UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática

Universitat Politècnica de València

Escenarios Virtuales Interactivos 3D

Proyecto Final de Carrera

Ingeniería Técnica Informática de Sistemas

Autor: José Vicente Mora Espí

Director: Vicente Pelechano Ferragud

25/07/12

Autor: José Vicente Mora Espí
Escuela técnica superior de ingeniería informática

Resumen

La Ortofotografía ha llegado a nuestras vidas aunque muchos no sepan siquiera lo que significa, esta técnica se basa en realizar fotografías aéreas que posteriormente son tratadas, mejoradas y/o modificadas con el fin de desarrollar, en sus inicios planos precisos del terreno o verificar las fronteras y lindes. Con la aparición de técnicas más avanzadas como Lidar se puede, además de realizar ortofotografías, realizar un barrido del terreno para su posterior tratamiento con el que conseguir un terreno en 3D casi perfecto y con una precisión asombrosa.

Amparándonos en estos dos sistemas y mediante la investigación trataremos de desarrollar una aplicación en JAVA la cual sea capaz de personalizar un globo terráqueo al máximo posible, mediante la inserción de puntos de interés, ortofotos personalizadas en mayor o menor calidad y modelos digitales del terreno que pueden ser adquiridos mediante licencias de uso o mediante otros canales gratuitos y legales. Para la realización de este proyecto se contará con la herramienta de desarrollo Eclipse y la API WorldWind, la cual es open source, que cuenta con potentes librerías ya implementadas para mostrar la información referente a cartografía u otros elementos. También se quiere realizar un estudio de las soluciones que existen actualmente en el mercado, como pueda ser el driver DielmoOpenLidar®, Google Earth, Google Maps o
Microsoft Maps. Se incidirá en sus virtudes pero también en sus defectos, comparando cuales son estos y tratando de mejorar en la medida de lo posible aquellos que estén en nuestro alcance.

Por último también se tratará la integración en web para dar una mayor libertad a la aplicación además de una mayor globalización y alcance al mismo, para ello se contará con un servidor web Apache y la integración de JAVA en web vía APPLET, estando toda la carga posible en el lado servidor, para que el usuario perciba el menor retraso posible en cuanto a carga de proceso.

Autor: José Vicente Mora Espí
Escuela técnica superior de ingeniería informática

Tabla de contenido

Índice de ilustraciones

Autor: José Vicente Mora Espí
Escuela técnica superior de ingeniería informática

Agradecimientos

Este proyecto ha sido desarrollado por mi persona José Vicente Mora Espí, pero no habría sido posible sin la ayuda de todas esas personas más o menos importantes, pero que sin duda han marcado mi vida en algún momento. En primer lugar quiero agradecer a mis padres José Vicente y María del Carmen, que son las personas más importantes en mi vida, agradecerles su dedicación, esfuerzo y comprensión a lo largo de mi vida para hacerme la persona que soy hoy en día, gracias a ellos he conseguido todo lo que tengo y me han apoyado incondicionalmente incluso en los peores momentos o cuando rechazaba su ayuda, sin ellos esto no habría sido posible.

También tengo que agradecer a mis hermanas Mamen e Inma, ellas también han sido una parte importante en mi vida, un ejemplo en algunos casos y un apoyo en otros muchos, a pesar de las típicas riñas entre hermanos y algún que otro encontronazo las quiero y les agradezco todos los detalles que han tenido hacia mí en algún momento de nuestras vidas.

No puedo olvidar tampoco a mi abuela Milagro, a mi abuelo Vicente, a mi tío Onofre o a mi tía Dolores que lamentablemente no se encuentran ya aquí para ver este proyecto, pero que también me enseñaron a ser la persona que soy, dándome su cariño y sus consejos, colaborando en mi educación como si de mis padres se tratara, a ellos se lo dedico también allá donde se encuentren.

Tampoco puedo olvidarme de mi yaya Dolores, quién también ha sido una persona importante en mi vida, ni de mis tíos Miguel, María y Juan Vicente, que me ayudaron cuando lo necesité y me aconsejaron cuando lo necesité.

No quisiera dejar de lado a todos mis amigos desde mi juventud, Paco, Jato, Enrique, Xema, etc. y todos mis compañeros de carrera, Víctor, Dani, “Larcos”, Carras, Eugenio, etc., pues con ellos he pasado grandes momentos y trabajado conjuntamente para aprobar esta carrera y llegar a este punto. Ni por supuesto a mi mejor amigo “Capri” allá donde esté. Y por supuesto a mi novia Vanessa, por aguantarme todos estos años.

También agradecer a la empresa Dielmo donde se empezó a gestar este proyecto.

Y por último pero no menos importante a todos los profesores que han contribuido en mi formación académica, en especial a Vicente Pelechano Ferragud, director de este proyecto.

A todos ellos muchas gracias.

Autor: José Vicente Mora Espí
Escuela técnica superior de ingeniería informática

I – Introducción

1.1 – Objetivos

Desde la aparición de Google Earth se ha desencadenado una proliferación de aplicaciones de "visores 3D" o globos virtuales facilitando a los usuarios la navegación y exploración de cualquier localización de la Tierra en formato 3D. A estas alturas, nadie puede negar la repercusión mediática de este tipo de aplicaciones y, sobretodo, el impacto diseminador de conceptos geográficos (capas, mapas, etc.) que ha tenido sobre la gran mayoría de usuarios, normalmente no especializados en el dominio geográfico.

En el ámbito del software libre, a los originales World Wind (.NET) y Ossimplanet se han venido uniendo iniciativas en otras plataformas como Marble KDE, nuevas versiones de WordWind (Java SDK), a las que últimamente se están añadiendo iniciativas españolas dignas de consideración, como el Capaware canario y el módulo 3D de gvSIG.

Los escenarios virtuales 3D ofrecen soluciones a proyectos aplicados sobre el entorno geográfico y cartográfico, permitiendo ofrecer una simulación de cómo es o cómo va a quedar el territorio. De esta forma se obtiene información visual privilegiada vinculada al territorio, con el detalle deseado, y añadiendo toda la información que sea necesaria, mediante el uso de marcadores de posición, infos, POI's.

Visualizar una representación fotográfica del territorio en 3D permite interpretar con mayor criterio cualquier parámetro de valoración.

Las empresas y organismos que manejan grandes bases de datos georeferenciados, pueden asignarlos a los lugares en los que se encuentra, disponiendo así de una presentación espacial de la información, consultable a través de una interfaz natural. De esta manera, la titularidad de una parcela, las características técnicas de una torre eléctrica o la fotografía de un paso a nivel, se

encuentran integradas en un escenario que es una réplica exacta del lugar. La posibilidad de integrar modelos 3D dentro de los escenarios permite valorar una obra civil o un complejo proyecto urbanístico en su conjunto, viendo de qué manera afecta al entorno. En este sentido se benefician los proyectos relativos a carreteras, aerogeneradores, urbanizaciones y en general todas aquellas actuaciones humanas de gran impacto territorial.

1.1.1 - LA VISUALIZACIÓN GEOGRÁFICA TRIDIMENSIONAL

No cabe duda de que la representación tridimensional del territorio abre nuevas posibilidades en el ámbito geográfico. Pero el 3D por sí solo no está justificado. Las acciones para la navegación por una escena tridimensional son más complejas que las necesarias para la navegación en un plano. Cada aplicación de software ha resuelto de de manera distinta la manera de controlar la elevación, rotación y cabeceo del punto de vista, lo que requiere un aprendizaje por parte del usuario. Además, la sintetización en tiempo real de las escenas exige más cantidad de recursos, tanto de cálculo como de datos. Así, por ejemplo, la consulta de un mapa de carreteras suele ser más eficaz en una representación en dos dimensiones.

La representación tridimensional es conveniente cuando la visualización de una tercera magnitud, típicamente la elevación del terreno, resulta útil para la interpretación de los datos que se quieren mostrar. Se presentan a continuación algunos de los usos más comunes.

Para las rutas a pie o en bicicleta, la forma del terreno facilita la orientación y ayudar a calibrar su dificultad. En este caso, la ortofotografía juega un papel mucho más útil cuando se utiliza en combinación con los datos de elevación del terreno.

Ilustración 1-Comparación de representaciones 2D y 3D de un itinereario excursionista

Ilustración 2 - Itinerario 3D

La representación tridimensional también juega un papel importante en el terreno educativo, facilitando la exploración de los fondos oceánicos o la interpretación de la geología. En combinación con representaciones bidimensionales, puede facilitar la comprensión de conceptos como las curvas de nivel o las deformaciones proyectivas.

