismo modo. Cada símbolo requiere de un tratamiento distinto para mejorar la velocidad de visualización, por eso en el proceso de construcción del grafo se ha utilizado el patrón de diseño visitor [5]. Esencialmente, cuando el símbolo acepta al visitor, éste recorre todas las geometrías contenidas en el símbolo, creando un grafo de escena eficiente y listo para ser visualizado. Esta mejora de eficiencia se logra mediante el uso de vertex arrays, index arrays y primitive sets reduciendo las llamadas a funciones y eliminando la redundancia de vértices compartidos.

El funcionamiento es el siguiente: un *vertex array* almacena toda la geometría de un símbolo, compactando así todas las entidades que se expresen mediante ese símbolo en un único vector. Más tarde, se declaran los *primitive sets* estableciendo una relación entre conjuntos de vértices y primitivas OpenGL [2]. El grafo de escena es el encargado de gestionar estos *vertex arrays* por medio de *display lists*, almacenando en memoria de tarjeta gráfica la información geométrica.

Para obtener el máximo partido de este método se ha decidido utilizar color arrays para definir el color de cada vértice. De este modo, el color de cada entidad no influye en el número de símbolos declarados, ya que en el mismo símbolo se pueden almacenar entidades de diferentes colores. Factores que obligan a la diferenciación entre símbolos son el tamaño, el patrón de relleno, la anchura de línea, etc.

[2] http://www.opengl.org

A continuación se detallan las decisiones referentes a la arquitectura e implementación adoptadas para tratar los problemas asociados a cada tipo de símbolo.

3.2.1. Puntos

Un requerimiento corriente en los SIG es poder expresar puntos geográficos por medio de símbolos que sean fácilmente reconocibles por el usuario final (como gasolineras, hospitales, hoteles, etc.). Para lograr este objetivo las clases básicas han derivado en otras específicas siguiendo el siguiente esquema:

Point2DSymbol: pueden ser representados por medio de círculos, cuadrados, triángulos y en general, cualquier tipo de polígono. A estos elementos se les puede aplicar una textura, ya que en la construcción del grafo se detallan las coordenadas de textura de cada geometría.

Point3DSymbol: se da soporte para el dibujado de figuras geométricas tridimensionales básicas como cajas, esferas, conos y cilindros, además de la posibilidad de cargar cualquier modelo geométrico complejo desde archivo y utilizarlo para expresar dichos puntos geográficos.

Otro requerimiento típico es poder representar puntos en diferentes unidades de medida: píxeles y metros. En el caso de tener que representar los puntos en píxeles se añaden nodos AutoTransfrom al grafo de escena para mantener el tamaño del símbolo en pantalla aunque cambie el punto de vista.

3.2.2. Texto

Un factor decisivo en la usabilidad de un SIG es poder representar texto masivamente sin perder la capacidad de interacción. En el modelo que se plantea en osgVPSymbology se proporcionan tres maneras diferentes de representar texto en un mapa tridimensional:

Text2D: esta entidad es útil en el caso de renderizar textos que no se solapen entre ellos, o en el caso en el que la velocidad de render sea un factor clave en la visualización de determinada zona geográfica. A estos objetos se les puede aplicar billboarding, con lo que se consigue que el texto siempre esté encarado hacia el observador y que, por tanto, sea legible independientemente del punto de vista del usuario.

Text3D: cuando un usuario desea visualizar texto con volumen y que no deba estar encarado necesariamente al observador se debe utilizar la estructura de *Text3D*. La

visualización de este tipo de objetos tiene un coste computacional alto, por lo que deberían ser utilizados en el caso de que los factores estéticos primen sobre la velocidad de visualización.

FadeText: esta estructura ha sido diseñada para los casos en los que unos textos están ocluidos por otros, dificultando la interpretación por parte del usuario de los datos representados. Cuando la cantidad de textos que deben ser visualizados es relativamente grande, o se encuentran en una zona geográfica pequeña la escena puede resultar ilegible. Por eso *FadeText* hace desaparecer automáticamente y por medio de una transición suave los textos que quedan ocultos por otros.

La API de osgVP-Symbology permite asignar parámetros tales como el tamaño, el color, la alineación o el tipo de fuente a utilizar en la representación de cualquier tipo de texto.

3.2.3. Polilíneas y polígonos

La visualización de polilíneas y de polígonos es una actividad que los usuarios de SIG realizan con frecuencia. La siguiente figura muestra la estructura interna de un símbolo de tipo polilínea, en el que varias entidades se compactan en el mismo *vertex array*.

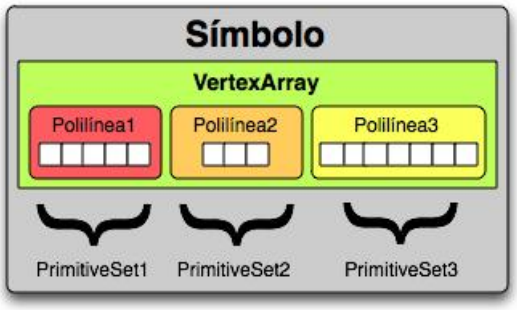

Figura 3.1: Estructura interna de un símbolo de tipo polilínea

Los usuarios podrán modificar el aspecto de una polilínea cambiando su grosor o patrón, y en caso de polígonos añadiendo textura.

3.2.4. Símbolos compuestos (*Composite Symbols*)

La combinación de símbolos es útil cuando se pretenda dibujar símbolos complejos o cuando se desee tratar varios símbolos como si fuesen uno sólo. En el modelo presentado, estas estructuras se han implementado siguiendo el patrón composite. Un ejemplo típico de la necesidad de este tipo de símbolos es la visualización de polígonos con borde con la finalidad de visualizar líneas fronterizas.

3.2.5. Geometrías definidas por extrusión

El componente osgVP-Symbology ofrece herramientas para visualizar entidades que tienen dimensión vertical en la realidad, como vallas, edificios, etc. Se ofrece un conjunto de herramientas de extrusión basado en un sistema de pilas de matrices que acumulan las transformaciones realizadas sobre la geometría original. Se dispone de extrusores especialistas en cada tipo de geometría que se desee extruir: puntos, polilíneas y polígonos.

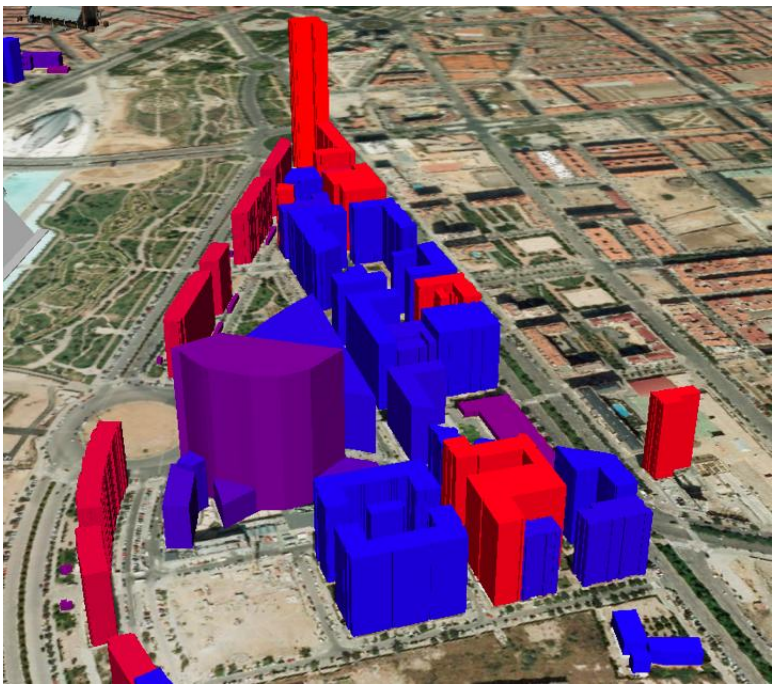

Figura 3.2: Zona urbana de Valencia representada mediante símbolos de extrusión.

- **PointExtruder**: permite al usuario convertir puntos en líneas. Estas estructuras se evidencian de verdadera utilidad cuando se usan combinadas con otros símbolos, por ejemplo en el caso de querer visualizar postes de electricidad o farolas. La dirección y el color de la extrusión puede ser definida por el usuario.

- **PolylineExtruder**: esta entidad tiene la capacidad de transformar polilíneas en múltiples polígonos. Puede utilizarse masivamente en la representación de carreteras, vallas, etc. A estas estructuras se les puede aplicar textura, logrando una imagen más realista y permitiendo de este modo la fácil interpretación de los datos por parte del usuario.

- **PolygonExtruder**: la geometría resultante de la extrusión de un polígono puede utilizarse para representar edificios, estadísticas, símbolos volumétricos, etc. Este efecto se consigue mediante el uso de cintas de triángulos y asignación de coordenadas de textura.

3.3. Herramientas de edición

Una vez establecida la arquitectura necesaria para la visualización de la simbología 3D, se ha desarrollado un sistema que permite la edición de las entidades representadas por estos símbolos. El sistema es capaz de editar los puntos, líneas, polígonos y texto que forman las entidades 3D. Esta característica permite transformar de forma sencilla e interactiva las entidades presentes en los mapas. Esto será de utilidad, por ejemplo, si se quiere modificar la posición de una ciudad en un mapa (punto), cambiar el curso de un río (polilínea), o modificar los límites de una población (polígono). La realización de todas estas tareas es más fácil e intuitiva si se puede realizar visualmente en lugar de tener que editar los archivos que contienen esas entidades.

La implementación de los símbolos realizada por osgVPSymbology hace que estos sean añadidos como nodos del grafo de escena. Por ello, se ha utilizado el componente osgVPManipulator para completar algunas de estas tareas de edición. Para llevar a cabo todas las transformaciones requeridas ha sido necesario realizar modificaciones en el componente, puesto que en un principio éste no permitía la edición individual de vértices. Actualmente, osgVPManipulator ofrece dos opciones para manipular un nodo.

En primer lugar, se puede aplicar una transformación a toda la geometría incluida en el nodo. Por otro lado, se puede modificar sólo un conjunto de vértices de la misma.

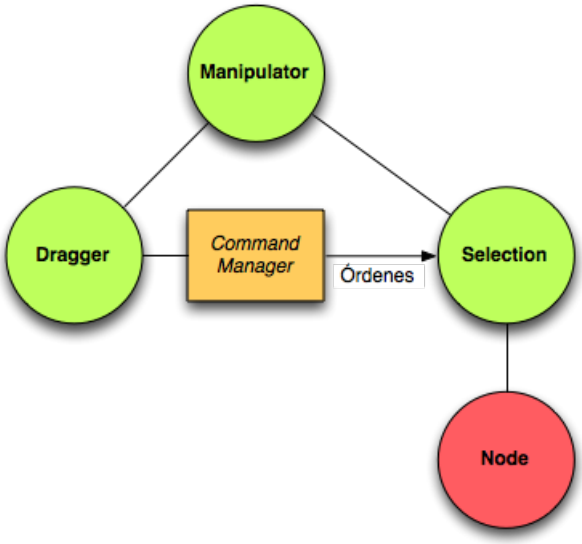

Figura 3.3: Arquitectura del nodo Manipulator. En él podemos observar los elementos que lo forman: Dragger, Selection y CommandManager.

A continuación, se explica con más detalle las posibilidades que nos ofrece el componente osgVP-Manipulator como editor de grafos de escena y los cambios realizados para soportar la edición de conjuntos de vértices de las geometrías. Más tarde, se mostrará cómo adaptar la librería para que sea capaz de editar la simbología proporcionada por osgVP-Symbology.

3.3.1. Edición con Manipulator

El primer objetivo es que el sistema sea capaz de editar los nodos como una sola entidad, transformando las geometrías que se encuentran dentro de ellos. Esto es, todos los vértices de la geometría serán transformados conjuntamente. Para conseguir esto, se ha hecho uso del nodo *Manipulator* desarrollado en osgVP-Manipulator. Este nodo es especialmente útil si se quiere transformar toda la geometría al mismo tiempo. En la figura 3 se puede observar la arquitectura del subgrafo generado por el nodo *Manipulator*, y en la figura 5 se muestran distintos tipos de dragger de los que disponemos [6], aplicados sobre un nodo.

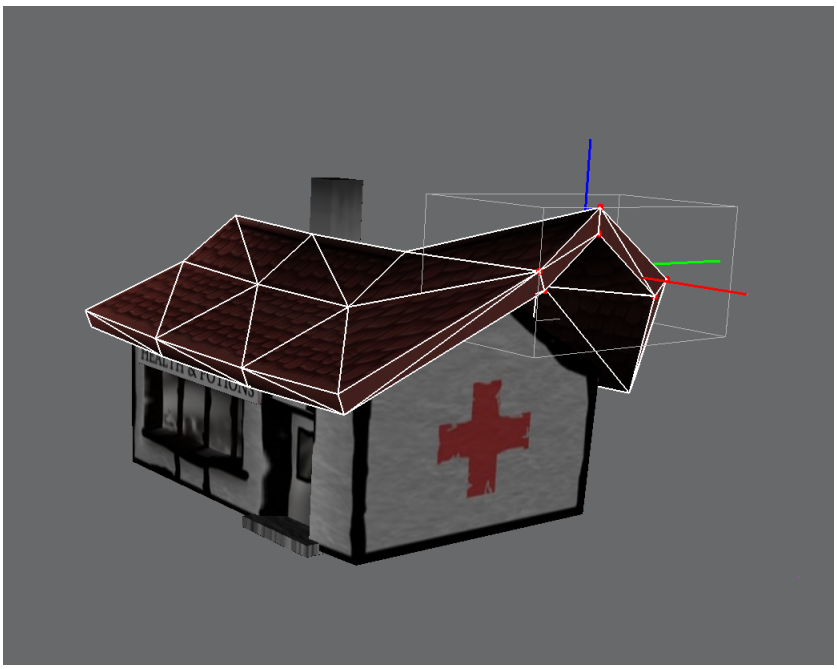

Figura 3.4: Nodo *GeometryManipulator*. Podemos observar los 3 dragger de traslación 1D representando los ejes de coordenadas, y los 6 planos de traslación 2D que forman el *MultiVertexDragger*, representando la caja de inclusión que contiene los vértices a editar.

3.3.2. Edición con GeometryManipulator

Para editar un subconjunto de vértices de un nodo, el sistema tiene que permitir la selección individual de los mismos. Para conseguir esto se ha desarrollado un nuevo nodo llamado *GeometryManipulator*. Este nodo toma como base la implementación de los manipuladores de osgVP-Manipulator, modificando su funcionamiento para adaptarse a estos nuevos requerimientos. Entre las modificaciones realizadas cabe señalar la creación de un nuevo tipo de dragger llamado *MultiVertexDragger*, el cual es una combinación de otros draggers. En la figura 6 podemos ver una captura de este tipo de estructura. También se ha creado un nuevo tipo de selección que se adapta a las nuevas órdenes generadas por el *MultiVertexDragger*, llamada *GeometrySelection*.

Los cambios realizados en la nueva selección residen en cómo ésta procesa las órdenes que recibe. En este caso, las órdenes no modifican una matriz de transformación, como en el caso del nodo Manipulator, sino que directamente modifican el valor de los vértices de la geometría. Una diferencia importante es que esta selección no actúa sobre cualquier tipo de nodo, sólo puede actuar sobre nodos de tipo *Geometry*, que son los que almacenan la información sobre la posición de los vértices.