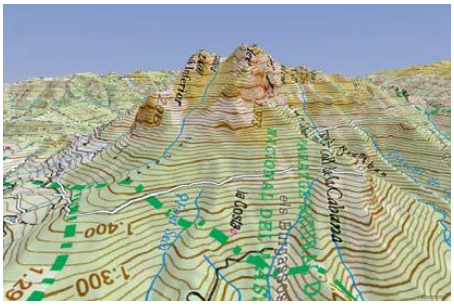

Ilustración 3 - Curvas de nivel

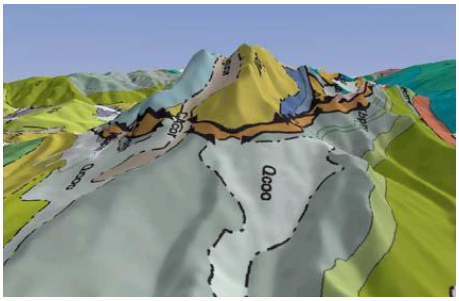

Ilustración 4 - Mapa geológico 3D

En lugar del terreno, también puede ser útil representar otra magnitud en la tercera dimensión. Donde las técnicas bidimensionales recurren a las isolíneas o las coropletas, en 3D podemos representar físicamente la medida como datos de elevación, extrusiones o símbolos proporcionales. Una buena discusión sobre estas técnicas se puede encontrar en [Bjorn Sandvic].

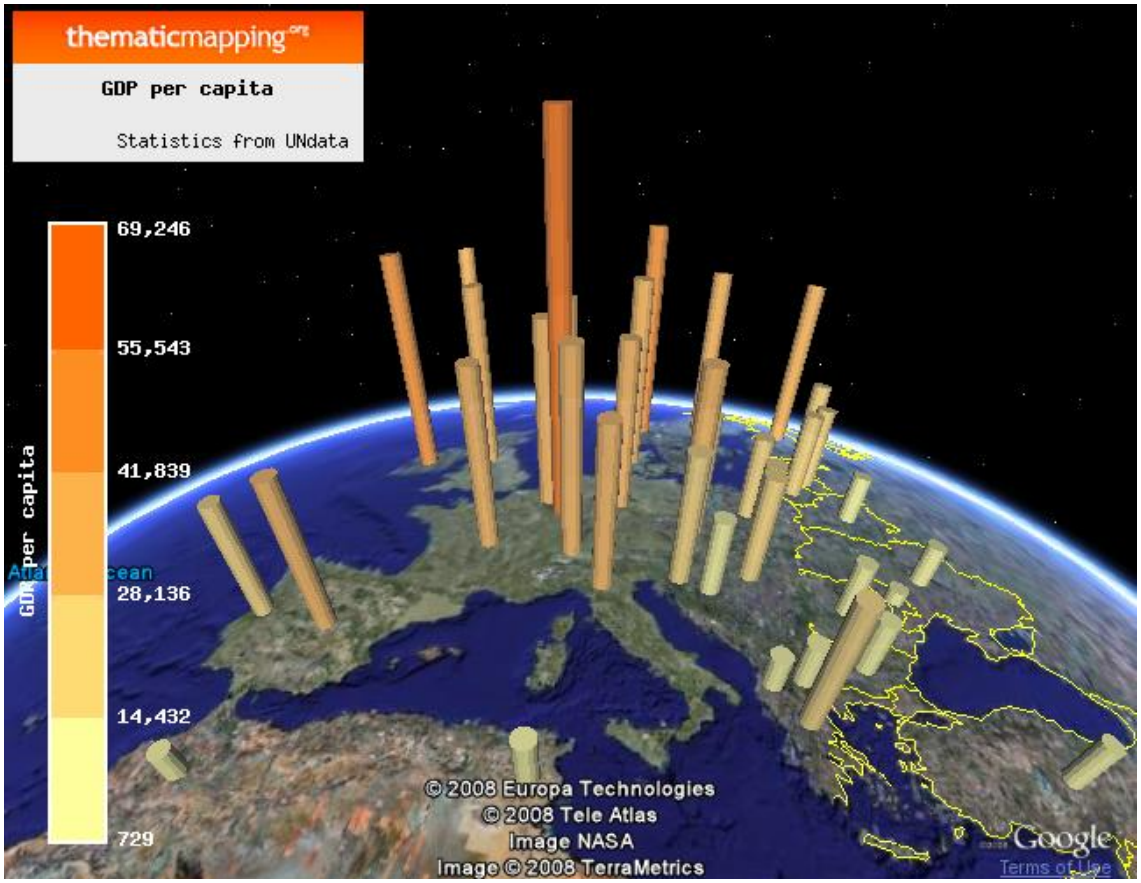

Ilustración 5 - Cloroptelas y extrusión combinadas en un mapa temático

Por último, la representación tridimensional permite la inclusión de modelos sintéticos de detalle para dar lugar a escenarios más realistas. Es el nexo entre el SIG, y el modelado tridimensional del CAD.

Técnicas de visualización e interacción

La visualización más común de un escenario 3D acaba siendo una proyección bidimensional sobre el plano de la pantalla, teniendo en cuenta la perspectiva. Existen técnicas estereoscópicas que, a partir de dos puntos de vista de la escena, permiten percibir la tercera dimensión.

La más simple es la representación en anaglifo, pues sólo requiere unas gafas pasivas coloreadas que actúan de filtro complementario para cada ojo. Al utilizar un filtrado cromático para discriminar entre las dos vistas necesarias, interfiere con la información de color de la propia imagen, con lo que produce mejores resultados sobre escenas en escalas de gris. La visualización durante largos períodos de tiempo no es saludable para la vista.

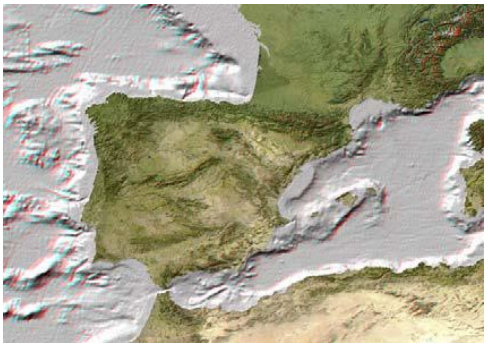

Ilustración 6 - Anaglifo de la península ibérica

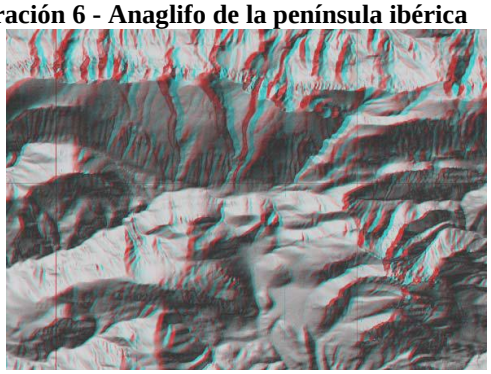

Ilustración 7 - Anaglifo de pena roca

Para obtener resultados de mayor calidad, deben utilizarse técnicas de filtrado distintas para cada ojo. Una solución económica para proyección colectiva consiste en utilizar dos cañones de proyección con sendos filtros polarizadores, dispuestos ortogonalmente el uno respecto al otro. Con una pantalla capaz de conservar la polarización de la luz, y unas gafas pasivas de bajo coste, se consigue el mismo efecto sin cansar la vista de la audiencia y conservando la información de color. Son necesarios dos proyectores de gran luminosidad y un software capaz de generar el par de imágenes para cada uno de forma sincronizada y en tiempo real. Esta técnica se utiliza en entornos académicos.

Ilustración 8 - Par estereográfico.

Para puestos de trabajo individuales se utiliza una técnica con gafas activas y monitor de rayos catódicos. Las gafas ocluyen la visión de cada ojo alternativamente, a una frecuencia sincronizada con la frecuencia de refresco del monitor, que muestra también el par de imágenes de forma alternativa. Es la técnica habitual utilizada para la restitución fotogramétrica y en el sistema de cine Imax 3D.

En cuanto a la interacción con escenarios 3D, como se ha comentado, los dispositivos comunes (ratón y teclado) resultan poco adecuados. Se han realizado pruebas que permiten manejar OssimPlanet con el mando Nunchuk de Wii, o pantallas Multitouch para moverse por Nasa WorldWind. Logitech comercializa 'ratones 3D' destinados al gran consumo, y ha publicado la API que permite integrar su uso.

1.1.2 - DATOS Y ESTÁNDARES

Modelos digitales de elevación del terreno

A nivel global, los datos de elevación que se utilizan son los de la Shuttle Radar Topography Mision (SRTM) del Jet Propulsión Laboratory (JPL) de la NASA7. Estos datos se tomaron en una misión espacial durante el mes de febrero de 2000 con una resolución de un arco de segundo (equivalente a unos 30 metros sobre el ecuador). Los datos son públicos, y accesibles por FTP8. Se dan en las siguientes resoluciones:

- Datos de base SRTM1 (< 30 m), para el territorio de los EUA.
- Promediados STRM3 (<90 m) y SRTM30 (<1000 m), para el resto del mundo.

Para España, el IGN dispone de los datos de elevación con un paso de malla de 25 metros, y otras Comunidades Autónomas, como Castilla y León, disponen de mallas de mayor precisión9. Por tanto, es interesante que los visores 3D puedan incluir datos de elevación de otras fuentes locales más precisas, y no sólo de SRTM.

Ilustración 9 - Comparativa de una escena con distintas fuentes de datos

La figura anterior ilustra la importancia de poder incorporar datos de elevación más precisos. En la izquierda, vista de Google Earth (ortofoto 1:5 000 del ICC y SRTM3). En el centro, vista con los datos por defecto en WorldWind (Landsat y SRTM30). A la derecha, WorldWind con datos propios (ortofoto 1:5 000 y MET 30x30 del ICC).

En general, existe aún una falta de uniformidad en el formato y la forma de acceso a estos datos, lo que dificulta su integración. Una solución es el uso del estándar de OGC Web Coverage Service, (WCS). La IDEE dispone de dicho servicio, que proporciona el modelo de elevación a 25 metros en formatos GeoTIFF, ASCII Grid y FloatGrid+ZIP10.

Pero el protocolo WCS adolece de la misma carencia que WMS: Su lentitud. Por ello es recomendable establecer una caché también para los datos de elevación. WorldWind publica de esta caché a través de un servicio web con los datos SRTM30. Se indica en los parámetros el nivel de detalle (L), y la fila (X) y columna (Y) de la tesela a recuperar. Las peticiones tienen un formato como éste:

Los datos devueltos son ficheros RAW de 512x512 muestras de elevación, donde cada muestra es un entero de 16 bits. Estos datos se comprimen en formato ZIP para ahorrar ancho de banda.

Las posibles implementaciones de una caché de terreno pueden tomar como referencia el servicio de NASA WorldWind, o utilizar el candidato a estándar de la OGC Web Map Tiling Service (WMTS), añadiendo los formatos mime propios de una cobertura.

Cartografía de base raster

La cartografía de base ráster se utiliza como textura para el terreno, y generalmente se usa una imagen de satélite o una ortofotografía. En cuanto a datos globales libremente disponibles, debemos recurrir nuevamente a la NASA. Existen los siguientes datos:

• Blue Marble Next Generation: Con una resolución máxima de 1 km/píxel, dispone de 12 coberturas para los 12 meses del año, capturadas durante el 2004.

• Landsat 7, con resoluciones de hasta 15 metros por píxel.

• Modis: Datos diarios con información ambiental, como la disposición de las nubes, con una resolución de hasta 250 m/píxel.

En cuanto a España, existe un servicio WMS del Plan Nacional de Ortofotografía Aérea12 (PNOA), con ortoimágenes de 50 cm./píxel.

La solución técnica para el acceso ágil a los datos pasa por una caché de teselas en forma de *quadtree*. Hay diferentes cachés públicas de teselas para los conjuntos de datos de la NASA.

Para crear una caché propia, es conveniente tener en cuenta la convención en cuanto al marco de referencia (EPSG:4326) y niveles de resolución. En las recomendaciones no oficiales TMS13 y WMS-C14 se puede encontrar bajo los epígrafes *unprojected profile* y *global-geodetic* respectivamente. En cuanto al protocolo de acceso, además de TMS, el propio de World Wind se puede considerar estándar de facto. TileCache15 es una buena herramienta para crear la cachés propias, pues será accesible con los tres protocolos. Se espera que la propuesta WMTS de OGC, una vez fijada como estándar oficial, unifique el criterio de acceso a estos datos.

Datos vectoriales y modelos de objeto

CityGML ha sido adoptado como estándar por OGC y realmente se trata de un esquema de aplicación (o perfil) de GML versión 3.1.1, diseñado para la representación, almacenamiento e intercambio de modelos de de ciudades virtuales en 3D (por ejemplo planeamiento urbano, arquitectura, etc.). Se basa en las primitivas geométricas disponibles de GML para extenderlas con una serie de propiedades topológicas, semánticas y de apariencia, propias para objetos urbanos en 3D, el contexto para el que ha sido concebido.

Una de las características más importantes de CityGML es su capacidad para contener datos vectoriales 3D complejos y georreferenciados junto a la semántica asociada a esos datos. En contraste con otros formatos 3D, CityGML se basa en modelo de información rico y de propósito general que va más allá del contenido gráfico y geométrico. Para trabajar en dominios específicos, CityGML proporciona también un mecanismo de extensión con el objetivo de enriquecer los datos con características identificables que permitan la interoperabilidad semántica en esos dominios.

KML (Keyhole Markup Language) versión 2.2 ya forma parte del grupo de estándares de OGC. Este formato se popularizó rápidamente con el lanzamiento de Google Earth como mecanismo para describir y publicar todo tipo de contenidos georreferenciados, como fotos, modelos 3D, panoramas, etiquetas, etc. sobre el globo virtual.

KML es un lenguaje relativamente sencillo (comparado con GML) cuyo objeto no es la descripción de un dato sino su visualización, en 2 o en 3 dimensiones, así como el control de la navegación del usuario sobre el mapa o el globo. KML es el estándar popular para el intercambio de datos vectoriales que todo visor 3D debería soportar.

COLLADA (COLLAborative Design Activity) es un esquema XML estándar y abierto gestionado por un consorcio sin ánimo de lucro llamado Khronos Group16. Al ser un formato basado en XML, resulta fácil de integrar en cualquier entorno. De hecho, su intención original fue servir meramente como formato de intercambio. Los objetos SketchUp contienen el modelo en Collada y las texturas asociadas a cada elemento, así como la georreferenciación del objeto.

Descripción de escenas

X3D (eXtensible 3D) es un estándar reconocido por ISO y por el consorcio Web3D. Se trata de un formato de almacenamiento vectorial 3D que permite ser codificado tanto en sintaxis VRML (aunque en desuso) y XML.

El diseño de X3D está basado en componentes y no todas las implementaciones de los visualizadores implementan todos los componentes. Un fichero X3D contiene metadatos indicando qué componentes son necesarios para una visualización correcta. Existen, por ejemplo, componentes definidos para CAD Geometry, Cube Map Environmental Texturing, Environmental Effects, Geometry2D, Geometry3D, Geospatial, Layering, NURBS; Particle Systems, Texturing 3D, etc.

X3D es una especificación muy ambiciosa y, como tal, tiene ventajas e inconvenientes. Su principal ventaja radica en la flexibilidad de su estructura de componentes, permitiendo que el formato X3D sea capaz de albergar

prácticamente cualquier novedad que aparezca en el campo gráfico. Como contrapartida, la implementación de un visor X3D que soporte un buen número de componentes es una tarea compleja, pudiendo impedir de esta forma su rápida diseminación y adopción por parte de la industria. Además, una característica interesante de X3D frente a otros formatos similares es que el usuario puede interactuar con una escena X3D mediante un API basada en JavaScript (Ajax3D).

Estándares OGC para la recuperación de escenas 3D

WTS19 (Web Terrain Server, borrador en versión 0.3.2) ofrece la posibilidad de recuperar una vista tridimensional de un área geográfica determinada. A partir de un punto de vista proporcionado en una petición, el servicio genera una vista 3D a partir del conjunto de datos contenidos o accesibles por el proveedor. La escena 3D devuelta es el resultado de la renderización de diferentes datos accesibles por el servicio, en el cual se pueden combinar tanto información ráster como vectorial. Sin embargo, WTS resulta poco atractivo para sistemas de visualización intensiva de escenarios 3D, tales como navegación en tiempo real, dado su protocolo de petición-respuesta y de la propia generación de la vista en la parte servidora. WTS es poco más que un servicio WMS donde se puede indicar un punto de vista para obtener una imagen en perspectiva.

Posteriormente a WTS, OGC lanzó el Web 3D Service20 (borrador versión 0.3.0) que ofrece la posibilidad de recuperar elementos gráficos dado un área geográfica. La principal diferencia con WTS radica en que la fase de renderización debe ser realizada en el propio cliente. Al igual que WTS, se basa en petición-respuesta, cuyos parámetros básicos describen el área geográfica a visualizar y las capas a recuperar. Cabe destacar, que Web 3D Service contempla la posibilidad de parametrizar las peticiones con información de entorno como hora del día, color o imagen de fondo, descriptores de estilo para las diferentes capas, etc., para conseguir un mayor realismo. Para ello se podrán definir diferentes fuentes de luz, fenómenos atmosféricos como niebla y lluvia. Los formatos típicos de recuperación de la información son X3D y su antecesor geoVRML.

1.1.3 - PROYECTOS DE SOFTWARE LIBRE

Nasa World Wind

Nasa World Wind21 es anterior a Google Earth, y hasta la fecha el proyecto de software libre más parecido a él, con lo que se ha convertido en una referencia.

Para el acceso a los datos de elevación y de textura remotos utiliza un protocolo propio de caché, muy simple, que popularmente ha tomado el nombre del proyecto, World Wind.

Se pueden añadir datos de elevación y de textura propios que cumplan con este protocolo, datos de textura que cumplan con el protocolo WMS, y toponimia y datos vectoriales siguiendo el protocolo WFS. Incluso pueden crearse planetas propios. Toda esta configuración se realiza mediante ficheros XML simples, sin necesidad de modificar el código fuente. También es posible la inclusión de plug-ins. Esta arquitectura abierta y basada en geoservicios estándar da una flexibilidad enorme a la aplicación, que puede ser adaptada a voluntad.

Las primeras versiones se desarrollaron utilizando la plataforma .NET y utilizan DirectX como librería gráfica, lo que hace a la aplicación totalmente dependiente del sistema operativo Windows. El desarrollo se produjo en el seno de la NASA por un solo programador, dando lugar a un código bien estructurado pero pobremente documentado. Llegada la versión 1.4.0, la NASA dejó de mantener la versión para .NET, que ha quedado en manos de una comunidad construida alrededor de WorldWindCentral. WorldWind .NET es un desarrollo ejemplar en cuanto al uso de estándares y modularidad, y pionero en sus orígenes, pero que se ha estancado por falta de liderazgo, y por uso de tecnologías de base privativas.