3.3.3. Edición de las entidades

Anteriormente, se ha explicado cómo funciona la arquitectura del sistema de edición del grafo de escena y las diferentes posibilidades que nos ofrece. A continuación procederemos a ver más específicamente cómo editar las entidades representadas por los símbolos de osgVP-Symbology.

En el modo de visualización, cada entidad puede representarse por un conjunto de diferentes símbolos, pero cuando una entidad pasa a estar en modo de edición, se le asocia una geometría básica (punto, polilínea o polígono) a la que se le pueden aplicar transformaciones afines mediante los manipuladores que aporta osgVP-Manipulator.

El paso de modo visualización a mode de edición se realiza mediante una selección interactiva de los símbolos dibujados en pantalla por medio del usuario. Estas transformaciones se hacen con el objetivo de modificar la geometría de la entidad, manteniendo la consistencia topológica con los datos vectoriales asociados.

Para hacer esta tarea más sencilla, cuando el modo de edición esté activo, se muestran puntos en el lugar correspondiente a cada una de las coordenadas pertenecientes a la geometría de las entidades y todas las transformaciones se realizarán sobre dichos puntos. Los cambios realizados en las entidades se verán reflejados en los símbolos automáticamente, en cuanto el usuario vuelva al modo de visualización. La edición de las entidades puede llevarse a cabo de modos diferentes según el tipo de manipulador escogido. El nodo *GeometryManipulator* permite realizar las operaciones de edición sobre los vértices en cualquier tipo de símbolo, al actuar directamente sobre las geometría propia de cada entidad. En el caso del nodo Manipulator, se transformarán al mismo tiempo todas las entidades que se encuentren representadas por el mismo símbolo. Esto es debido a las optimizaciones que realiza osgVP-Symbology en la generación del grafo de escena, agrupando todas las entidades en la misma geometría.

Por otro lado, la apariencia de los símbolos que representa a las entidades puede ser modificada por el interfaz gráfico de usuario que provea el sistema SIG en el que se implante la librería.

4

Nuevo soporte para capas vectoriales

En el modelo anterior, las capas vectoriales cargaban todas las geometrías del archivo y las añaden directamente al grafo de escena del globo 3D, sin pasar por el sistema de gestión de capas que siguen las capas ráster. A la hora de añadir un sistema de filtros, es interesante que todas las capas pasen por el sistema de gestión de capas, al que denominamos **LayerManager**, para poder aplicar filtros de forma homogeneizada.

Para ello, se ha introducido un nuevo concepto dentro de la arquitectura del globo 3D al que denominaremos **VectorialTile**. Un **VectorialTile** es una entidad abstracta que abarca toda la extensión de una capa vectorial y que utilizará para gestionar los datos de dicha capa. Este *tile* se añade a la raíz del grafo de escena tal y como muestra la figura 4.1.

Si la capa no es multirresolución, la gestión de los datos y la visualización de los mismos se realizará dentro de dicho *tile*, cargando todos los datos de la capa y mostrándolos en la escena. Sin embargo cuando las capas vectoriales sean multiresolución, el **VectorialTile** deberá generar una estructura de tipo *quadtree* u *octree* con nodos paginados, muy similar a la que se generá para las capas ráster. Los **VectorialTile** implementaran el mismo sistema de petición de datos que los *tiles* empleados

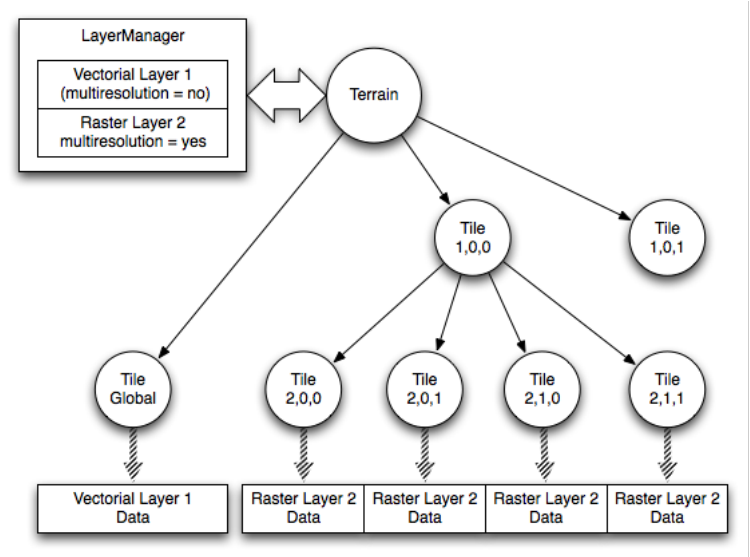

Figura 4.1: Diagrama del grafo de escena generado para un globo 3D con una capa ráster multirresolución y una capa vectorial sin multirresolución.

para ráster, permitiendo emplear los mecanismos que ofrece el **LayerManager** para la conexión con los distintos drivers o proveedores de datos.

4.1. *Drivers* de datos

Dado que el formato de los datos de una capa puede ser de diversa índole (servicios remotos o archivos locales: png, jpg, tiff, etc) se necesitan distintos proveedores de datos. La arquitectura del globo 3D debe permitir conectarse a estos proveedores, ya sea solicitando los datos directamente a una aplicación como puede ser gvSIG o una librería como podría ser *GDAL*. Para gestionar la demanda y recepción de estos datos, se debe implementar una clase abstracta llamada **DataDriver**.

Un **DataDriver** consta una cola de eventos de entrada y una cola de eventos de salida, tal y como se muestra en la figura 4.2. Cuando un *tile* necesita datos, lo que hace es añadir un evento de petición de datos a la cola de entrada del **DataDriver** asignado a la capa. Dentro de este evento se incluyen todos los datos referentes a la capa y al *tile* que ha realizado la petición, para poder solicitar al *driver* o a la aplicación las imágenes o geometrías correspondientes a la región del *tile*.

Toda capa debe tener asignado un **DataDriver** o de lo contrario los eventos de petición de datos se descartarán. Sin embargo, un mismo **DataDriver** puede gestionar los

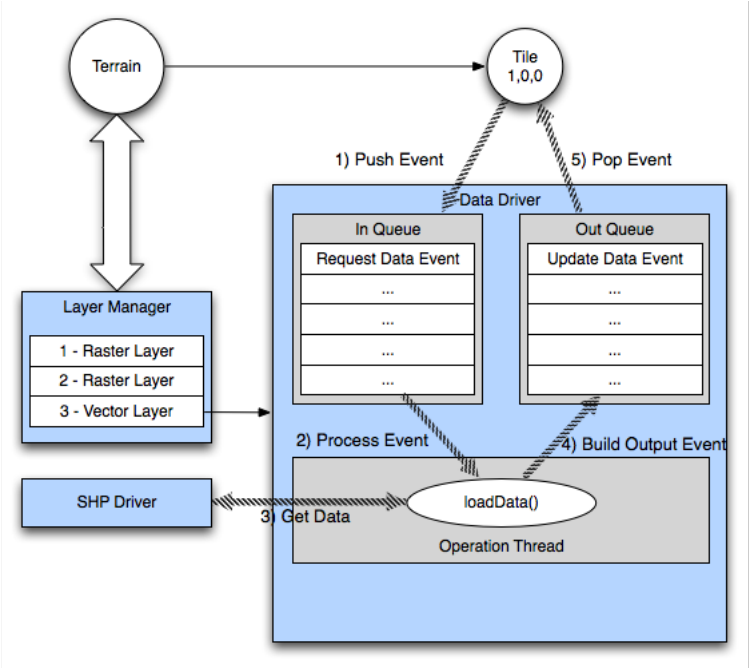

Figura 4.2: Diagrama de funcionamiento de un **DataDriver**.

eventos de varias capas, aunque lo lógico sería asignar uno por cada fuente de datos. De esta forma la implementación se simplifica, pues de cara al globo 3D todas las peticiones se gestionan de la misma forma, sea cual sea la fuente de datos, mientras que el **DataDriver** realiza la función de traductor, convirtiendo los eventos que le llegan desde el globo en una serie de llamadas a una librería concreta o un módulo de una aplicación.

Los eventos encolados en el **DataDriver** se procesan en un *thread* independiente, pues es un proceso costoso, sobre todo cuando hay que descargar los datos desde un servicio remoto. Si se realiza este proceso dentro del mismo *thread* que el renderizado, la visualización quedaría bloqueada durante la obtención de datos y la navegación dejaría de ser fluida.

Todo **DataDriver** deberá implementar el método **loadData** donde, dado un evento de petición de datos, se realizan los pasos necesarios para conectarse al *driver* o módulo fuente, cargar los datos y empaquetarlos dentro de un nuevo evento que será añadido a la cola de eventos de salida. De esta forma, cuando el globo entre en su fase de actualización, podrá recoger estos datos y aplicarlos correctamente sin que la navegación se vea afectada.

Finalmente, se quiere ofrecer la posibilidad de aplicar una serie de filtros sobre

estos datos, de los que se hablará más en profundidad a continuación. Sin embargo, se debe tener en cuenta que muchas veces el coste de aplicar estos filtros puede suponer una reducción del rendimiento durante la visualización, así que es aconsejable añadir un sistema dentro de los **DataDriver** que permitan la inclusión y procesado de estos filtros durante la fase de carga de datos.

4.2. Filtros y operadores

En lo sucesivo, al mencionar filtros u operadores, se estará haciendo referencia a una caja negra donde a partir de una serie de datos de entrada, se aplicarán ciertas operaciones y obtendremos un conjunto de datos de salida. Estos datos de salida se podrán utilizar directamente en el grafo de escena o bien se podrán usar como entrada para otros filtros u operadores.

Al conjunto de uno o varios filtros conectados entre si, le llamaremos *pipeline* o tubería. El patrón de diseño *Pipes and Filters* es ampliamente utilizado en el diseño de arquitecturas software escalables y se describe con más detalle en el libro de Frank Buschmann [7].

Como se ha comentado, algunos filtros necesitarán un tiempo de computo considerable mientras que otros serán operaciones sencillas que podemos aplicar durante la fase de renderizado. Por tanto, vamos a distinguir entre 2 tipos de filtros que aplicaremos en fases distintas utilizando diferentes *pipelines*:

- Filtros, que por su coste, se aplican directamente sobre los datos en el **DataDriver**, es decir, se preprocesan. Por ejemplo filtros de rasterización, simbología, rampas de color, etc. De este tipo de filtros se hablará a lo largo de esta sección.
- Filtros, que se aplican directamente sobre el *thread* de renderizado. Serán básicamente filtros para mejorar la visualización y producir algunos efectos, como por ejemplo la generación de *shaders*. El uso y aplicación de estos filtros se explicará con más detalle en otra sección del presente documento[8].

Para simplificar el uso de filtros, toda capa añadida al globo 3D tiene asignada una *pipeline* entre sus atributos, que al principio esta vacía. Cuando un usuario añada un filtro a una capa dentro de la aplicación, lo que se hará será buscar el filtro dentro de una factoría y añadirlo a esa *pipeline*, conectando correctamente las entradas y las salidas. De esta forma cuando un **DataDriver** solicite datos a un *driver* o aplicación

externa, se utilizarán estos datos como parámetro de entrada de la *pipeline* asignada a la capa y procesarla. Los datos resultantes al final del *pipeline* tendrán aplicados los filtros añadidos por el usuario, y estarán listos para ser encapsulados y enviados al globo 3D como eventos.

El uso de una factoría de filtros permitirá también implementar una caché de filtros, pues estos se pueden reutilizar tantas veces como se quiera cambiando únicamente las entradas. De esta forma, se evita la creación de nuevos objetos en memoria, reduciendo su consumo. Sin embargo, hay que tener en cuenta que si un mismo filtro se esta utilizando a la vez en dos *pipelines* distintas, es posible que una de ellas este cambiando las entradas del filtro antes de que la segunda haya terminado de procesarlos, produciendo resultados erróneos que se deberían evitar con el uso de *mutex* o mecanismos similares.

4.3. Uso de filtros para la generación de simbología

En el caso de las capas vectoriales, ya sean con multirresolución o sin ella, los datos que se obtienen del *driver* son geometrías simples (básicamente puntos, líneas y polígonos) a los que posteriormente se desea asociar una serie de símbolos. El uso de filtros para establecer esta simbología, simplificará su implementación y su uso.

Cada capa vectorial tiene un símbolo asociado como parámetro de la capa. Existen una serie de símbolos básicos para representar cada tipo de geometría; **MarkerSymbol** para puntos, **LineSymbol** para líneas, **FillSymbol** para polígonos. A partir de estos símbolos básicos, el usuario puede asociar símbolos más complejos como objetos 3D, partículas o vídeo, utilizando una serie de *templates* o plantillas que se generan en la factoría del símbolos.

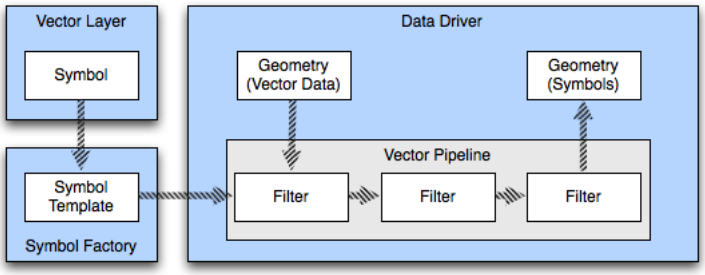

Figura 4.3: Diagrama de funcionamiento de un *pipeline* para aplicar símbolos a capas vectoriales.

Como puede observarse en la Figura 4.3, cuando un **DataDriver** obtiene los datos vectoriales para un *tile*, procede a consultar si la capa tiene algún símbolo asociado. Si tiene algún símbolo, solicita un *template* para ese símbolo a la factoría de símbolos, que se utiliza como parámetro de entrada del *pipeline* junto con las geometrías que hemos obtenido del driver. Dentro del *pipeline* se aplicarán los filtros necesarios para cambiar la simbología del la geometría, obteniendo como resultado un nuevo conjunto de geometrías listo para ser añadido a la escena.

4.4. Edición de capas vectoriales

El uso de filtros también nos permitirá simplificar la edición de capas vectoriales, en el caso de que se quisiera incorporar esta funcionalidad al globo 3D. Utilizando el sistema de filtros es posible crear una nueva *pipeline* especifica para el modo de edición que realice las siguientes tareas:

- Volver a solicitar al *driver* las geometrías originales para poder mantenerlas en memoria, editarlas y poder guardarlas posteriormente. [1]
- Visualizar símbolos sencillos, en lugar de los establecidos por el usuario, para hacer más intuitiva y sencilla la edición. [2]
- Añadir los manipuladores necesarios a la escena para permitir seleccionar y editar las geometrías.

La *pipeline* de edición que se muestra en la Figura 4.4, además de la geometría que se va a añadir al grafo de escena, devuelve también la geometría original (con las modificaciones realizadas por el usuario), de forma que cuando el usuario quiera guardar sus cambios solo se tendría que guardar dicha geometría en disco. Añadiendo un método **saveData** al **DataDriver** es posible realizar las llamadas necesarias para guardar las modificaciones dentro de un nuevo archivo o reemplazando el original.

Para establecer si una capa esta en modo edición y si debe utilizar la *pipeline* de edición, se debe añadir un *flag* a las capas vectoriales. También, necesitaremos implementar manipuladores específicos para selección y edición de geometrías, además de

[1] Hay que tener en cuenta que en visualización, la geometría que tenemos en memoria durante la visualización no tiene porque corresponder a la que se ha obtenido del *driver*, sino que es el resultado de aplicar una serie de símbolos y filtros.