Tras el abandono de la versión .NET, la NASA se ha puesto a trabajar en un SDK basado en Java. En lugar de una aplicación final, se dispondrá de un componente y una API que permita integrarlo en otras aplicaciones. El desarrollo del núcleo continúa en manos de la NASA, que confía en publicar la primera versión estable del plugin a finales de 2009. Actualmente se encuentra en su versión 0.5.0, y ya ha sido utilizada en un buen número de aplicaciones que necesitan ir más allá de lo que Google Earth ofrece. Estas aplicaciones están reunidas en una galería de demos.

OssimPlanet

OssimPlanet es un proyecto desarrollado sobre Ossim (Open Source Security Information Management) y OpenSceneGraph, una librería en C++ que usa OpenGL como API para la renderización de escenas tridimensionales. El desarrollo de OssimPlanet está liderado por RadiantBlue Technologies y financiado por los organismos de defensa de los EUA, y puede ejecutarse en plataformas Linux, OSX, Windows y Solaris.

OssimPlanet destaca por su capacidad para cargar cualquier imagen ráster local en multitud de formatos y sistemas de referencia, pues utiliza GDAL para su transformación. Además, es capaz de descargar datos de servicios WMS o World Wind remotos. Está concebido para la visualización de grandes cantidades de datos ráster sin que tengan que ser preprocesados. También permite la incorporación de datos de elevación diversos, videos y ficheros KML además de compartir sesión y sincronizar la vista con otros clientes OssimPlanet remotos.

El visor no se caracteriza por su rapidez, pero si pero por su precisión y versatilidad, al no usar datos pretratados y hacer reproyecciones en tiempo real. Su público potencial es por el momento limitado

Marble KDE

Marble KDE25 es la visión opuesta a OssimPlanet. Forma parte del *KDE Education Project*. Es un cliente muy ligero, con versiones tanto para KDE como para Windows, que tarda muy poco en iniciarse y no requiere OpenGL. Además de los datos de la NASA, puede visualizar el callejero OpenStreetMap y, clicando sobre los topónimos, se puede acceder al artículo de Wikipedia de un lugar determinado. La vista del planeta puede ser esférica, plana (no proyectiva) o Mercator. Puede cargar datos adicionales de disco, en formatos GPX y KML.

Es en definitiva una aplicación de muy simple manejo con fines educativos. También existe en versión widget o control para integrarse en otros desarrollos.

Ilustración 10 - Imagen de MARBLE

Extensión gvSIG 3D

Esta extensión añade la capacidad de crear vistas 3D en gvSIG. Se puede configurar la escena a voluntad: Modelos de tierra plana o esférica, modelos de elevación, y cualquier capa convencional que pueda ser cargada en gvSIG, además de modelos de edificios tridimensionales. Permite trabajar en modo estereográfico o anaglifo, y crear animaciones. Actualmente se distribuye como versión alfa para gvSIG 1.9, y es bastante inestable.

El módulo 3D de gvSIG utiliza para la representación en pantalla el framework osgVP (por Virtual Planets), especialmente orientado a la visualización de información geográfica, y que actúa como wrapper de OpenSceneGraph para su utilización desde Java.

1.2 – Soluciones

De acuerdo a la problemática expuesta anteriormente nace este proyecto, mediante el uso del SDK NASA WorldWind, en su versión Java. Existe también una versión en .Net, pero esta ha sido abandonada en pos de una mayor portabilidad multiplataforma, ya que esta versión únicamente podía ser ejecutada bajo Windows, no obstante la versión en Java ofrece la posibilidad de ejecutarlo en cualquier S.O. que tenga instalada la JVM.

Con este programa pretendemos realizar una aplicación basada en WorldWind que incorpore las herramientas más habituales en cuanto a la elaboración, personalización y consulta en escenarios virtuales. Se elaborará un escenario virtual de prueba mediante la adición de datos geográficos en distintos formatos y mediante las distintas posibilidades que nos ofertará este programa, tales como mdt, ortofoto en modo local, servidores wms, shapefile en modo local, imágenes raster.

Otro de los problemas que encontramos es que worldwind trabaja con JOGL y los distintos objetos que se añaden se representan en 3D, para ello habremos de intentar hacer una representación en pantalla de los objetos lo más eficiente posible.

Ilustración 11 – Representación en worldwind de topónimos

Como vemos en la figura 1.3 no se aprecia del todo bien el nombre de algunos topónimos, mientras que otros podrían aparecer a una distancia de nuestro objetivo excesivamente grande como para ser de nuestro interés, produciéndose una superposición de unos con otros.

Plataformas

Cómo ya hemos comentado, la principal ventaja que nos ofrece Java, es la de desarrollar un producto que pueda ser utilizado independientemente del Sistema Operativo, por tanto nuestro producto será usable tanto en sistemas UNIX, cómo en sistemas Windows sin ningún tipo de problema. Además otro beneficio que nos ofrece Java, es la posibilidad de utilizar la aplicación en entornos Web, mediante la simple operación de transformar la aplicación de escritorio en un Applet de Java, de modo que no será simplemente independiente del Sistema Operativo sino que también podremos utilizarlo en cualquier ordenador con cualquier navegador que tenga instalada la JVM.

Principios de diseño

En cuanto a la filosofía de trabajo hay varios aspectos que han querido ser cuidados muy especialmente, a pesar de haber supuesto aumentar el esfuerzo:

- La documentación generada: Dado que el proyecto presenta unas características ideales para ser retomado y ampliado por otro alumno, la documentación se pretenderá que sea lo más clara posible, y extensa cubriendo de la mejor manera posible todas las fases y apartados del desarrollo. Además, se ha hecho hincapié en la descripción de la bibliografía, con unas breves explicaciones de cada una de las direcciones o libros consultados, así como un glosario de términos para clarificar cualquier duda. Todo ello queda recopilado en un CD-ROM, con los diferentes aspectos del proyecto (véase “Estructura del CD-ROM del proyecto”).

- La estandarización: El desarrollo, y muy especialmente la codificación se han realizado lo más estandarizadamente posible, pues al haber sido realizado por una sola persona se podría caer en unos códigos fuente ilegibles difíciles de mantener. Para ello se han seguido los consejos dados por [Hoff, 1997] y [Johnson, 1996].

- La reutilización: Se han creado documentos específicos para aquellos programadores que deseen utilizar cualquiera de las partes codificadas. No en vano la reutilización es seguramente la propiedad más destacable de un lenguaje como Java, orientado a objetos. Las partes del proyecto que se presten a la reutilización han sigo diseñadas conforme a ello, presentando una genericidad lo más amplia posible, además al ser un SDK incluye ya numerosas clases generadas por la propia NASA que representan diversos objetos que se han utilizado y podrán reutilizarse.

- La multiplataforma: Se ha buscado una portabilidad del código fuente a cuantas más plataformas mejor, este es el principal motivo por el cual el proyecto se ha desarrollado en JAVA, un lenguaje orientado a ser multiplataforma.

II – TÉCNICAS Y HERRAMIENTAS UTILIZADAS

2.1 – Orientación General

Este proyecto ha sido desarrollado bajo el paradigma de la programación orientada a objetos, lo cual es una característica propia del lenguaje utilizado, JAVA, esto nos proporciona una gran reutilización del código codificado, además de una gran claridad del mismo.

Dado que el SDK incorpora un gran número de clases ya implementadas, gran parte del trabajo se ha focalizado en analizar y comprender el funcionamiento de las mismas para poder trabajar con ellas, esto unido a la poca documentación sobre el programa, ha supuesto un gran esfuerzo en cuanto a tiempo ya que si bien es difícil programar determinadas cosas, más difícil es comprender el funcionamiento de algo ya programado por alguien. Afortunadamente en el foro dispuesto por la propia NASA para servir de apoyo, hemos podido solventar muchas de nuestras dudas y recibido ayudas de distintos usuarios.

El funcionamiento de worldwind se puede observar a grandes rasgos en las siguientes figuras.

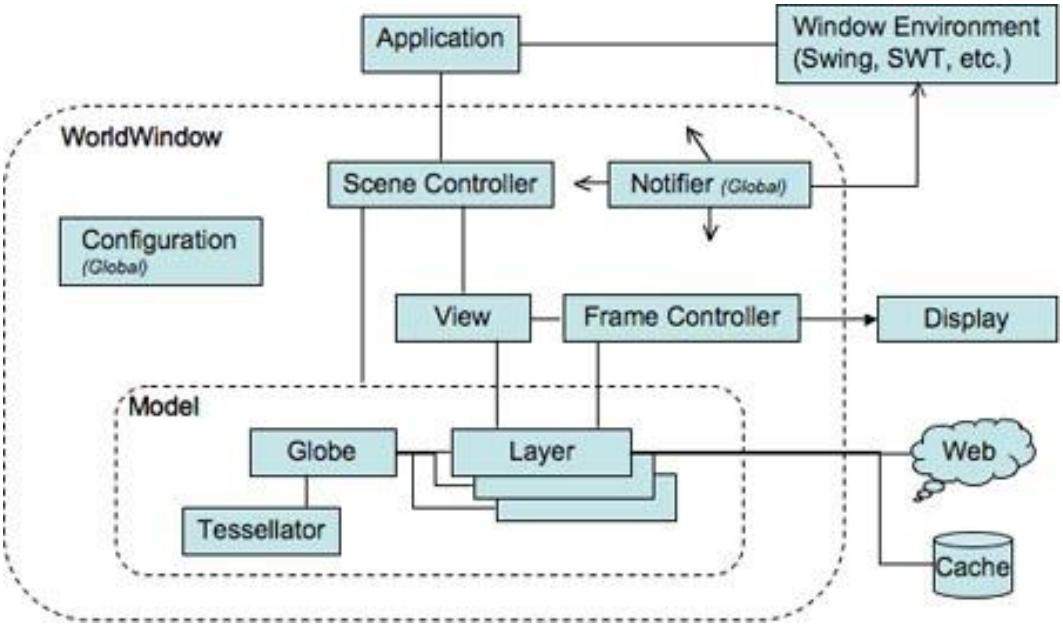

Ilustración 12 – Arquitectura del sistema WORLDWIND JAVA

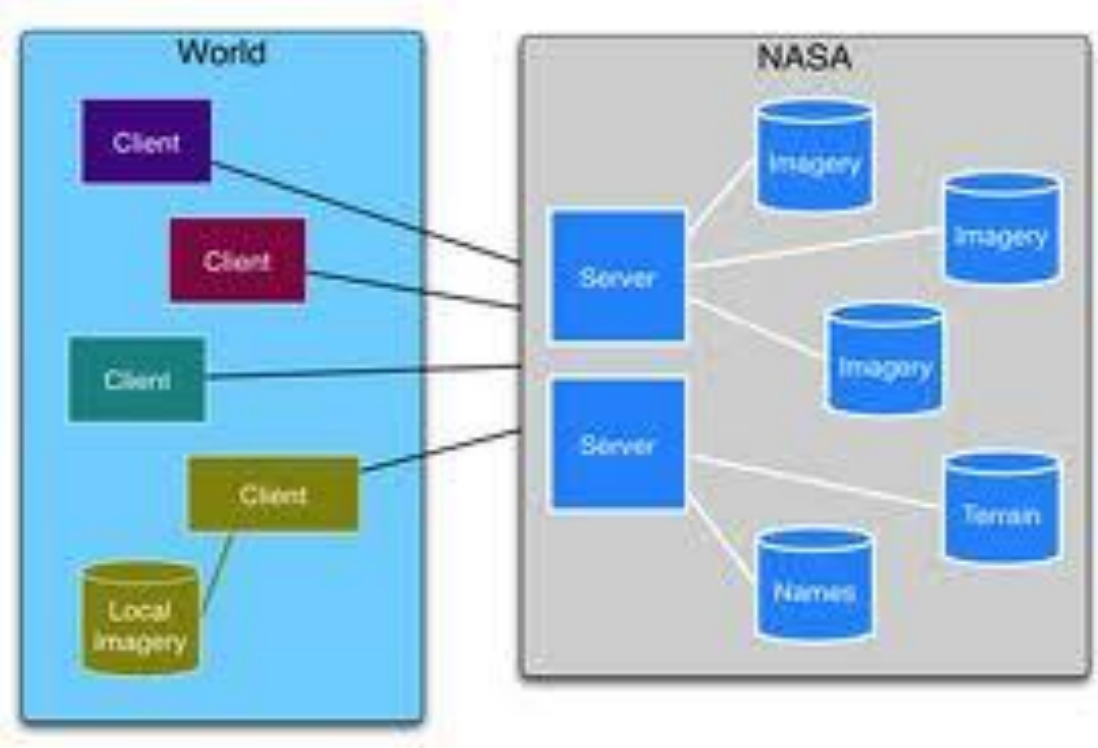

Ilustración 13 – Comunicación cliente-servidor para la obtención de imágenes y otros datos

Como podemos observar en la figura 2.1.1 será el sistema ya desarrollado por NASA y proporcionado en el SDK el que se encargará de gran parte del funcionamiento, nosotros únicamente habremos de comprender el funcionamiento para ser capaces de interactuar con este sistema.

En la figura 2.1.2 lo que se nos muestra es el modo en que la aplicación obtiene los datos, ya sea mediante servidores WMS o de modo local, la segunda forma es mucho más rápida ya que no implica a internet, cosa que nos puede limitar bastante suponiendo que no dispongamos de una conexión lo suficientemente rápida para descargar grandes volúmenes de datos.

El proyecto dispone de una interfaz de usuario bastante clara, al menos ese ha sido el objetivo, en ella observaremos la bola del mundo y distintos botones que abrirán nuevos formularios que tendrán distintas funciones, tales como capturar las coordenadas, medir áreas o distancias, o viajar a distintos puntos del escenario.

Además dispondremos de una interfaz Web, para probar el uso de nuestro programa a modo de Applet, con una página Web de prueba dividida en frames mediante la cuál un usuario podrá acceder al escenario, pinchando en distintos links que serán los encargados de comunicar con el Applet mediante javascript. La Web ha sido desarrollada con el programa Adobe Dreamweaver y escrita en HTML, usando javascript y un poco de php.

2.2 – El paradigma: La orientación al objeto

La orientación a objetos es un paradigma de programación que facilita la creación de software de calidad por sus factores que potencian el mantenimiento, la extensión y la reutilización del software generado bajo este paradigma.

La programación orientada a objetos trata de amoldarse al modo de pensar del hombre y no al de la máquina. Esto es posible gracias a la forma racional con la que se manejan las abstracciones que representan las entidades del dominio del problema, y a propiedades como la jerarquía o el encapsulamiento.

El elemento básico de este paradigma no es la función (elemento básico de la programación estructurada), sino un ente denominado objeto. Un objeto es la representación de un concepto para un programa, y contiene toda la información necesaria para abstraer dicho concepto: los datos que describen su estado y las operaciones que pueden modificar dicho estado, y determinan las capacidades del objeto.

Java incorpora el uso de la orientación a objetos como uno de los pilares básicos de su lenguaje.

Los objetos

Podemos definir objeto como el "encapsulamiento de un conjunto de operaciones (métodos) que pueden ser invocados externamente, y de un estado que recuerda el efecto de los servicios". [Piattini et al., 1996].

Un objeto además de un estado interno, presenta una interfaz para poder interactuar con el exterior. Es por esto por lo que se dice que en la programación orientada a objetos "se unen datos y procesos", y no como en su predecesora, la programación estructurada, en la que estaban separados en forma de variables y funciones.

Un objeto consta de:

- Tiempo de vida: La duración de un objeto en un programa siempre está limitada en el tiempo. La mayoría de los objetos sólo existen durante una parte de la ejecución del programa. Los objetos son creados mediante un mecanismo denominado instanciación, y cuando dejan de existir se dice que son destruidos. En Java esto se encarga de realizarlo el llamado "*Garbage collector*".

- Estado: Todo objeto posee un estado, definido por sus atributos. Con él se definen las propiedades del objeto, y el estado en que se encuentra en un momento determinado de su existencia.

- Comportamiento: Todo objeto ha de presentar una interfaz, definida por sus métodos, para que el resto de objetos que componen los programas puedan interactuar con él.

El equivalente de un objeto en el paradigma estructurado sería una variable. Así mismo la instanciación de objetos equivaldría a la declaración de variables, y el tiempo de vida de un objeto al ámbito de una variable.

Las clases

Las clases son abstracciones que representan a un conjunto de objetos con un comportamiento e interfaz común.

Podemos definir una clase como "un conjunto de cosas (físicas o abstractas) que tienen el mismo comportamiento y características... Es la implementación de un tipo de objeto (considerando los objetos como instancias de las clases)". [Piattini et al., 1996].

Una clase no es más que una plantilla para la creación de objetos. Cuando se crea un objeto (instanciación) se ha de especificar de qué clase es el objeto instanciado, para que el compilador comprenda las características del objeto.

protected Clase <nombre del objeto> = new Clase();

Ejemplo 2.2.1: Instanciación mediante constructor vacío

Las clases presentan el estado de los objetos a los que representan mediante variables denominadas atributos. Cuando se instancia un objeto el compilador crea en la memoria dinámica un espacio para tantas variables como atributos tenga la clase a la que pertenece el objeto.

Los métodos son las funciones mediante las que las clases representan el comportamiento de los objetos. En dichos métodos se modifican los valores de los atributos del objeto, y representan las capacidades del objeto (en muchos textos se les denomina servicios).

Desde el punto de vista de la programación estructurada, una clase se asemejaría a un módulo, los atributos a las variables globales de dicho módulo, y los métodos a las funciones del módulo.

Modelo de objetos

Existen una serie de principios fundamentales para comprender cómo se modeliza la realidad al crear un programa bajo el paradigma de la orientación a

objetos. Estos principios son: la abstracción, el encapsulamiento, la modularidad, la jerarquía, el paso de mensajes y el poliforfismo.

a.) Principio de Abstracción

Mediante la abstracción la mente humana modeliza la realidad en forma de objetos. Para ello busca parecidos entre la realidad y la posible implementación de objetos del programa que simulen el funcionamiento de los objetos reales.