[2] Los símbolos establecidos por el usuario pueden ser demasiado pequeños para poder seleccionarlos con facilidad o bien, demasiados complejos para saber si la geometría está situada en la posición o lugar deseado.

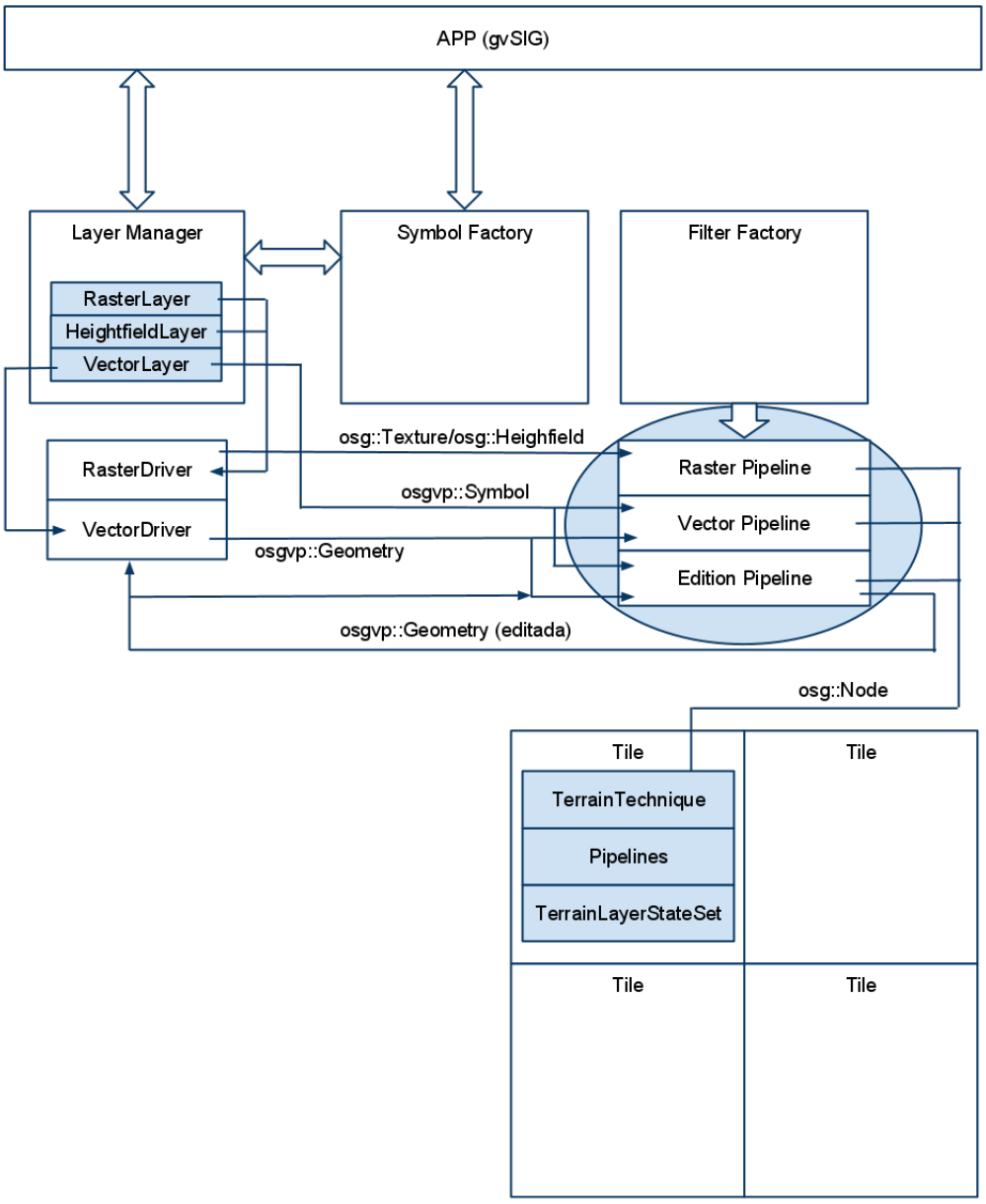

Figura 4.4: Diagrama de arquitectura del globo 3D donde se muestra el uso de diferentes *pipelines* según el origen de datos y de si estamos en modo de edición o no.

los manipuladores ya existentes para modificar la simbología o los objetos 3D añadidos al globo.

5

Simbología animada

En esta sección se explicarán los diferentes avances ques se han realizado en el campo de la simbología animada tridimensional. El objetivo es dotar de mayor realismo a la escena, así como conseguir símbolos que ayuden en mayor medida que un símbolo convencional a categorizar los datos por parte del usuario. También es importante que estos símbolos se representen preservando la interactividad usuario-escena y permitiendo una navegación fluida por la misma. Por ello, para la implementación de estos símbolos se han utilizado técnicas de representación que hacen uso extensivo de las capacidades de la tarjeta gráfica.

En lo referente a la implementación se ha creado una nueva librería dentro del motor de render de gvSIG 3D, osgVirtualPlanets. Esta librería denominada osgVPsymbology se ha desarrollado siguiendo el paradigma *pipes and filters* (filtros y tuberías), explicado en [9].

De este modo se consigue un sistema de construcción del grafo de escena modular en el que el cambio de uno de los módulos revierte en la construcción de un nuevo grafo. Este sistema también permite la creación de tuberías ya preparadas para obtener símbolos por defecto. También permite optimizar los datos para su posterior representación sin perjuicio para los datos originales, que siempre se conservan.

A continuación se detallan algunos de los símbolos implementados hasta el momento, explicando qué técnicas se han utilizado en cada uno de ellos.

5.1. Point Sprites

Muchas aplicaciones, incluyendo los sistemas de información geográfica, utilizan símbolos de imagen 2D o *sprites* para representar puntos. En un entorno 3D normalmente sólo es necesario tener las coordenadas del punto central de la imagen ya que si se almacena la geometría correspondiente a la imagen (un QUAD) y sus coordenadas de textura el gasto de memoria aumentaría significativamente -se cuadruplicaría- por punto añadido, además de tener que calcular las coodernadas de cada vértice en función del punto central.

Para evitar este gasto de recursos existe una extensión de OpenGL que calcula por nosotros la geometría y las coordenadas de textura del QUAD necesario para dibujar la imagen. Estos cálculos se hacen utilizando el procesador de la tarjeta gráfica, que está optimizado para efectuar este tipo de cálculos. Por ello no se consumen recursos del procesador principal del sistema.

Por otro lado se utiliza una técnica denominada *Billboarding* que es de especial interés para entornos SIG. En ella se utilizan los sprites de manera que la imagen siempre esté de cara al punto de vista de la cámara. Cada cambio en el punto de vista de la cámara implicará un cambio en la orientación del sprite.

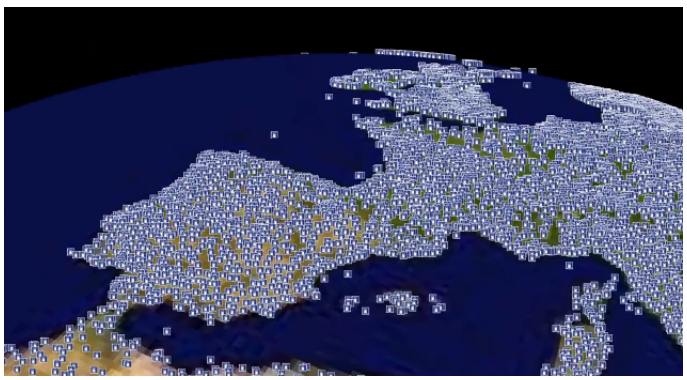

Figura 5.1: Capa de gasolineras utilizando point sprites como símbolo

5.2. Sistemas de partículas

Un sistema de partículas es un sistema que cumple determinadas leyes físicas qeu definen el comportamiento de dos o más puntos materiales. Gráficamente cada

partícula será renderizada en relación a diferentes parámetros que pueden ser internos(masa, velocidad, posición, etc.) o externos (gravedad, fuerzas exteriores, densidad del medio, etc.). También se deben tener en cuenta los parámetros que definen al emisor de estas partículas como los ángulos de lanzamiento o la velocidad y el número de partículas por segundo.

Para la integración con gvSIG 3D se han predefinido algunos sistemas de partículas con el objetivo de ocultar la complejidad del sistema al usuario final. En la figura 5.2 se puede observar un sistema que simboliza un fuego por cada punto de una capa de incendios.

Figura 5.2: Capa de incendios utilizando sistemas de partículas como símbolo

5.3. Instanciación de Geometría

La instanciación de geometría es una técnica que permite renderizar copias de la misma geometría en una escena de una vez. Esta técnica se utiliza normalmente para dibujar árboles, edificios o cualquier otra geometría que aparezca repetidamente en una escena. Aunque los vértices de la geometría sean los mismos, cada instancia puede dibujarse teniendo en cuenta otros parámetros que las diferencien de las demás, como mediante transformaciones afines o colores diferentes.

Estas cualidades hacen de la instanciación de geometría una técnica muy interesante en el uso de SIG 3D, ya que se pueden simbolizar puntos mediante geometrías complejas sin que ello suponga un cuello de botella para el sistema. Por ello el siguiente paso sería definir un catálogo de geometrías listas para usar en el SIG.

Para hacer uso de la instanciación de geometría se necesita una tarjeta gráfica y

drivers que soporte la extensión de OpenGL ”GL EXT draw instanced”. También es necesario el uso de shaders.

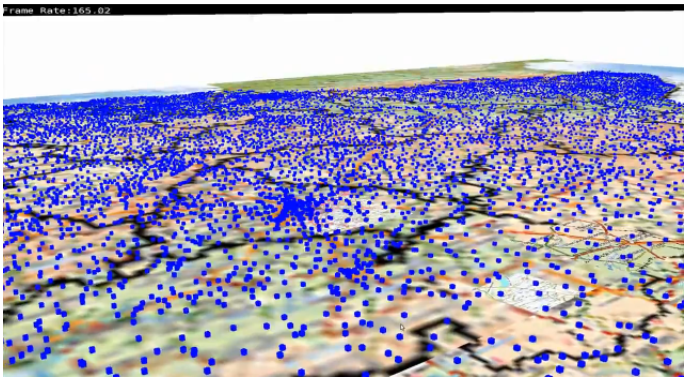

Figura 5.3: Capa de poblaciones de España, cada punto es representado por un cubo que gira en torno a su eje vertical con una tasa de 165 frames por segundo

6

Visualización estereoscópica en un globo virtual

La visualización estereoscópica de imágenes aéreas y de satélite ha sido una práctica común durante los últimos 30 años. Hasta ahora se han venido utilizando técnicas fotogramétricas para conseguir el efecto estereoscópico en sistemas de información geográfica 2D. Para ello se suelen usar cámaras fotogramétricas aéreas que van obteniendo fotografías de zonas previamente planificadas en un vuelo.

Si utilizamos esta técnica en un globo virtual, el principal inconveniente es que el efecto sólo es apreciable desde un punto de vista, desde aquel donde se tomó la fotografía, limitando la interacción y el propio efecto estereoscópico. Además de que sólo podremos conseguir estereoscopía en aquellas zonas en las que se haya llevado a cabo un vuelo fotogramétrico, con el coste que ello acarrea.

Con la llegada de los Sistemas de Información Geográfica (SIG) que incluyen visualización 3D interactiva, se pueden probar nuevas técnicas, mucho más económicas, en las que se genera el efecto estéreo a partir de síntesis de imagen digital. En lugar de utilizar fotogrametría, se utiliza la información geométrica de la que se dispone para conseguir el efecto 3D.

Esto sirve para potenciar la interactividad con el usuario, explorando puntos de vista imposibles de obtener mediante fotografía aérea. También nos permite añadir elementos que no aparecen en la ortofotografía, pudiendo así planificar proyectos de ingeniería o de arquitectura.

Algunas de estas técnicas han sido incluídas en gvSIG 3D [10], el globo virtual para gvSIG [11], y para validar la investigación se ha realizado un test a 51 participantes en un congreso SIG con tres dispositivos diferentes. A partir de los datos recogidos en un cuestionario posterior a la prueba, se presenta un análisis estadístico, con el objetivo de valorar la experiencia vivida por los participantes.

También se han realizado pruebas de funcionamiento en dispositivos que no se podían trasladar y montar en el recinto del congreso. La muestra de usuarios utilizada en estas pruebas no es significativa, y por eso no se han incluído en el estudio estadístico.

A continuación se presenta un resumen de las técnicas estereoscópicas más comunes en el marco de las tecnologías SIG. Después se describe la inclusión de estos procedimientos en gvSIG 3D, para luego comparar los globos virtuales con capacidad estéreo que se encuentran en el mercado actualmente. Para finalizar se detallan los resultados de la encuesta, haciendo un análisis estadístico de los resultados.

6.1. Técnicas estereoscópicas incluidas en gvSIG 3D

Todos los sistemas de estereoscopía se basan en un mismo principio: mostrar a cada ojo del observador una imagen diferente [12]. Si estas dos imágenes han sido tomadas o generadas de forma adecuada, nuestro cerebro las compara y percibe la escena en profundidad (3D). Este efecto no es perfecto, puesto que nuestro cerebro utiliza otros mecanismos para la percepción de la profundidad, como el enfoque realizado por el ojo, que no puede reproducirse hoy en día mediante los sistemas estereoscópicos.

En la mayoría de sistemas se utiliza un dispositivo sobre los ojos para que cada uno

de ellos perciba una imagen diferente. En esto se basan las diferentes variedades de gafas 3D, cascos de realidad virtual y sistemas de espejos o de filtros de polarización. Por otro lado la autoestereoscopía no requiere ningún dispositivo sobre los ojos, utiliza unas pantallas especiales para mandar las imágenes de cada ojo en diferentes ángulos, de manera que a cada ojo sólo llega una de las imágenes.

Las técnicas utilizadas en gvSIG 3D se basan en calcular el par estéreo a partir de geometría tridimensional, ya disponible. El procedimiento cambia según el tipo de dispositivo que se quiera utilizar, pero en cualquier caso sigue tres etapas claramente diferenciables: generación del número de vistas necesarias, renderizado de cada vista por separado y mezclado de vistas.

El proceso es el siguiente, en la fase de generación de vista se calculan las matrices de cada vista teniendo en cuenta parámetros como la distancia de fusión, distancia interocular y distancia del espectador a la pantalla. Estos parámetros son fácilmente configurables por medio de un diálogo en la interfaz gráfica de gvSIG 3D.

Los datos son enviados a la tarjeta gráfica que se encarga de renderizar cada vista en un *buffer*, para luego mezclar las vistas generadas según el dispositivo que se vaya a utilizar.

También se ha implementado un conjunto de herramientas para configurar el modo multimonitor y pantalla completa. Y a su vez poder sincronizar las vistas en el caso de estar ejecutando una animación definida por el usuario.

6.1.1. Anaglifo

La visión anaglifa es una de las técnicas más utilizadas para obtener sensación estéreo. Se puede utilizar con monitores convencionales junto con unas gafas especiales. Las gafas que se necesitan son muy baratas y cualquier persona puede disponer de ellas. Utilizan filtros de colores para separar las dos imágenes (Figura 6.1). Si vemos a través de un filtro rojo, los colores verde o azul se ven como negro. Si utilizamos un filtro verde, azul o cyan, el rojo parece negro.

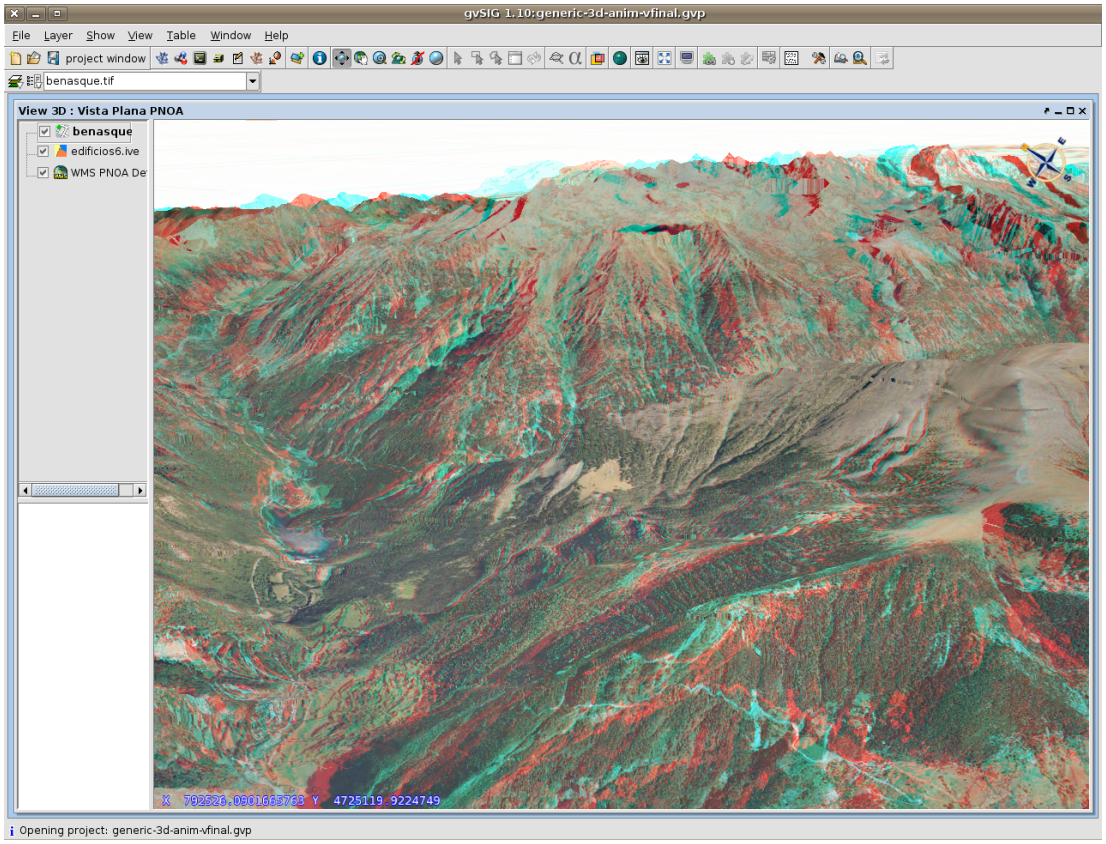

Figura 6.1: Imagen anaglifa del valle de Benasque , con gvSIG 3D utilizando como fuente de datos un servicio WMS y una capa de elevaciones.

En la fase de renderizado de las vistas se aplica una máscara de color a cada vista (rojo y verde). Una vez mezcladas las imágenes y en combinación con el funcionamiento de los filtros cada ojo verá la imagen que le corresponda del par estéreo. Esta técnica tiene el inconveniente de distorsionar los colores reales de los datos que se dispone, pudiendo afectar a la sensación de inmersión del espectador. Sin embargo el bajo coste del sistema y los resultados que se obtienen hacen de esta técnica una de las más utilizadas.

6.1.2. Sistema QuadBuffer

El sistema *Quadbuffer* es una de las opciones más caras de las cuatro presentadas en esta sección. Requiere la sincronización de las gafas con el dispositivo de visualización y por ello precisan de hardware adicional. No obstante, es la que produce a

Figura 6.2: Monitor Samsung de 120 Hz y gafas de conmutación NVidia 3D Vision utilizadas en la prueba, el objeto similar a una pirámide es el encargado de la sincronía entre las partes del sistema, lo hace en modo inalámbrico (imagen obtenida de http://3dvision-blog.com).

priori mejores resultados visuales.

El sistema consta de tres elementos básicos, mostrados en la Figura 6.2:

- Gafas de conmutación (*Shutterglasses*):las gafas de conmutación consisten en cristales de cristal líquido que son capaces de oscurecerse por completo y no dejar pasar la luz. Las gafas van alternando rápidamente la apertura y el cierre del LCD delante de cada ojo. Esto, junto con la proyección alternativa de las imágenes de cada ojo, permite que cada uno vea una imagen diferente, consiguiendo así la ilusión 3D.
- Monitor de 120 Hz: es necesario un monitor capaz de refrescar la pantalla al doble de la velocidad habitual(60-75 Hz) para poder hacer el intercambio de imágenes para cada ojo para poder crear el efecto estereoscópico. Además debe existir un dispositivo que sincronice la proyección de imágenes con las gafas de conmutación.

- Tarjeta gráfica *Quadbuffer*: para que no se resienta el rendimiento gráfico, debido a que se tienen que generar el doble de frames, se necesita una tarjeta profesional capaz de soportar esos niveles de carga. Estas mismas tarjetas se utilizan con cascos de realidad virtual.

6.1.3. Pantalla Autoestereoscópica

La clave de las pantallas autoestereoscópicas es conseguir que cada imagen (dos en el caso más sencillo, pero pueden ser hasta doce) se vea en un ángulo diferente, y suponiendo que cada ojo se encuentra en uno de esos ángulos, verá por tanto una imagen diferente. Existen dos formas de conseguir esto, partiendo de un monitor LCD o plasma normal, con el único requisito de que sus pixeles tengan forma rectangular:

- Barrera de paralaje: a unos milímetros de los pixels se coloca una rejilla que permite ver solamente algunos de ellos desde un ángulo determinado.
- Filtro lenticular: utiliza una lámina con pequeñas ondulaciones que actúan como lentes diminutas, enviando la luz de los pixeles en diferentes direcciones.

Originalmente, estas pantallas trabajaban solo con dos vistas, enviando la imagen del ojo izquierdo hacia la izquierda, y del ojo derecho hacia la derecha. Este sistema permitía percibir el volumen cuando cada ojo se situaba a un lado del centro de la pantalla, pero el efecto no podía percibirse desde otras posiciones.

Actualmente casi todos los sistemas autoestereoscópicos son multivista: trabajan con varias imágenes (de cinco a nueve es lo habitual) visibles en diferentes ángulos, y esto permite percibir el efecto desde diferentes posiciones. La Figura 6.3 muestra la generación de las 8 vistas necesarias para un monitor autoestereoscópico lenticular.

Para conseguir el efecto 3D, la imagen que se muestra en la pantalla debe seguir un entrelazado especial, dependiente de cada fabricante, que permita que el filtro lenticular o la barrera de paralaje envien cada una de las imágenes en una dirección diferente (ver Figura 6.4). La tecnología más extendida actualmente son los monitores lenticulares de 8 o 9 vistas, aunque conforme aumente la resolución de los monitores el número de vistas aumentará permitiendo mejorar la calidad del efecto 3D.

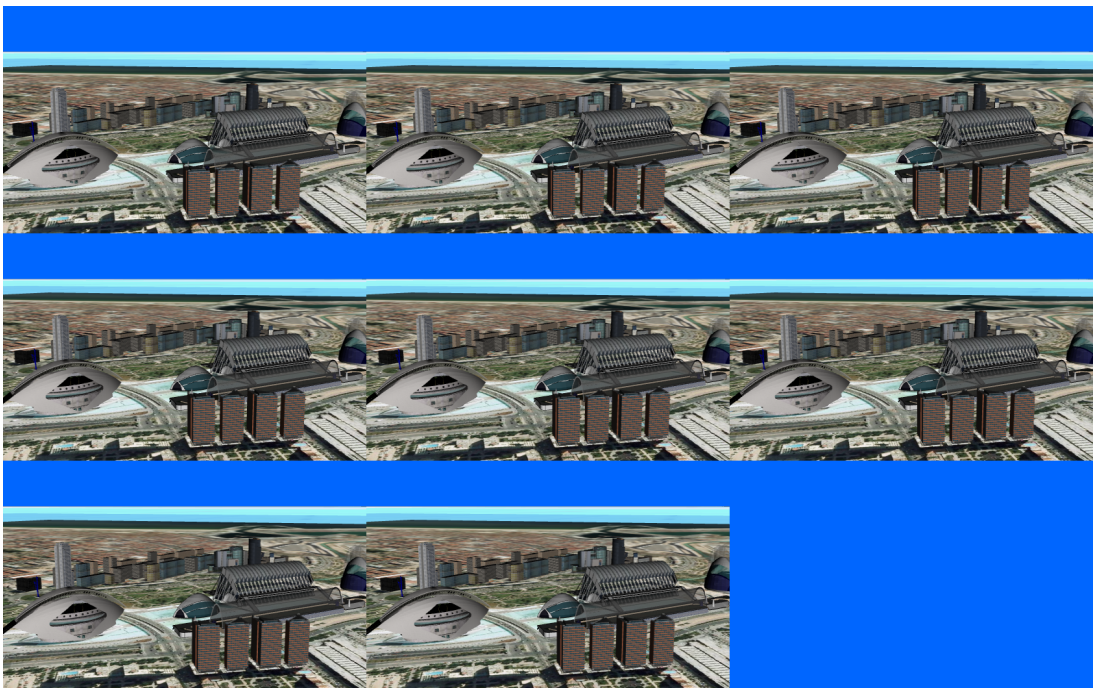

Figura 6.3: Mosaico con las 8 vistas generadas de una zona de la ciudad de Valencia. Cada una de ellas tiene una pequeña separación para conseguir el efecto 3D.

Finalmente la Figura 6.5 muestra una imagen que simula un efecto similar al que se siente al verlo en una pantalla autoestereoscópica.

6.1.4. Proyección dual con filtros de polarización

En la mayoría de los casos los anaglifos no permiten representar el color correctamente. Por ello se utilizan las gafas de lentes polarizas que requiren una mayor inversión por necesitar dispositivos de visualización especializados.

La técnica está basada en la polarización de la luz. Si se proyecta luz polarizada en una dirección y la vemos con un filtro polarizado, colocando el filtro a una inclinación de 90 grados respecto a la luz original, toda la luz será bloqueada. Por ello podemos proyectar dos imágenes, una polarizada en un sentido y la otra a 90 grados y utilizar dos filtros polarizados para que cada ojo vea una imagen distinta.

Los filtros son baratos, lo más habitual es que se utilicen con sistemas de proyec-

Figura 6.4: Imagen final de la zona de la ciudad de Valencia tras haber aplicado el entrelazado de las 8 Vistas.

ción, aunque también existen televisores que los incluyen. Además, requiere generalmente dos proyectores o un proyector especialmente modificado, además de una pantalla que no despolarice la luz. También hay que considerar que los filtros polarizados oscurecen la imagen y se necesitan proyectores muy luminosos.

Este tipo de estéreo se está empezando a comercializar en televisiones de hogar, permitiendo que más usuarios puedan ver el efecto 3D con un coste menor, ya que las gafas son muy económicas, a diferencia de las que usan un sistema de conmutación.

Para la integración de esta tecnología en gvSIG 3D se han utilizado dos opciones diferentes con idéntico resultado. La primera de ellas ha sido disponer de un menú que permitiera asignar las imágenes correspondientes a cada ojo a un proyector diferente, es decir se necesita disponer de una salida para las imágenes del ojo izquierdo que asignaremos a un proyector y otra para las de ojo derecho que asignaremos al otro proyector.

Por otro lado también existe la posibilidad de mandar una señal entrelazada uti-

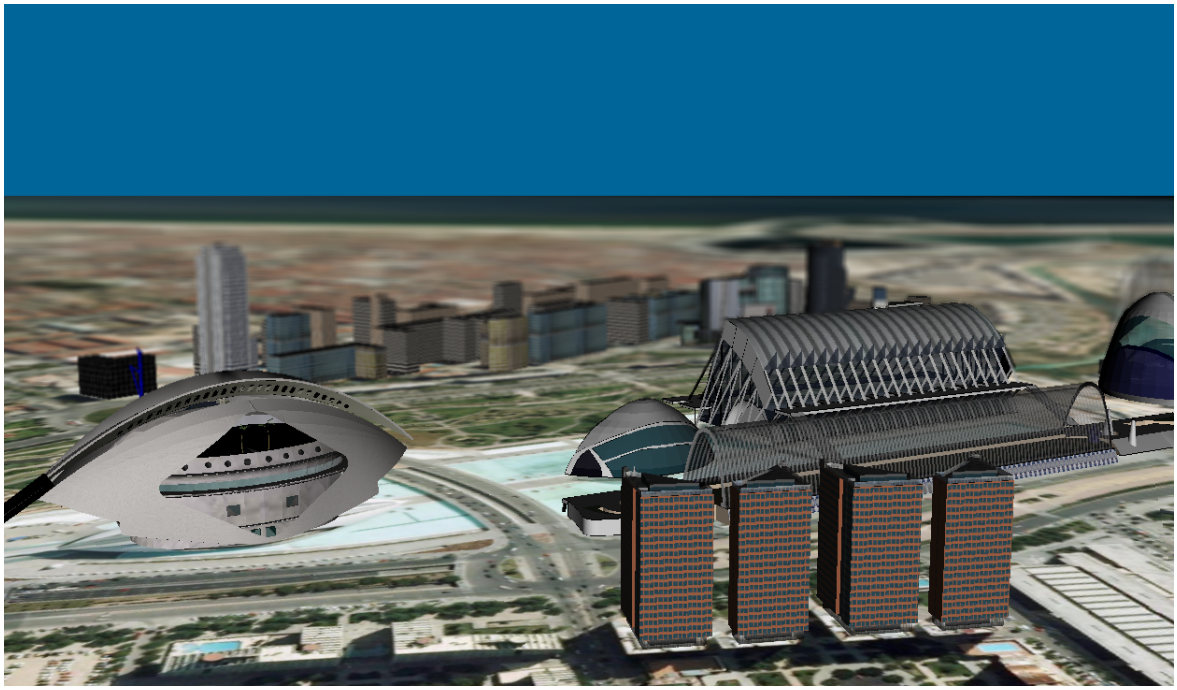

Figura 6.5: Imagen que emula el efecto autoestereoscópico de una zona de la ciudad de Valencia.

lizando una partición(*split*) horizontal o vertical de modo que el correcto envío de la señal a cada uno de los proyectores no depende de gvSIG 3D, sino del software que subyace. Esta técnica puede ser muy útil para los televisores que usan gafas pasivas de reciente comercialización. En estos televisores el formato de entrada que se suele utilizar es el de *split* horizontal.

6.2. Visualización de un globo virtual en un entorno CAVE

Un entorno CAVE (Cave Automatic Virtual Environment) es un entorno de realidad virtual inmersiva que consta de una sala en forma de cubo y una serie de cámaras que recogen la posición del usuario y retroproyectores de alta resolución que proyectan en las paredes y suelo de esta sala. El usuario entra en el cubo con unas gafas que le permiten, mediante visión estereoscópica, visualizar en tres dimensiones y con 6 grados de libertad. Cuando esto ocurre, el punto de vista del usuario se corrige mediante una adecuada proyección de la perspectiva, calculada en cada caso para la posición y orientación del usuario, que previamente ha sido detectada por las cámaras.