Los seres humanos no pensamos en las cosas como un conjunto de cosas menores; por ejemplo, no vemos un cuerpo humano como un conjunto de células. Los humanos entendemos la realidad como objetos con comportamientos bien definidos. No necesitamos conocer los detalles de porqué ni cómo funcionan las cosas; simplemente solicitamos determinadas acciones en espera de una respuesta; cuando una persona desea desplazarse, su cuerpo le responde comenzando a caminar.

Pero la abstracción humana se gestiona de una manera jerárquica, dividiendo sucesivamente sistemas complejos en conjuntos de subsistemas, para así entender más fácilmente la realidad. Esta es la forma de pensar que la orientación a objeto intenta cubrir.

b.) Principio de Encapsulamiento

El encapsulamiento permite a los objetos elegir qué información es publicada y qué información es ocultada al resto de los objetos. Para ello los objetos suelen presentar sus métodos como interfaces públicas y sus atributos como datos privados e inaccesibles desde otros objetos.

Para permitir que otros objetos consulten o modifiquen los atributos de los objetos, las clases suelen presentar métodos de acceso. De esta manera el

acceso a los datos de los objetos es controlado por el programador, evitando efectos laterales no deseados.

Con el encapsulado de los datos se consigue que las personas que utilicen un objeto sólo tengan que comprender su interfaz, olvidándose de cómo está implementada, y en definitiva, reduciendo la complejidad de utilización.

c.) Principio de Modularidad

Mediante la modularidad, se propone al programador dividir su aplicación en varios módulos diferentes (ya sea en forma de clases, paquetes o bibliotecas), cada uno de ellos con un sentido propio.

Esta fragmentación disminuye el grado de dificultad del problema al que da respuesta el programa, pues se afronta el problema como un conjunto de problemas de menor dificultad, además de facilitar la comprensión del programa.

d.) Principio de Jerarquía

La mayoría de nosotros ve de manera natural nuestro mundo como objetos que se relacionan entre sí de una manera jerárquica. Por ejemplo, un perro es un mamífero, y los mamíferos son animales, y los animales seres vivos...

Del mismo modo, las distintas clases de un programa se organizan mediante la jerarquía. La representación de dicha organización da lugar a los denominados árboles de herencia:

Mediante la herencia una clase hija puede tomar determinadas propiedades de una clase padre. Así se simplifican los diseños y se evita la duplicación de código al no tener que volver a codificar métodos ya implementados.

Al acto de tomar propiedades de una clase padre se denomina heredar.

e.) Principio del Paso de Mensajes

Mediante el denominado paso de mensajes, un objeto puede solicitar de otro objeto que realice una acción determinada o que modifique su estado. El paso de mensajes se suele implementar como llamadas a los métodos de otros objetos.

Desde el punto de vista de la programación estructurada, esto correspondería con la llamada a funciones.

f.) Principio de Polimorfismo

Polimorfismo quiere decir "un objeto y muchas formas". Esta propiedad permite que un objeto presente diferentes comportamientos en función del contexto en que se encuentre. Por ejemplo un método puede presentar diferentes implementaciones en función de los argumentos que recibe, recibir diferentes números de parámetros para realizar una misma operación, y realizar diferentes acciones dependiendo del nivel de abstracción en que sea llamado.

Relaciones entre objetos

Durante la ejecución de un programa, los diversos objetos que lo componen han de interactuar entre sí para lograr una serie de objetivos comunes.

Existen varios tipos de relaciones que pueden unir a los diferentes objetos, pero entre ellas destacan las relaciones de: asociación, todo/parte, y generalización/especialización.

a.) Relaciones de Asociación

Serían relaciones generales, en las que un objeto realiza llamadas a los servicios (métodos) de otro, interactuando de esta forma con él.

Representan las relaciones con menos riqueza semántica.

b.) Relaciones de Todo/Parte

Muchas veces una determinada entidad existe como conjunción de otras entidades, como un conglomerado de ellas. La orientación al objeto recoge este tipo de relaciones como dos conceptos; la agregación y la composición.

En este tipo de relaciones un objeto componente se integra en un objeto compuesto. La diferencia entre agregación y composición es que mientras que la composición se entiende que dura durante toda la vida del objeto componedor, en la agregación no tiene por qué ser así.

Esto se puede implementar como un objeto (objeto compuesto) que cuenta entre sus atributos con otro objeto distinto (objeto componente).

c.) Relaciones de Generalización/Especialización

A veces sucede que dos clases tiene muchas de sus partes en común, lo que normalmente se abstrae en la creación de una tercera clase (padre de las dos) que reúne todas sus características comunes.

El ejemplo más extendido de este tipo de relaciones es la herencia, propiedad por la que una clase (clase hija) recoge aquellos métodos y atributos que una segunda clase (clase padre) ha especificado como "heredables".

Este tipo de relaciones es característico de la programación orientada a objetos.

En realidad, la generalización y la especialización son diferentes perspectivas del mismo concepto, la generalización es una perspectiva ascendente (bottom-up), mientras que la especialización es una perspectiva descendente (top-down).

Para más información sobre el modelo de objetos en la programación avanzada, y las relaciones entre objetos véase **[García, 1998]** o para una información más detallada consulte **[Booch, 1996].**

2.3 – GLOBAL MAPPER

Global mapper es un programa muy utilizado en el mundo de la información geográfica, es desarrollado por la empresa *Global Mapper Software, LLC.* Esta aplicación permite el tratamiento de multitud de formatos de imagen entre los que se encuentran

- DEM: Modelos de digitales de elevación.
- DTED: Modelos de digitales del terreno.
- GPX: Esquema XML pensado para transferir datos GPS entre aplicaciones.
- SDTS DEM: Representación digital de información cartográfica en formato raster o de imagen. Creado por la USGS(*United States Geological Survey)*.
- ESRI Shapefiles: Formato estándar para la representación de shapes.
- JPEG2000: formato de imagen.
- KML/KMZ: es un lenguaje de marcado basado en XML para representar datos geográficos en tres dimensiones, es utilizado principalmente en Google Earth.
- Lidar LAS: Archivos que representan los datos obtenidos mediante tecnología LiDAR.

El principal uso que hemos hecho de este programa en este proyecto ha sido el conseguir la estructura de niveles necesaria para representar una imagen en

worldwind de forma eficiente, del mismo modo que se hace con los servidores WMS. Para ello hemos obtenido una ortofoto de un lugar, del cual deseamos realizar el escenario virtual y por tanto deseamos mayor resolución de la que nos ofrece un servidor WMS convencional, y mediante global mapper obtendremos la estructura de niveles partiendo la imagen en distintos *tiles* y directorios de menor a mayor resolución.

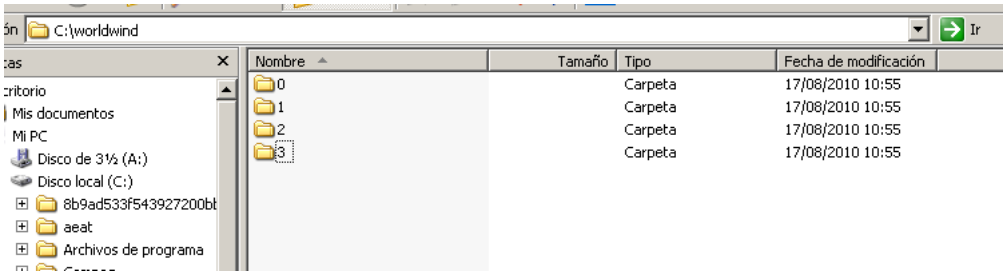

Ilustración 14 – Estructura de niveles

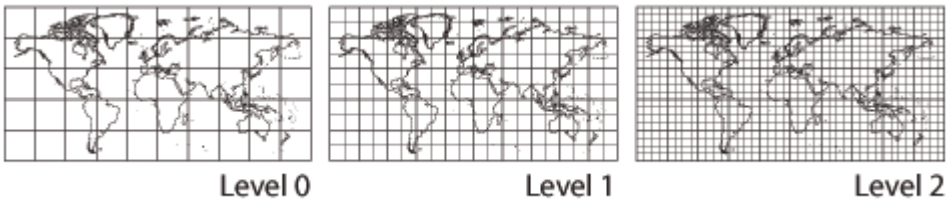

Ilustración 15 – Ilustración de niveles, a mayor nivel, mayor resolución.

Para ello debemos abrir la ortofoto referenciada, esto es habiéndole aplicado las coordenadas donde corresponde en el globo terráqueo, con el susodicho programa y seguir los pasos indicados en el documento *WWJ_Exporting_from_Global_mapper.pdf,* el cual podemos encontrar fácilmente en el foro de soporte de worldwind JAVA. A grandes rasgos hemos de exportar la imagen indicando el número de niveles deseado, la resolución, la cuál deberá ser acorde con la máxima resolución de la ortofoto, y la proyección y datum de la imagen(los más comunes son coordenadas geográficas y datum WGS84). En caso de no seguir alguno de estos pasos podemos obtener los directorios igualmente, pero quizás al intentar visualizarla en worldwind no aparezca correctamente referenciada o con una proyección extraña, esto es que la imagen pudiera verse excesivamente achatada o alargada.