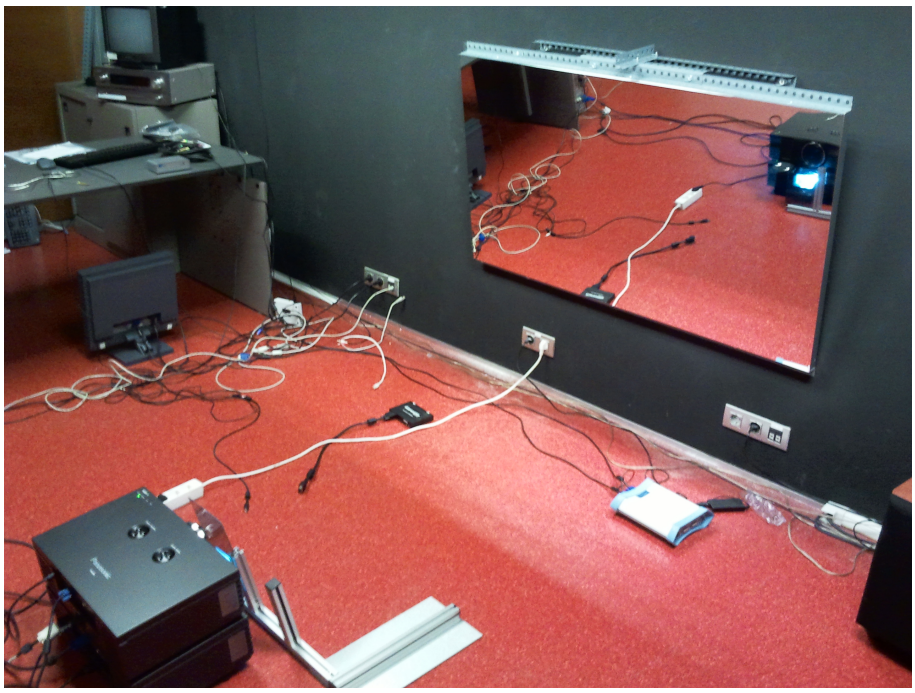

Figura 6.6: Sala de control del sistema de doble retroproyección del *visionarium* de la UPV. Un filtro se aplica a la salida del proyector, para después reflejarse en el espejo. Este espejo esta alineado con la pantalla que no despolariza la luz.

A la hora de visualizar escenarios tridimensionales en la CAVE se ha de tener en cuenta que el software disponible está preparado para visualizar imágenes estáticas, donde lo único que cambia es la posición del observador. Por eso se ha tenido que desarrollar una serie de controles para poder rotar y escalar los modelos, con el inconveniente de que la impletación de la herramienta zoom no es trivial y requeriría de un estudio más profundo. El esquema de funcionamiento planeado seria el que se muestra en la Figura 6.8.

Por eso para visualizar escenarios tridimensionales de terreno en la CAVE se han seguido principalmente dos estrategias:

- Visualización de terreno tridimensional en un plano: de este modo se pueden ver modelos de alta resolución y el usuario puede inspeccionar el terreno desde el nivel del suelo. Podemos ver un ejemplo de esta estrategia funcionando en la Figura 6.9.

Figura 6.7: gvSIG 3D con estéreo pasivo utilizando la sala de visualización *visionarium* de la UPV.

- Visualización del globo virtual en formato esférico: la idea en este caso el ofrecer al usuario el globo virtual en formato esférico con el inconveniente de que el usuario no es capaz de hacer zoom con las herramientas que se disponen hasta el momento. El usuario tiene la sensación de poder rodear el globo con sus brazos como se puede observar en la Figura 6.10

6.3. Comparativa de globos virtuales con soporte estereoscópico

Existen precedentes de integración de visión estéreo tanto en SIG 2D/3D, como en globos virtuales [13]. La mayoría de ellos operan en 2D y obteniendo imágenes anaglifas estáticas, pero los de especial interés para este trabajo son aquellos que son capaces de obtener estereoscopía al vuelo, es decir, dinámicamente y con capacidad interactiva.

La empresa ESRI con su extensión para ArcGis llamada Analyst3D [14] ha incluido recientemente soporte estéreo que incluye modos de visualización adecuados para

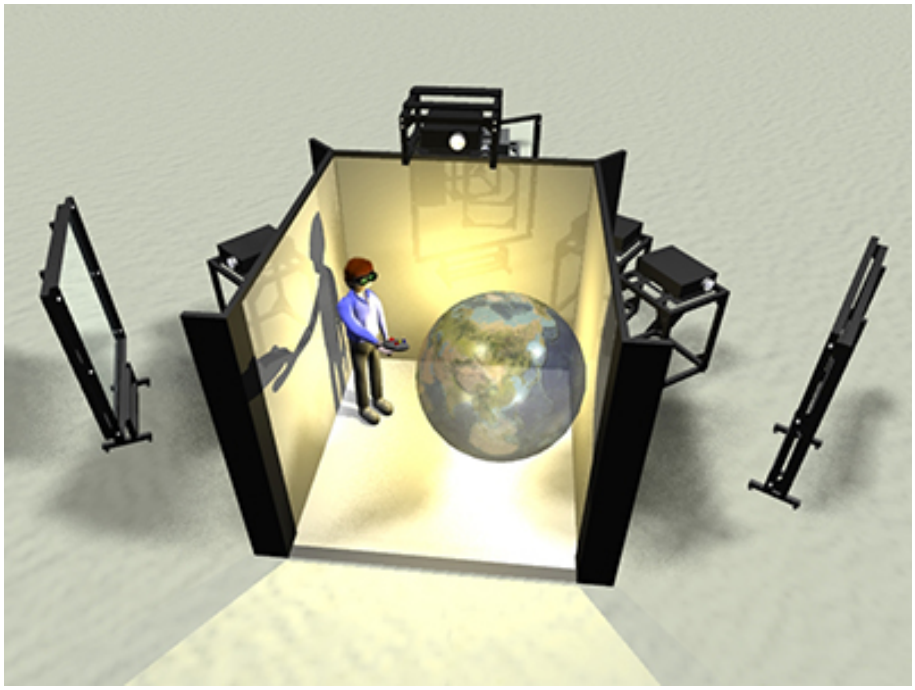

Figura 6.8: Esquema de un entorno *CAVE* para la visualización de un globo virtual.

FeaturesGIS 3D	ArcScene (ESRI)	Nasa World Wind	StereoGE Browser	TriDef GE	gvSIG 3D
License	privative	LGPL	freeware	privative	GNU/GPL
Platform	Windows	Windows, MacOSX, Linux	Windows	Windows	Windows, MacOSX, Linux
Anaglyphic stereo	Yes	Yes	Yes	Yes	Yes
QuadBuffer stereo	Yes	No	No	Yes	Yes
Autostereoscopy	No	No	No	Yes	Yes
Dual Projection	Yes	No	No	Yes	Yes
Full Screen	Yes	Yes	No	Yes	Yes

Cuadro 6.1: Tabla comparativa de los globos virtuales que incorporan soporte estéreo.

diferentes tipos de estéreo [15], entre ellos anaglifo y sistema *quadbuffer*.

Por otro lado para la aplicación Google Earth [16] existe Stereo GE Browser [17] que es un navegador estereoscópico basado en el plugin libre Google Earth Browser. Se caracteriza por utilizar tres instancias diferentes de Google Earth, una con la vista estéreo y otras dos correspondientes a cada ojo.

Otra opción para Google Earth es utilizar el visualizador de TriDef [18]. Forma parte de un conjunto de herramientas que sirve para visualizar en estéreo aplicaciones de terceros. En él se incluyen visualización por medio de anaglifos, autoesteroscopía

Figura 6.9: Escena de terreno de alta resolución en formato plano en la *CAVE* de la UPV.

y sistema *quadbuffer*.

También existen precedentes, aunque en menor medida, en aplicaciones SIG open source. Existen algunos globos virtuales libres que se han integrado en aplicaciones SIG [19], pero muy pocos presentan posibilidades estereoscópicas. Para Nasa World Wind [20] se ofrece un plugin libre para la visualización en anaglifo. Por otro lado existe una versión estereoscópica en desarrollo que soporta navegación *multitouch* [21].

Este soporte se está implementando en QuantumGis [22] y el globo virtual que están utilizando, osgEarth [23], pero el desarrollo no presenta un estado suficientemente avanzado para ser incluído en este trabajo.

GvSIG 3D es la aplicación de código libre que mayor soporte estereoscópico ofrece (ver Tabla 6.1), incluyendo multitud de modos de visualización estéreo y posibilidad de pantalla completa. Cabe destacar el soporte multiplataforma de la aplicación, funcionando en los sistemas operativos Linux, Windows y MacOSX.

Figura 6.10: Escena de globo virtual en la *CAVE* de la UPV, el usuario percibe la sensación de poder coger el globo con sus manos.

6.3.1. Pruebas Realizadas

Con el objetivo de valorar la experiencia subjetiva del usuario se ha realizado una prueba en un congreso SIG (6as Jornadas Internacionales de gvSIG), en la que han participado 51 voluntarios, de ellos 39 hombres y 12 mujeres en un rango de edad comprendido entre 24 y 65 años. El 52 % de los participantes llevaban gafas o lentillas y uno de los participantes no fue capaz de percibir sensación estereoscópica por padecer estrabismo. De los 51 voluntarios, 37 habían visto alguna película 3D en salas de cine.

Para la realización de la prueba se prepararon tres equipos, cada uno de ellos con un dispostivo estereoscópico diferente. Se puede ver una imagen de los equipos en la Figura 6.12.

Por motivos de logística no se pudo hacer la prueba con el sistema de doble proyección en este congreso. Sin embargo también se han realizado algunos tests en el *PowerWall* del *Visionarium* de la Universidad Politécnica de Valencia (UPV) como se puede

Figura 6.11: El *Powerwall* de la UPV consiste en una sala con un aforo de 20 personas ideal para presentaciones o proyección de películas en estéreo. Formada por una pantalla de 2x5 metros, se divide en dos zonas: la zona de proyección y monitorización, y la zona de visualización, donde se sientan los espectadores.

ver la Figura 6.11.

Marco de las pruebas

En la primera parte de la prueba se mostraba un vuelo tridimensional sobre dos zonas diferentes (una urbana y una montañosa) en cada uno de los dispositivos (ver Figura 6.13). Después se ofrecía un cuestionario que constaba de 5 preguntas y en la mayoría de ellas se le pedía a cada participante que juzgara sus sensaciones en cada dispositivo con un valor númerico en un intervalo del 1 al 5 . En la Figura **??** se presenta un modelo del cuestionario ofrecido a los participantes.

Las preguntas son las siguientes:

Figura 6.12: Equipos utilizados en la prueba, todos ellos con la opción de pantalla completa activada, para aumentar la sensación de inmersión. De izquierda a derecha: sistema *quadbuffer*, anaglifos y pantalla autoestereoscópica con filtro lenticular.

Q1. *Valora en una escala de 1 a 5 la sensación de profundidad/relieve que estás experimentando (donde 1 representa ninguna profundidad y 5 mucha profundidad).*

Q2. *Valora en una escala de 1 a 5 la sensación de inmersión en cada tipo de tecnología (donde 1 representa nada inmerso y 5 totalmente inmerso).*

Q3. *¿Has experimentado sensación de mareo/malestar? valora la experiencia de 1 a 5 con cada tipo de tecnología (donde 1 representa nada mareado y 5 muy mareado).*

Q4. *Si has ido al cine en 3D, evalúa la experiencia comparando con las tecnologías vistas. (donde1 es peor experiencia y 5 es mejor experiencia).*

Q5. *¿Qué tipo de tecnología crees que tiene más utilidad en un entorno GIS? Ordénalas de menor a mayor (del 1 al 3).*

Figura 6.13: Participantes en el momento de la prueba. El sistema anaglifo puede ser observado por más de una persona a la vez (participantes con gafas blancas), igual que en caso de la pantalla autoestereoscópica. En el caso del sistema *quadbuffer* sólo se disponía de unas *shutter-glasses* (participante con gafas negras). Imagen de http://picasaweb.google.com/gvsigproject.

6.3.2. Resultados de la prueba

En esta sección se analizarán los resultados obtenidos en las pruebas. Para ello se ha calculado la media y la desviación típica para las respuestas a cada una de las preguntas del cuestionario. Los resultados se muestran en la Tabla 6.2.

Como se puede observar en las respuestas para la pregunta Q1, la tecnología *quadbuffer* causa una mayor sensación de relieve y profundidad en la mayoría de participantes, siendo la tecnología autoestereoscópica la que menos. Además las gafas de conmutación hacen que el usuario tenga una sensación de inmersión mayor que con los otros dispositivos.

En cuanto a la sensación de mareo o malestar que pudiera derivarse del uso de alguno de los dispositivos (pregunta Q3) cabe destacar que un 32 % de los participantes no tuvieron esta sensación con ninguno de los dispositivos. Del subconjunto de participantes que sí tuvieron este tipo de sensación, valoraron peor la tecnología de anaglifos. La desviación típica tiene un valor elevado porque aquellos que se sentían mareados adjudicaban puntuaciones cerca de la máxima sensación de mareo.

TechnologyQuestion	Q1	Q2	Q3	Q4	Q5
Anaglyph	3.82± 0.71	3.38±0.85	1.94±1.85	2.83±1.16	1.79±0.76
Autostereo	2.94±0.84	2.84±0,93	1.84±1.18	2.24±1.09	2.02±0.82
Shutterglasses	4.78±0.50	4.56±0.67	1.50±0.97	4.13±1.08	2.25±0.84

Cuadro 6.2: Tabla con resultados de media y desviación estándar de las respuestas del cuestionario.

De entre los participantes que habían acudido con anterioridad al cine en 3D (pregunta Q4), la mayoría valoró como mejor experiencia la que sintió con el sistema *quadbuffer*, dejando a la autoestereoscopía y a los anaglifos a un nivel muy parecido al que sintió en el cine.

Gran parte de los encuestados opinó que el sistema *quadbuffer* es que más utilidad podía tener en un entorno GIS (pregunta Q5), después la tecnología autoestéreo y por último los anaglifos.

Muchos usuarios se dieron cuenta de que el sistema de anaglifos distorsiona los colores de la ortofoto, con lo que la sensación de inmersión se reduce. Aunque también se sorprendían de la calidad del efecto en este dispositivo, que creían mucho peor por ser el sistema más antiguo.

De los comentarios recibidos al dorso de la encuesta se puede extrapolar que la pantalla autoestéreo es la que más impresionó a los encuestados, y a la que más futuro veían por el hecho de no tener que llevar gafas. Sin embargo, muchos de ellos coincidían en que estos dispositivos todavía deben mejorar para poder equiparar la sensación estéreo a sistemas como el *quadbuffer*.

Para la prueba en el *visonarium* de la UPV la muestra de usuarios fue mucho menor que en las otras pruebas y además no pudieron comparar la sensación de inmersión respecto de las otras tecnologías, por lo que no cabe un estudio estadístico para esta parte. No obstante los asistentes quedaron altamente satisfechos con las sensaciones percibidas.

7

Formatos para la visualización realista de ciudades

La representación de entornos urbanos en 3D ha sido objeto de investigación durante muchos años. Existen multitud de técnicas para generar este tipo de entornos, desde el diseño con herramientas CAD a la generación procedural de edificios y fachadas. La visualización sobre entornos GIS de modelos urbanos pueden aportan mayor comprensión en la planificiación urbana y en la evaluación de relaciones espaciales. En [24] se afirma que la integración del modelado tridimensional del CAD con datos provenientes de GIS le permiten al planificador tener una visión real del medio ambiente tanto en situaciones pasadas, presentes como futuras.