2.3 – gvSIG

Un poco de historia

El origen de gvSIG se remonta al año 2004, en el seno del proyecto de migración a software libre de los sistemas informáticos de la Conselleria de Infraestructuras y Transporte (CIT). Inicialmente nace con unos objetivos acordes a las necesidades de la CIT. Estos objetivos se ven rápidamente ampliados, fruto por un lado de la naturaleza del software libre -que facilita enormemente la expansión de la tecnología, del conocimiento y establece las bases sobre la que constituir una comunidad- y por otro de una visión de proyecto materializada en unas líneas de demarcación y un plan acorde para llevarlas a cabo.

Visión y Misión

La visión del proyecto gvSIG nace como respuesta a una serie de preguntas que nos planteábamos al iniciar el proyecto:

- ¿Cómo Interpretamos el modelo del Software Libre?
- ¿Qué hacer para que sea un proyecto duradero, sostenible en el tiempo?
- ¿Cómo construir una Comunidad sólida?

Las respuestas a estas preguntas hay que ubicarlas en la situación que vivía la geomática libre en el año 2004. Iniciábamos un proyecto de Software Libre en unos años donde las soluciones en el campo de la geomática libre no estaban tan maduras como están ahora y donde existían, y siguen existiendo, diversas maneras de interpretar el modelo del Software Libre. Un entorno donde estábamos muy acostumbrados a ver proyectos de Software Libre que por determinadas causas casi terminaban antes de empezar o se mantenían con un impacto residual sin provocar cambios significativos en el sistema predominante. Un entorno donde los únicos actores relevantes eran un puñado de multinacionales de software privativo.

De manera estructural, consideramos que podemos clasificar en dos grandes categorías la forma de avanzar en el desarrollo del conocimiento:

- Poniendo a diversos grupos rivalizando alrededor de un problema o temática específica, utilizando el conocimiento adquirido como argumento principal y especulando con el mismo
- Convirtiendo el conocimiento adquirido en conocimiento compartido, de forma que se puedan sumar cuantos más grupos mejor a la solución del problema o el desarrollo de la temática en cuestión.

Cambiar el modelo dominante en la actualidad, basado en la especulación del conocimiento adquirido para progresar individualmente por un modelo basado en el conocimiento compartido y la colaboración para progresar de manera conjunta, trabajar desde el mundo de la geomática por un modelo mejor y más justo se constituye en la principal misión del proyecto gvSIG.[www.gvsig.org]

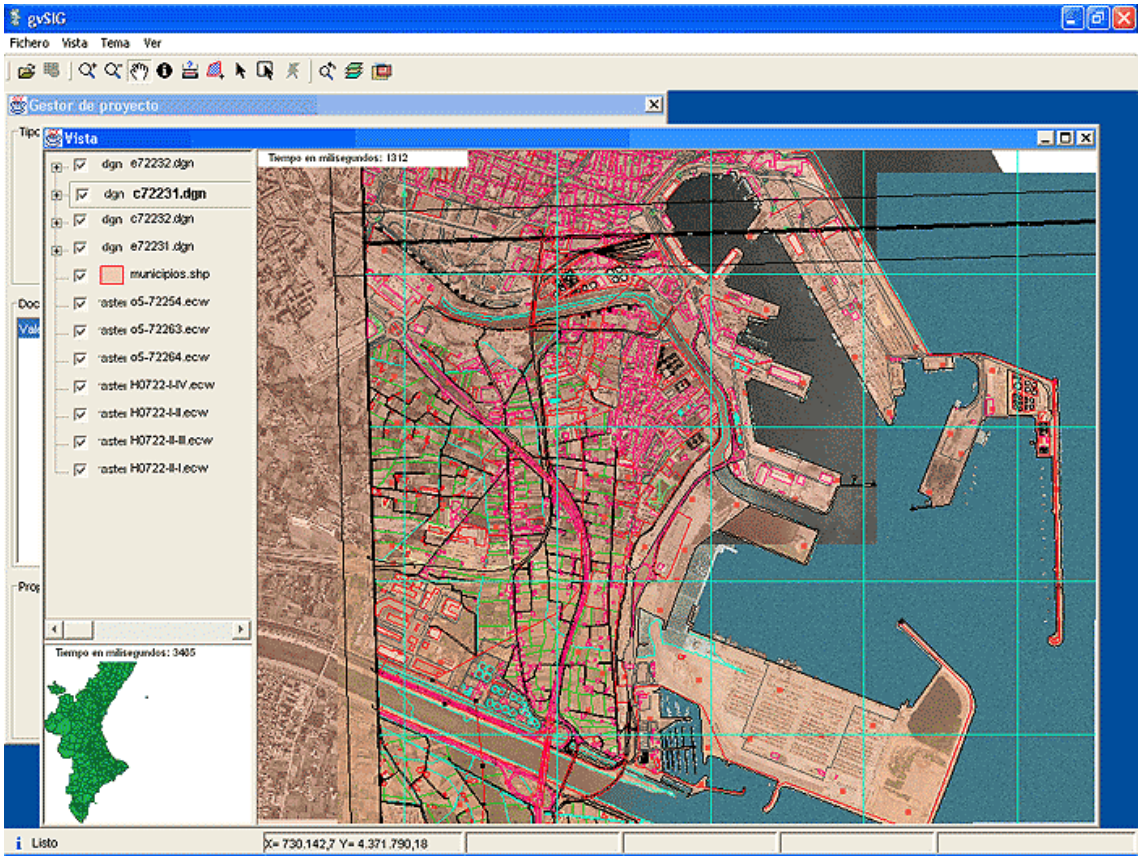

Ilustración 16 – gvSig, detalle de unas capas sobre el puerto de Valencia

Uso en el proyecto

Esta pequeña introducción extraída íntegramente de la Web de gvSIG es más que suficiente para ilustrar el programa, el uso que hemos hecho de este programa ha sido variado, desde testear servidores WMS para comprobar su correcto funcionamiento y comprobar que podíamos usarlos con worldwind, edición de algunas capas –trabajo realizado por otros empleados de la empresa DIELMO- y exportación de las bases de datos contenidos en los ficheros de tipo *"shape de puntos"*, generalmente estos ficheros de puntos indican lugares o topónimos, al exportarlo a una base de datos en algún formato conocido que pueda ser accedido fácilmente mediante JAVA podremos añadir una base de datos de topónimos de forma muy sencilla a nuestro escenario, ya que esta base de datos contendrá un identificador(generalmente el nombre del POI, info o topónimo) y unas coordenadas que marcarán la ubicación de este en objeto en el escenario, es importante reseñar que las coordenadas deberán estar en formato geográficas, ya que es el formato que utiliza worldwind, obviamente existen otros formatos como por ejemplo UTM.

2.3 – ADOBE DREAMWEAVER

Introducción a Dreamweaver

Dreamweaver es la herramienta de diseño de páginas Web más avanzada, tal como se ha afirmado en muchos medios.

Cumple perfectamente el objetivo de diseñar páginas con aspecto profesional, y soporta gran cantidad de tecnologías, además muy fáciles de usar:

- Hojas de estilo y capas
- Javascript para crear efectos e interactividades
- Inserción de archivos multimedia...
- PHP

En lo fundamental de las herramientas HTML WYSIWYG, también permite la conexión a Bases de Datos como MySQL y Microsoft Access, para filtrar y mostrar el contenido utilizando tecnología de script como, por ejemplo, ASP (Active Server Pages), ASP.NET, ColdFusion, JSP (JavaServer Pages) y PHP sin necesidad de tener experiencia previa en programación.

Uso en el proyecto

En lo que atañe al uso de esta aplicación en nuestro proyecto, ha sido de gran utilidad a la hora de implementar las páginas Web de prueba para nuestros Applets, con este programa hemos podido realizar de un modo sencillo y rápido Web que utilicen la tecnología Javascript, para la comunicación con el Applet, o php para acceso a bases de datos.

Las facilidades que nos ofrece esta aplicación para la creación de Webs son inmensas, hasta el punto de no ser necesario tener unos amplios conocimientos de programación Web para hacer páginas con un nivel mas que aceptable.

2.3 – EL LENGUAJE DE PROGRAMACIÓN JAVA

Java es un lenguaje de programación bajo el paradigma orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

El lenguaje Java, historia

La tecnología Java se creó como una herramienta de programación para ser usada en un proyecto de set-top-box en una pequeña operación denominada the Green Project en Sun Microsystems en el año 1991. El equipo (Green Team), compuesto por trece personas y dirigido por James Gosling, trabajó durante 18 meses en Sand Hill Road en Menlo Park en su desarrollo.

El lenguaje se denominó inicialmente Oak (por un roble que había fuera de la oficina de Gosling), luego pasó a denominarse Green tras descubrir que Oak era ya una marca comercial registrada para adaptadores de tarjetas gráficas y finalmente se renombró a Java.