Los formatos que se consideran de mayor importancia en este momento son *CityGML* y *kmz*. Se procederá a estudiar primero *CityGML* por ser un estándar del Open Geospatial Consortium (OGC) frente al al formato *kmz* que pertenece a la empresa Google. Con la inclusión de estos dos formatos con manifiesta capacidad para almacenar modelos arquitectónicos altamente detallados se pretende potenciar la percepción de inmersión subjetiva del usuario, a la vez que mejorar la calidad visual de los contenidos que puede mostrar la herramienta gvSIG 3D.

En la actualidad gvSIG 3D ya soporta algunos de los formatos más aceptados en el mercado, como son *3DStudio* y *Collada* (Collaborative Design Activity) el formato de Sony para intercambio de datos 3D basado en *XML*. La importancia de este formato reside en que otros formatos de representación se basan en él, como por ejemplo *kmz*. De hecho *kmz* no es más que un archivo comprimido con un modelo en formato *Collada* y un archivo de georreferenciación.

7.1. cityGML

Como se dice en la página de la OGC *CityGML* quiere ser un formato basado en *XML* para almacenar, representar e intercambiar modelos virtuales 3D de entornos urbanos. Proporciona una forma de describir objetos considerando su geometría, topología, semántica y apariencia, y define cinco niveles de detalle. Se pretende que *CityGML* permita el empleo de modelos 3D de ciudades para la visualización y análisis en diversos ámbitos de aplicación: navegación a pie, simulaciones medio ambientales, gestión de datos urbanos, etc.

CityGML está implementado como un esquema de aplicación sobre *GML3* (Geography Markup Language 3) un estándar internacional para intercambio de datos espaciales.

La integración en gvSIG 3D comprende dos fases de desarrollo:

- Lectura y visualización de modelos *cityGML* con la herramienta gvSIG 3D.
- Posibilidad de análisis, edición y escritura de estos modelos con gvSIG.

En cuanto a la primera fase la inclusión del formato ha sido posible haciendo uso de la librería **libcitygml** [25], una librería en lenguaje c++ fácil de usar y con licencia LGPL. La librería es capaz de analizar sintácticamente los archivos *XML* y preparar la geometría resultante de manera que pueda ser explotada por aplicaciones de renderizado 3D. Esta preparación comprende tanto la organización jerárquica de la escena como la teselación de polígonos y optimizaciones sobre la geometría.

A partir de los trabajos realizados se ha logrado un primer prototipo de integración del formato cityGML para la visualización realista de entornos urbanos dentro de la

aplicación *gvSIG 3D*. En la siguientes sección se documenta el resultado de dicha integración y las posibilidades que ofrece.

Para esta integración ha sido necesario trabajar en dos líneas:

- Integración del formato cityGML en el núcleo de la extensión osgVirtualPlanets,
- Integración del formato cityGML en gvSIG 3D

7.2. Integración del formato cityGML en osgVP

Para una primera aproximación ha sido necesario integrar este formato en el núcleo de la aplicación, es decir, en su parte nativa. Esta integración se ha realizado a partir de un plugin de OpenSceneGraph basado en la librería libcitygml. A partir de esta librería se analiza sintácticamente el fichero cityGML y se crea el grafo de escena convenientemente optimizado para su representación gráfica.

El plugin acepta las siguientes opciones:

- names: Añade un texto con el nombre de cada objeto en la parte superior del propio objeto.
- mask: Añade una máscara que permite ocultar o mostrar el objeto tratado.
- minLOD: Mínimo nivel de detalle.
- maxLOD: Máximo nivel de detalle.
- optimize: Optimiza las geometrías de cada objeto del fichero cityGML para reducir el número de objetos instanciados.
- pruneEmptyObjects: Elimina objetos vacíos (por ejemplo con geometría no soportada).
- destSRS: Transforma la geometría al sistema de referencia especificado.

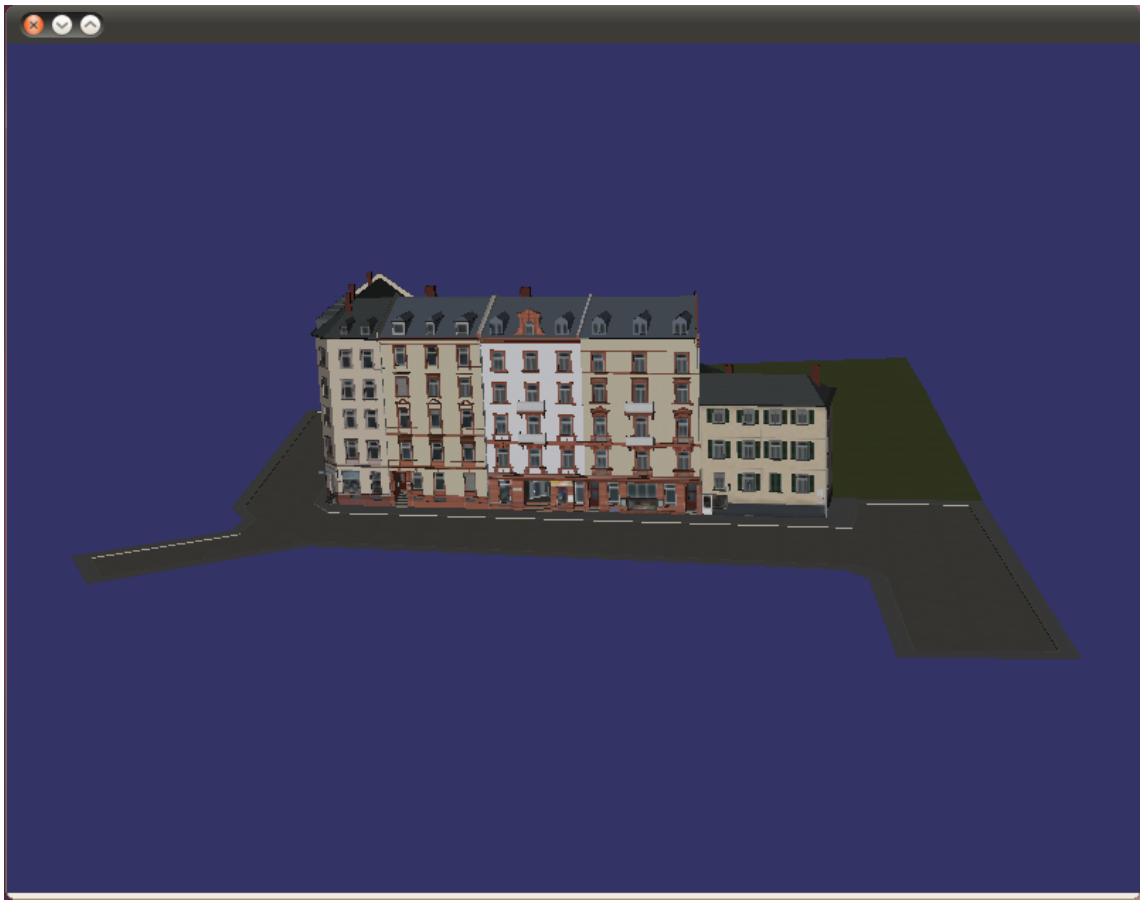

Figura 7.1: Primeras pruebas de integración del formato en la librería osgVirtualPlanets. En la imagen se puede observar el modelo Frankfurt Street Setting LOD3.

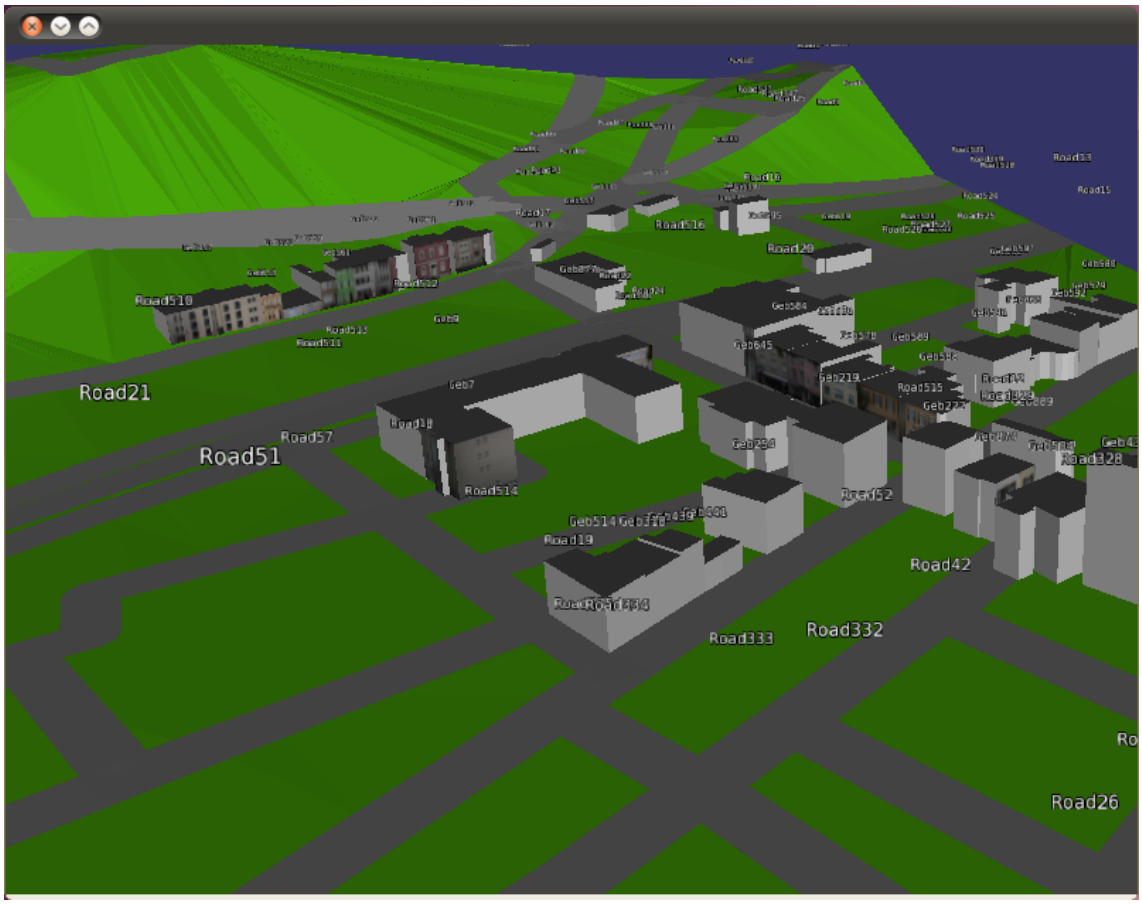

Figura 7.2: Modelo cityGML Koenigswinter Drachenfelsstrasse v1.0.0. En la imagen se muestra un texto con el nombre de cada objeto del modelo

7.3. Integración del formato cityGML en gvSIG 3D

La integración en gvSIG 3D comprende dos fases de desarrollo:

- Lectura y visualización de modelos cityGML con la herramienta gvSIG 3D.
- Posibilidad de análisis, edición y escritura de estos modelos con gvSIG.

Al terminar la actividad se ha conseguido concluir con éxito la primera fase de desarrollo (véase figura 7.3), dejando la segunda fase para trabajos futuros no enmarcados en el ámbito de este proyecto.

La integración de este formato se ha hecho de manera que el resto de herramientas de gvSIG 3D puedan acceder a él y aplicar las transformaciones (Véase Figura 7.4) que el usuario necesite. Aunque a la hora de persistir los cambios el modelo deberá ser guardado en formato binario nativo de osg(extensión de fichero .ive). La persistencia en el formato cityGML se afrontaría en una segunda fase de desarrollo que se ha comentado.

Para cargar estos modelos se ha utilizado el cargador de capas OpenSceneGraph, de modo que el modelo es analizado sintácticamente directamente en la parte nativa, por lo que la parte Java de la aplicación no tiene acceso a los objetos cargados. Esto tiene el inconveniente de no poder hacer edición no visual del modelo desde gvSIG, si no que se tendría que hacer uso de *software* específico a tal efecto. Por otro lado tiene la ventaja de que la generación del grafo de escena es mucho más rápida y el consumo de memoria es menor.

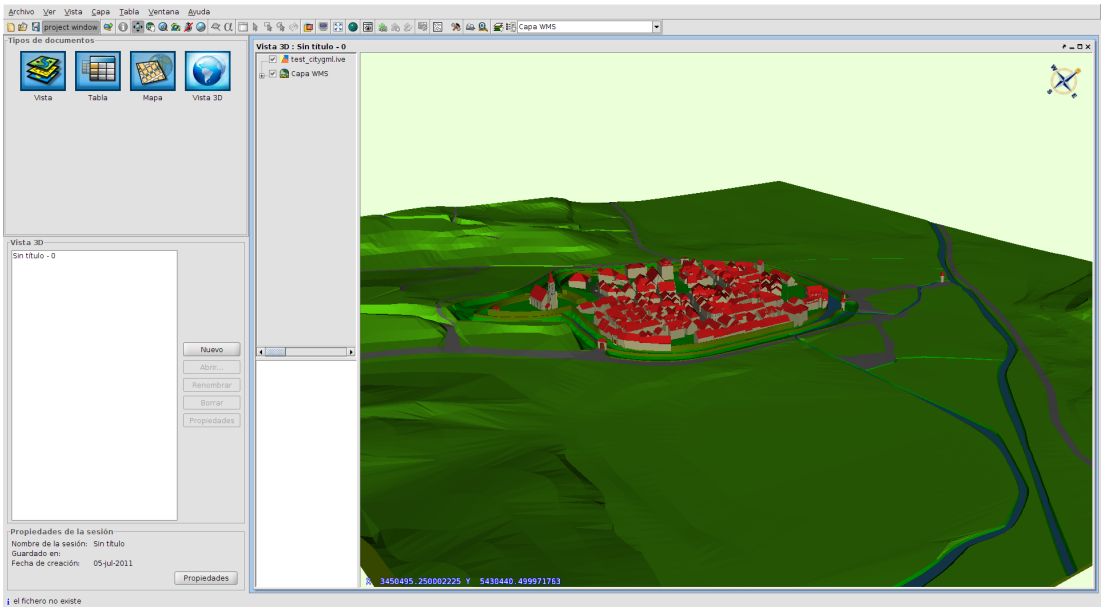

Figura 7.3: Test del modelo virtual 3D de la ciudad de Ettenheim en Alemania. Este dataset contiene la mayoría de las características soportadas por cityGML (Building, ReliefFeature, WaterBody, Road, CityFurniture, PlantCover, GenericCityObject) y está en el nivel de detalle LOD3.

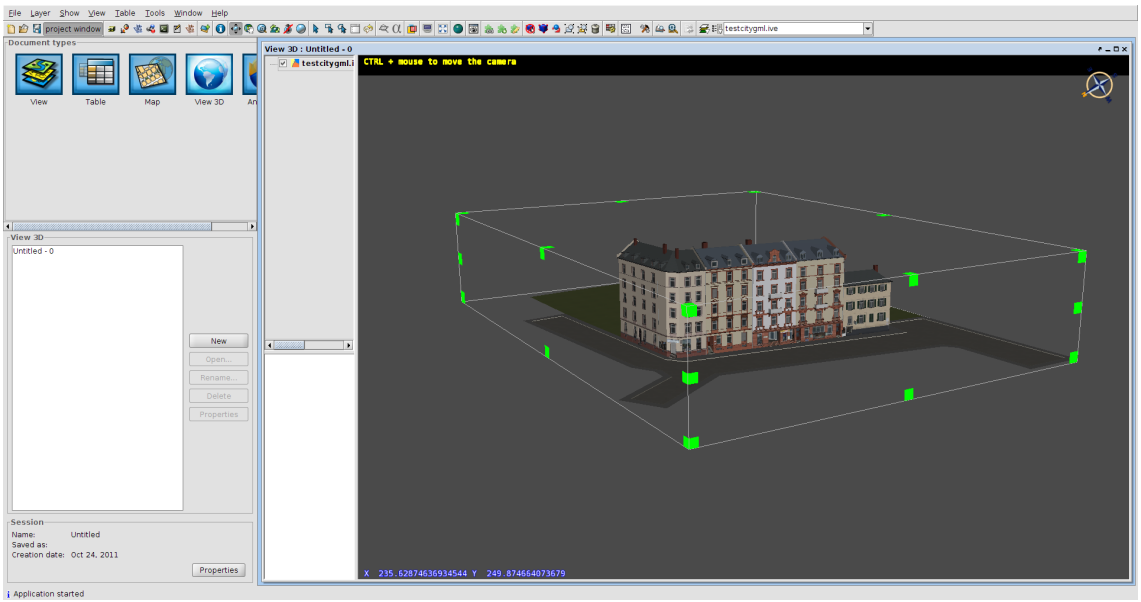

Figura 7.4: Modelo cityGML Frankfurt Street LOD3 siendo editado por la aplicación gvSIG 3D. La caja de inclusión que se muestra permite realizar transformaciones afines con el modelo como son escalar, rotar y desplazar. Así mismo también están disponibles las opciones de agrupar y desagrupar objetos para su edición.

8

Resultados

En esta sección se va a mostrar cómo la implementación de simbología avanzada en gvSIG 3D ha servido a otros miembros del consorcio para mostrar los resultados de sus investigaciones. En la mayoría de los casos estas investigaciones quedan fuera del ámbito de este trabajo de fin de máster.

Las primeras dos imágenes cedidas por "The Institute of Applied Computing Community Code"(IAC3) tienen que ver con la geovisualización tridimensional de fenómenos físicos. Y de cómo un campo electromagnético se distribuye y colisiona contra elementos urbanos, en este caso la catedral de Palma. Para la visualización se ha hecho uso de PointSprites y el modelo de la catedral está en formato Collada, también soportado por gvSIG 3D.

Por otro lado Indra y la Universidad Politécnica de Madrid están trabajando tanto en la representación tridimensional de edificios como en la visualización de datos 3D producidos por modelos atmosféricos y de calidad del aire. En este caso cada dato 3D viene representado por una esfera, cuyo color varia dependiendo del valor del dato. Además de las esferas, se han probado otros elementos de representación disponibles dentro de gvSIG como las polilíneas, aunque estos elementos en principio carecen de aspecto tridimensional, los resultados obtenidos para la visualización de datos de rejilla tridimensionales son bastante buenos y pueden ser otra forma de representar los resultados de los modelos atmosféricos junto a los edificios dentro de la plataforma

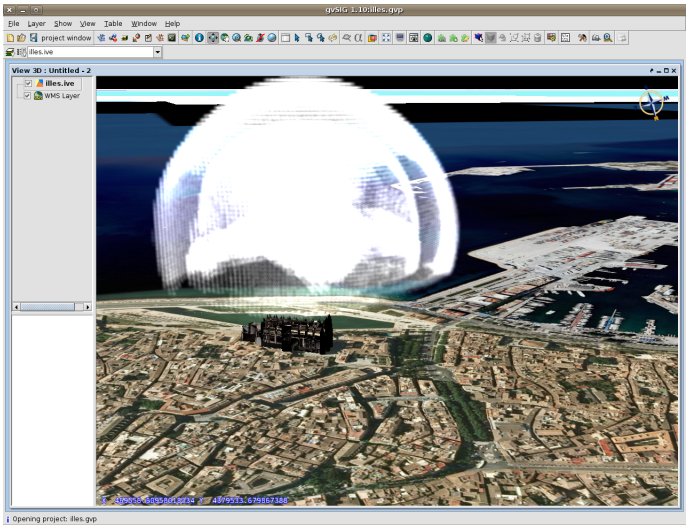

Figura 8.1: Vista de la catedral de Palma de Mallorca mostrando la expansión de un campo electromagnético en un determinado instante. La simbología utilizada se basa en PointSprites con transparencia.

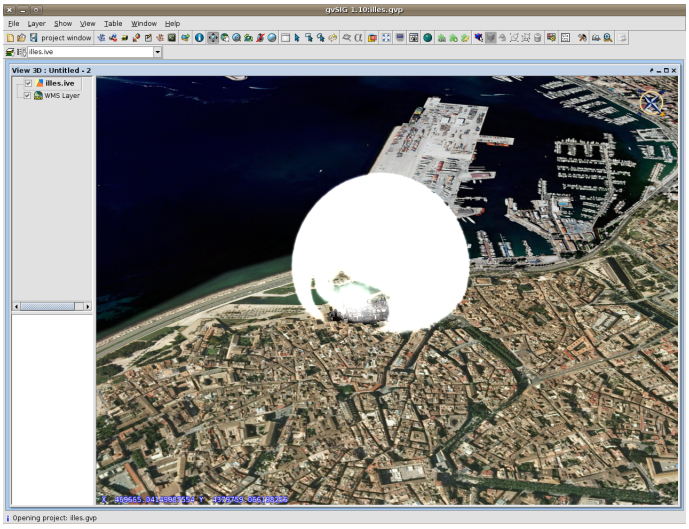

Figura 8.2: Vista ampliada del mismo campo electromagnético en un instante diferente.

gvSIG 3D.

Dentro del propio grupo de investigación del Instituto Universitario ai2 también se ha utilizado la simbología implementada en otra línea de investigación del proyecto: la multirresolución de datos masivos vectoriales. Hasta el momento se han utilizado los símbolos de tipo punto, muy comunes a la hora de representar datos en formato LiDAR o de escáner 3D.

Figura 8.3: Conversión de un modelo urbano en formato no estándar al modelo cityGML, soportado por gvSIG 3D

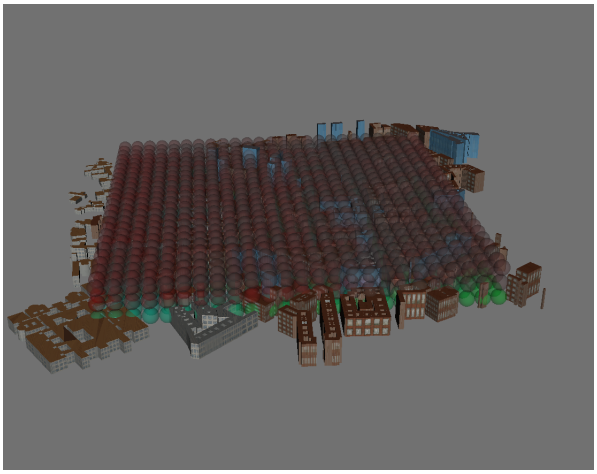

Figura 8.4: Representación de datos de temperatura en gvSIG 3D con 3 niveles verticales donde se observan los mayores valores(rojo) en las capas altas y los menores en la capa más baja(verde)

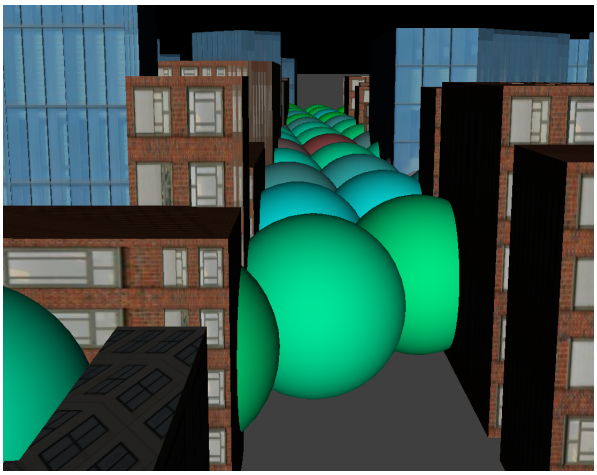

Figura 8.5: Misma representación que la Figura anterior pero hacienda un zoom sobre una calle para mostrar las posibilidades de navegación y zoom que puede realizar el usuario para explorar los resultados en gvSIG-3D.

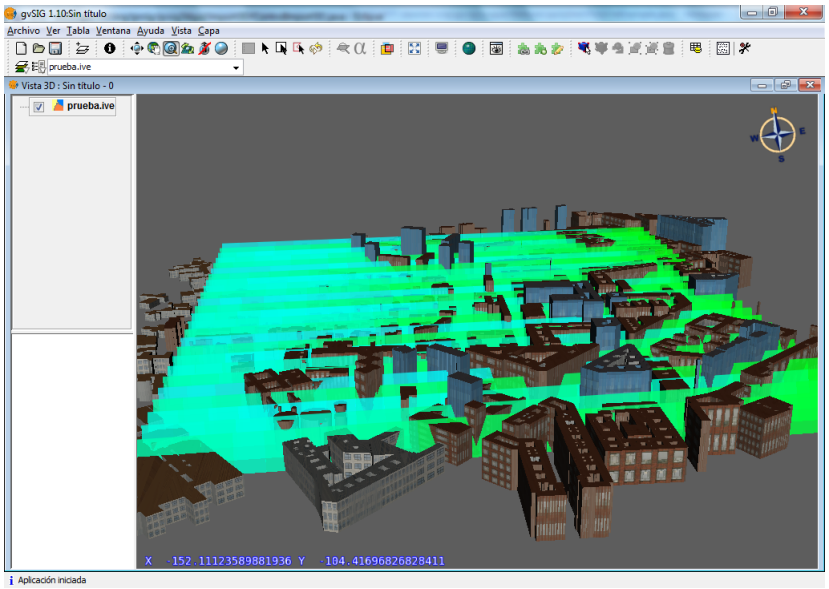

Figura 8.6: Representación de datos de temperatura en gvSIG 3D con 1 nivel vertical (superficie) mediante elementos "Polyline". Los colores verdosos representan las zonas de temperatura más baja, y los azules se corresponden con valores de temperatura medias, reservando los rojos para temperaturas máximas, que en este caso al ser un sólo nivel no aparecen.

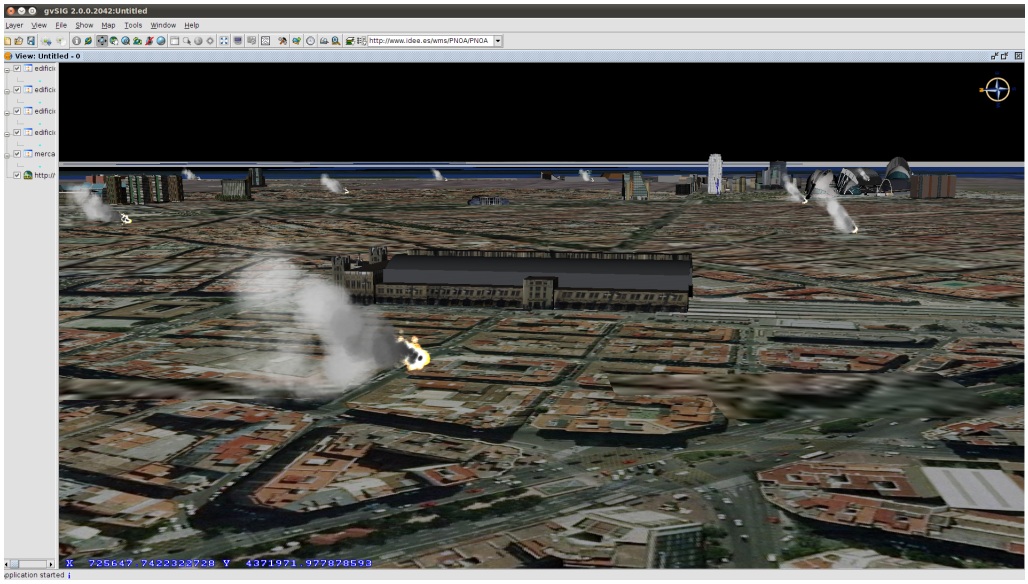

Figura 8.7: Resultado de la integración de la simbología avanzada mediante sistemas de partículas mostrando una capa de incendios en la ciudad de Valencia. También se muestran edificios en formato Collada también soportados por la extensión 3D de gvSIG

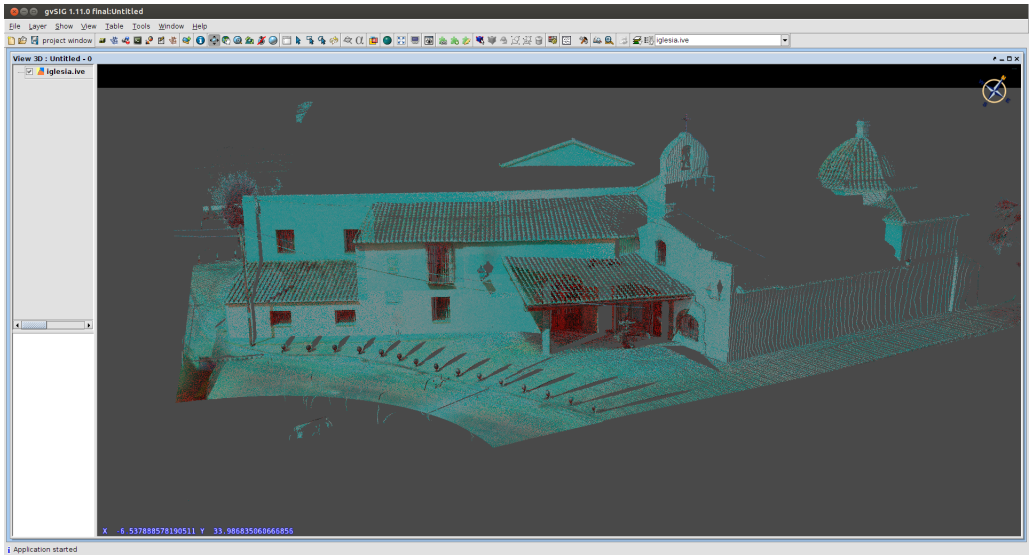

Figura 8.8: Visualización de una nube de puntos obtenida a partir de un escáner 3D.

Figura 8.9: Multirresolución de puntos con osgVP de a partir de una capa LiDAR, el framerate no se ve afectado por la cantidad de puntos.

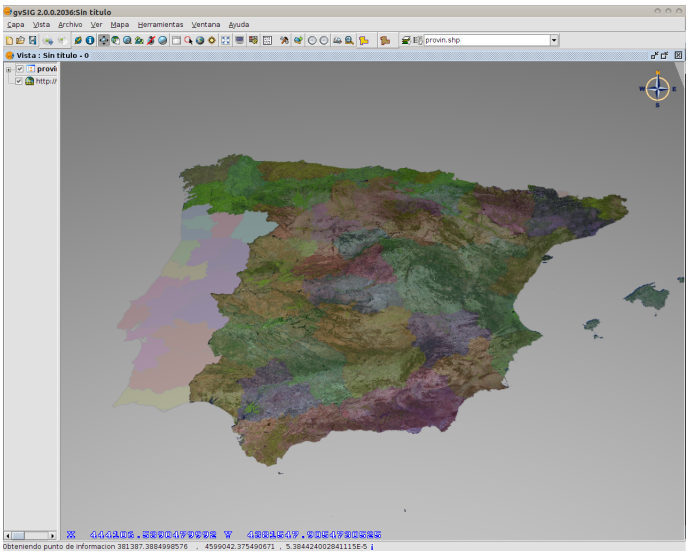

Figura 8.10: Provincias de España y Portugal rasterizadas con transparencia sobre una capa de ortofotografía. Esta imagen salió publicada en el diario El mundo en un artículo sobre España Virtual.

9

Difusión

Durante la ejecución del proyecto también se ha llevado un trabajo de difusión del mismo que ha abarcado muy diversos ámbitos, desde la creación de un blog y un canal de youtube a ponencias en congresos y workshops, así como cursos a nivel usuario en el centro de formación posgrado de la UPV.

También se debe tener en cuenta que todo el software generado ha sido liberado mediante licencia GNU/GPL, por lo que está a disposición del público general de manera gratuita y el código es accesible para cualquier usuario que desee modificarlo o simplemente saber cómo funciona. A día de hoy gvSIG 3D ha sido descargado más de 10.000 veces en los diferentes sistemas operativos para los que se han generado instaladores: Windows, MacOSX y Linux.

9.1. El blog de gvSIG 3D

El blog ha servido principalmente como escaparate de los avances obtenidos en el proyecto. También se ha hecho uso de este espacio para colgar tutoriales y videos para aprender a utilizar la herramienta, así como de medio de comunicación con la comunidad de usuarios. Como se puede ver en las estadísticas ha tenido una gran aceptación.

Para darle más empaque al blog utilizamos un canal de youtube donde subíamos

los vídeos que íbamos generando. Este canal tiene hasta la fecha 27.911 reproducciones. (http://www.youtube.com/user/gvSIG3D)

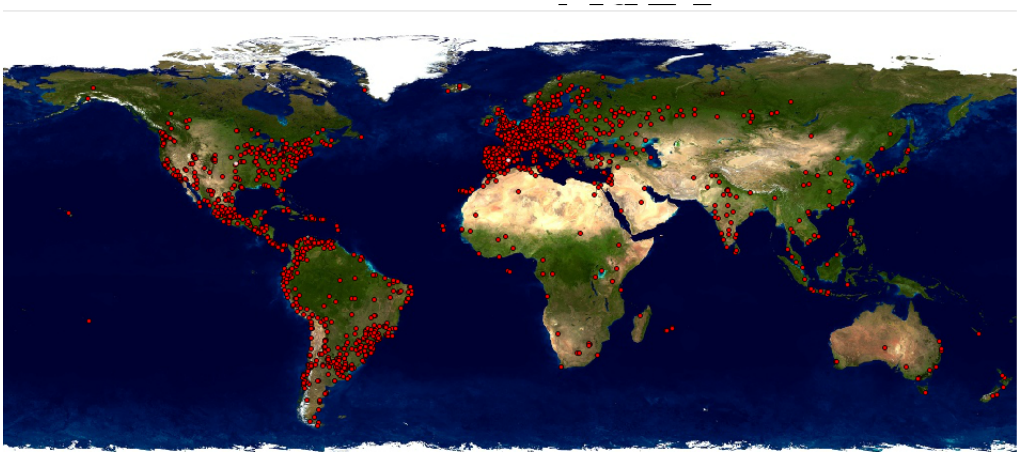

Figura 9.1: Mapa de visitas al blog del proyecto gvSIG 3D (http://gvsig3d.blogspot.com), usuarios de 124 países diferentes han visitado esta web en cerca de 50.000 visitas recibidas

9.2. Congresos, workshops y cursos

Durante los dos años en los que se ha desarrollado el proyecto, hemos tenido la oportunidad de asistir como ponentes a varios eventos de la comunidad de gvSIG así como a workshops del proyecto España Virtual, donde se explicaban los avances en cada iteración del proyecto. Concretamente hemos participado en las 5as, 6as y 7as jornadas internacionales de gvSIG explicando los desarrollos que se llevaban a cabo con el resto del equipo del Instituto Universitario ai2.

Se han impartido diferentes cursos a nivel usuario, tanto en el centro de formación posgrado como en la escuela de topografía de la UPV, donde se dio una charla junto con el profesor Jesús Palomar.

En el centro de formación posgrado se han llevado a cabo dos ediciones del curso "GVSIG AVANZADO. GEOPROCESAMIENTO Y VISUALIZACIÓN CON SEXTANTE Y GVSIG 3D", de 30 horas de duración.

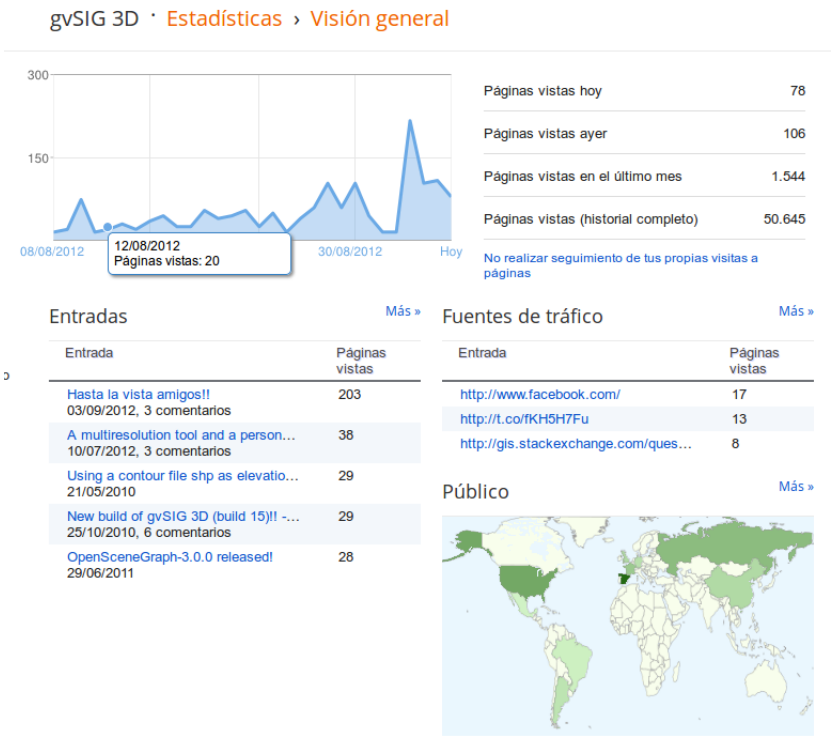

Figura 9.2: Estadisticas del blog gvSIG 3D

Figura 9.3: Curso para profesores del Este de Europa en la escuela de topografía de la UPV

9.3. Publicaciones

Los desarrollos previos a este trabajo de fin de master fueron publicados en las actas del Congreso Español de Informática Gráfica (CEIG) en particular:

Figura 9.4: 6as jornadas internacionales de gvSIG, explicando los avances en gvSIG3D

- Representación Vectorial 3D en un Sistema de Información Geográfica Euro-Graphics Congreso Español de Informática Gráfica (CEIG 2008) (ISBN 978-3-905673-69-2)
- Edición de Escenas 3D Sobre OpenSceneGraph EuroGraphics Congreso Español de Informática Gráfica (CEIG 2008) (ISBN 978-3-905673-69-2)
- Visualización Esférica de Terreno 3D para Sistemas de Información Geográfica EuroGraphics Congreso Español de Informática Gráfica (CEIG 2008) (ISBN 978-3-905673-69-2)
- Edición y Visualización de información vectorial en aplicaciones GIS Editar EuroGraphics Congreso Español de Informática Gráfica (CEIG 2009) (ISBN 978-3-905673-72-2)
- OsgFirefox: una extensión para la visualización web interactiva escenas 3D EuroGraphics Congreso Español de Informática Gráfica (CEIG 2010) (ISBN 978-84-92812-50-9)

A raíz del propio desarrollo del trabajo de fin de master se ha escrito un artículo con el siguiente título: *Comparative study of stereoscopic techniques applied to a virtual globe*. En este momento el artículo consta como aceptado y pendiente de publicar en "The Cartographic Journal - The World of mapping".

10

Conclusiones

En este trabajo de fin de máster se han realizado diversas implementaciones para mejorar en el ámbito de la visualización tridimensional de elementos georreferenciados. Importantes empresas y grupos de investigación de prestigio han hecho uso de las herramientas implementadas para mostrar los resultados de sus investigaciones en un proyecto CENIT de alto presupuesto.

En el campo de la simbología avanzada se han llevado a cabo los desarrollos necesarios para mostrar los elementos vectoriales de nuevas maneras, pudiendo dar un mayor realismo a la escena o utilizando simbología animada como en el caso de los sitemas de partículas. También mediante la nueva arquitectura en la que se utiliza el paradigma *pipes and filters* se ofrece una manera de caracterizar estos elementos de manera fácil y agilizando su visualización, permitiendo que se puedan mostrar a la vez millones de elementos sin que el rendimiento gráfico se resienta.

En cuanto a la estereoscopía existen métodos que nos permiten conseguir este efecto a partir de geometría tridimensional, no siendo necesario el costoso proceso de planificar vuelos fotogramétricos y pudiendo incluir elementos que no están en la ortofoto. Algunos de estos métodos han sido incluídos en gvSIG 3D, que se presenta como la alternativa de código libre más completa en cuanto a modos de estereoscopía.

A partir de los resultados obtenidos del cuestionario se puede inferir que la tecnología estereoscópica que mejores resultados ofrece en este momento en un globo

virtual es el sistema *quadbuffer*. Los participantes tuvieron mayor sensación de profundidad, de relieve y de inmersión en la escena. Muchos de los que habían acudido con anterioridad a una sala de cine en 3D la calificaron como mejor experiencia. También obtuvo el mejor resultado entre aquellas personas que sintieron malestar o mareo, siendo la tecnología de anaglifos la que mayores problemas causaba.

También se han realizado pruebas de funcionamiento con algunos disposivos que no pudieron ser incluidos en el estudio con usuarios por motivos de logística. Estos dispositivos son un *PowerWall* y una infrastructura de tipo CAVE, reseñando las principales ventajas e inconvenientes que se han encontrado.

Por otro lado, la tecnología que más impactó a los usuarios fue la autoestereoscopía, por el hecho de no tener que llevar gafas para sentir el efecto. Sin embargo la mayoría coincidió en que estos dispositivos deben mejorar para llegar a la altura del sistema *quadbuffer*. La sensación de profundidad y de inmersión es bastante menor, aunque el margen de mejora es grande. Con el aumento del número de vistas que puedan soportar las pantallas autoestereoscópicas es bastante posible que la valoración de los usuarios mejore.

Otras líneas de desarrollo en este trabajo han ido encaminadas a estudiar los formatos *citygml* y *kmz* para ser incluidos en gvSIG 3D. Se ha desarrollado un prototipo de integración en forma de visualizador de archivos de *cityGML*. Se ha procedido a estudiar primero *citygml* por ser un estándar del Open Geospatial Consortium (OGC) frente al al formato *kmz* que pertenece a la empresa Google. Con la inclusión de estos formatos con manifiesta capacidad para almacenar modelos arquitectónicos altamente detallados se pretende potenciar la percepción de inmersión subjetiva del usuario, a la vez que mejorar la calidad visual de los contenidos que puede mostrar la herramienta gvSIG 3D.

Por último todo el código resultante ha sido publicado como OpenSource, llegando a descargarse más de 10.000 veces la aplicación que incorpora las mejoras llevadas a cabo en el ámbito del trabajo. También se ha hecho un trabajo de difusión de los logros obtenidos que ha comprendido desde la elaboración de un blog con más de 50.000 visitas a la publicación de resultados en congresos nacionales como ponencias en workshops. También se han llevado a cabo cursos de formación en el centro de formación postgrado de la UPV. Finalmente se ha aceptado y pendiente de publicar un artículo a nivel internacional en *The cartographic Journal - The world of mapping*.

Bibliografía

[1] Jürgen Döllner Oliver Kersting. Interactive 3d visualization of vector data in gis. In *Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 107–112, 2002. Disponible en: http://portal.acm.org/citation.cfm?id=585147.585170.

[2] M Schneider, M Guthe, y R Klein. Real-time rendering of complex vector data on 3d terrain models. In *In Proceedings of The 11th International Conference on Virtual Systems and Multimedia*, pages 573–582, 2005. Disponible en: http://cg.cs.uni-bonn.de/aigaion2root//attachments/schneider-2005-vector.pdf.

[3] ESRI. Arcgis 3d analyst, 1999. http://www.esri.com/software/arcgis. Disponible en: http://www.esri.com/software/arcgis.

[4] Robert Osfield y Don Burns. Openscenegraph, 1999. http://www.openscenegraph.org. Disponible en: http://www.openscenegraph.org.

[5] Erich Gamma, Richard Helm, Ralph Johnson, y John Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series, 1995.

[6] Jesús Zarzoso, María Ten, Jordi Torres, Rafael Gaitán, y Javier Lluch. Edición vectorial de escenas 3d sobre openscenegraph. In *CEIG 2008: Congreso Español de Informática Gráfica*, pages 257–260. Eurographics Association, 2008.

[7] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, y Michael Stal. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Wiley, 1996.

[8] Estándar multiresolución 3d. Documento del Proyecto CENIT España Virtual, 2011. Consorcio España Virtual.

[9] E.5.1.7 Informe de avance en la investigación en Componentes de visualización y navegación 2D, 3D y 4D. Entregable del Proyecto CENIT España Virtual, 2011. Consorcio España Virtual.

[10] Instituto de Automática e Informática Industrial AI2. gvsig3d, 2008. Disponible en: http://www.osor.eu/projects/gvsig-3d.

[11] Conselleria d' Infraestructures i Transports. gvsig, 2003. Disponible en: http://www.gvsig.gva.es/.

[12] Charles Wheatstone. Contributions to the physiology of vision.part the first. on some remarkable, and hitherto unobserved, phenomena of binocular vision. *Philosophical Transactions of the Royal Society of London*, 128:371–394, 1838.

[13] M. Boulos y L. Robinson. Stereoscopic 3-d solutions for online maps and virtual globes. *International Journal of Health Geographics*, 8:59, 2009.

[14] Esri. Analyst 3d extension for arcgis, 2008. Disponible en: http://www.esri.com/software/arcgis/extensions/3danalyst/index.html.

[15] Esri. Viewing in stereo in arcscene, 2010. Disponible en: http://webhelp.esri.com/arcgiSDEsktop/9.3/index.cfm?TopicName=Viewing in stereo in ArcScene.

[16] Google. Google earth, 2005. Disponible en: http://www.google.com/earth.

[17] Masuji Suto. Stereoge browser, 2009. Disponible en: http://www.stereo.jpn.org/eng/stge/stbrhelp.html.

[18] TriDef. Google earth in stereoscopic 3d, 2010. Disponible en: http://www.tridef.com/promotions/google-earth.html.

[19] Mathias Walker y Pirmin Kalberer. Comparison of open source virtual globes. In *FOSS4G2010*, 2010. Disponible en: http://2010.foss4g.org/presentations$_s how.php?id = 3690$.

[20] National Aeronautics y Space Administration (NASA. Nasa world wind, 2004. Disponible en: http://worldwind.arc.nasa.gov/java/.

[21] Johannes Schöning y Florian Daiber. Multi-touch interaction with nasa world wind, 2010. Disponible en: http://ifgi.uni-muenster.de/ j_scho09/world-wind-touch/Home.html.

[22] Gary Sherman. Quantum gis, 2002. Disponible en: http://www.qgis.org.

[23] Pelican Mapping. osgearth, 2008. Disponible en: http://www.osgearth.org/.

[24] Van Dipten y Van Klaveren. The surplus of virtual reality in urban planning. *European Conference and Exhibition on Geographical Information*, 1996. Disponible en: http://www.osgearth.org/.

[25] BRGM. libcitygml, 2008. Disponible en: http://code.google.com/p/libcitygml/.