El término Java fue acuñado en una cafetería frecuentada por algunos de los miembros del equipo. Pero no está claro si es un acrónimo o no, aunque algunas fuentes señalan que podría tratarse de las iniciales de sus creadores: James Gosling, Arthur Van Hoff, y Andy Bechtolsheim. Otros abogan por el siguiente acrónimo, Just Another Vague Acronym ("sólo otro acrónimo ambiguo más"). La hipótesis que más fuerza tiene es la que Java debe su nombre a un tipo de café disponible en la cafetería cercana, de ahí que el icono de Java sea una taza de café caliente. Un pequeño signo que da fuerza a esta teoría es que los 4 primeros bytes (el número mágico) de los archivos .class que genera el compilador, son en hexadecimal, 0xCAFEBABE. Otros simplemente dicen que el nombre fue sacado al parecer de una lista aleatoria de palabras.

Los objetivos de Gosling eran implementar una máquina virtual y un lenguaje con una estructura y sintaxis similar a C++. Entre junio y julio de 1994, tras una sesión maratoniana de tres días entre John Gaga, James Gosling, Joy Naughton, Wayne Rosing y Eric Schmidt, el equipo reorientó la plataforma hacia la Web. Sintieron que la llegada del navegador web Mosaic, propiciaría que Internet se convirtiese en un medio interactivo, como el que pensaban era la televisión por cable. Naughton creó entonces un prototipo de navegador, WebRunner, que más tarde sería conocido como HotJava.

En 1994, se les hizo una demostración de HotJava y la plataforma Java a los ejecutivos de Sun. Java 1.0a pudo descargarse por primera vez en 1994, pero hubo que esperar al 23 de mayo de 1995, durante las conferencias de SunWorld, a que vieran la luz pública Java y HotJava, el navegador Web. El acontecimiento fue anunciado por John Gage, el Director Científico de Sun Microsystems. El acto estuvo acompañado por una pequeña sorpresa adicional, el anuncio por parte de

Marc Andreessen, Vicepresidente Ejecutivo de Netscape, de que Java sería soportado en sus navegadores. El 9 de enero del año siguiente, 1996, Sun fundó el grupo empresarial JavaSoft para que se encargase del desarrollo tecnológico. [1] Dos semanas más tarde la primera versión de Java fue publicada.

La promesa inicial de Gosling era Write Once, Run Anywhere (Escríbelo una vez, ejecútalo en cualquier lugar), proporcionando un lenguaje independiente de la plataforma y un entorno de ejecución (la JVM) ligero y gratuito para las plataformas más populares de forma que los binarios (bytecode) de las aplicaciones Java pudiesen ejecutarse en cualquier plataforma.

El entorno de ejecución era relativamente seguro y los principales navegadores web pronto incorporaron la posibilidad de ejecutar applets Java incrustadas en las páginas web.

¿Porqué hemos usado Java y no .NET?

El porqué de esta pregunta se basa principalmente en dos motivos de peso, el primero y más importante es que la versión .NET ha sido abandonada por la propia NASA, quedando obsoleta, aunque todavía funcional al 100%. Y el segundo motivo es la portabilidad que nos permite JAVA con independencia del sistema operativo, ya que con el mismo código podremos ejecutar nuestro programa en Windows, Linux, Solaris… Esto es una ventaja muy grande sobre todo de cara al futuro de nuestra aplicación, cuanto mayor sea el abanico de posibles usuarios, más probabilidades de éxito de nuestro producto.

Además la versión Java tiene una activa comunidad a través del foro mantenido por la propia NASA, en el que cientos de usuarios aportan nuevas soluciones, nuevas clases y nueva información o aplicaciones para worldwind.

2.6.1 – LOS APPLETS EN JAVA

Desde la primera versión de Java existe la posibilidad de desarrollar pequeñas aplicaciones (Applets) en Java que luego pueden ser incrustadas en una página HTML para que sean descargadas y ejecutadas por el navegador web. Estas mini-aplicaciones se ejecutan en una JVM que el navegador tiene configurada como extensión (plug-in) en un contexto de seguridad restringido configurable para impedir la ejecución local de código potencialmente malicioso.

El éxito de este tipo de aplicaciones (la visión del equipo de Gosling) no fue realmente el esperado debido a diversos factores, siendo quizás el más importante la lentitud y el reducido ancho de banda de las comunicaciones en aquel entonces que limitaba el tamaño de las applets que se incrustaban en el navegador. La aparición posterior de otras alternativas (aplicaciones web dinámicas de servidor) dejó un reducido ámbito de uso para esta tecnología, quedando hoy relegada fundamentalmente a componentes específicos para la intermediación desde una aplicación web dinámica de servidor con dispositivos ubicados en la máquina cliente donde se ejecuta el navegador.

Entre sus características podemos mencionar un esquema de seguridad que permite que los applets que se ejecutan en el equipo no tengan acceso a partes sensibles (por ej. no pueden escribir archivos), a menos que uno mismo le dé los permisos necesarios en el sistema; la desventaja de este enfoque es que la entrega de permisos es engorrosa para el usuario común, lo cual juega en contra de uno de los objetivos de los Java applets: proporcionar una forma fácil de ejecutar aplicaciones desde el navegador web.

En Java un applet (Subprograma), es un programa que puede incrustarse en un documento HTML; es decir en una página Web. Cuando un Navegador carga una página Web que contiene un Applet, éste se descarga en el navegador Web y comienza a ejecutarse. Esto nos permite crear programas que cualquier usuario puede ejecutar con tan solo cargar la página Web en su navegador.

El Navegador que carga y ejecuta el applet se conoce en términos genéricos como el contenedor de Applets. El kit de desarrollo de Software para Java 2 (J2SDK) 1.4.1 incluye el contenedor de Applets, llamado appletviewer, para probar los applets antes de incrustarlos en una página Web.

Uso en el proyecto

Esta tecnología nos es muy útil a nuestra causa, ya que permite una mayor difusión de nuestra aplicación, como ejemplo podemos poner un ayuntamiento que desee realizar un escenario virtual para sus ciudadanos, la forma más rápida de difusión hoy en día es sin lugar a dudas la web, de modo que todos los ciudadanos tendrían acceso en tiempo real a la aplicación sin necesidad de desplazarse a comprar el software, ni realizar instalaciones.

También cabe destacar que la traducción de un programa de escritorio escrito en Java, a un applet es una tarea bastante sencilla, al menos en nuestro caso, lo que nos da de nuevo una gran ventaja.

2.7 – HTML

HTML es el acrónimo de "Hiper Text Markup Languaje", es decir, Lenguaje de Marcas de Hipertexto. Es el utilizado para escribir páginas Web. Se compone exclusivamente de texto, y entre el texto que debe aparecer en pantalla se especifican con una sintaxis especial cuestiones referentes a la presentación de ese texto (tamaño del texto, alineación...).

Una de las mayores ventajas de HTML es que con un bloque de texto o palabra se consigue crear un enlace a otro documento diferente.

Es un lenguaje estandarizado, para que todos los clientes Web (navegadores) puedan entenderlo e interpretando presentando al usuario una salida equivalente.

Así se consigue que en la pantalla aparezcan animaciones, imágenes, distintos tipos de letras y fuentes, alineaciones, tablas y sobre todo los enlaces a otras páginas.

Existen una serie de principios de diseño con los que ha sido construido HTML:

- Interoperatividad: Una página Web ha de dar los recursos necesarios a sus desarrolladores para que no se salgan del estándar y caigan en una falta de portabilidad.
- Internacionalización: Los documentos pueden ser escritos en los diferentes lenguajes y códigos de letra que existen en el mundo (alfabetos orientales...).
- Accesibilidad: En las páginas debe posibilitarse alternativas a los objetos para gente con discapacidades o limitaciones físicas y agentes que no tengan la habilidad de interpretarlos.
- Objetos: Debe presentar objetos de muy diversos tipos (applets, imágenes, vídeo, sonido...).
- Interactividad: Debe permitir al diseñador presentar interactividad entre la página y el usuario que la lee. Esto se consigue mediante formularios y scripts.
- Facilidad: En el aprendizaje de lo básico y en la utilización. HTML es incremental, y así diseñar una página sencilla es una tarea fácil, esta tarea se va complicando a medida que buscamos estructuras más complejas.

Uso en el proyecto: Se ha utilizado principalmente con el programa Dreamweaver y de forma muy básica, para la creación de Webs de prueba en las que se han insertado applets.

2.8 – JAVASCRIPT

JavaScript es un lenguaje de scripting basado en objetos no tipeado y liviano, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. JavaScript es un dialecto de ECMAScript y se

caracteriza por ser un lenguaje basado en prototipos, con entrada dinámica y con funciones de primera clase. JavaScript ha tenido influencia de múltiples lenguajes y se diseñó con una sintaxis similar al lenguaje de programación Java, aunque más fácil de utilizar para personas que no programan.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM.

El lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, la que desarrolló los primeros navegadores web comerciales. Apareció por primera vez en el producto de Netscape llamado Netscape Navigator 2.0.

Uso en el proyecto: Javascript ha sido utilizado para acceder, en las páginas web, a los distintos métodos públicos de nuestro applet, como pueden ser la búsqueda de algún topónimo o viajar hasta el.

2.9 – PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

En nuestro proyecto el uso de PHP se ha limitado a alguna función en alguna web de prueba para acceder a bases de datos y realizar alguna búsqueda dinámica.

2.10 - LOS SERVIDORES WMS Y WFS

WMS

El servicio Web Map Service (WMS) definido por el OGC (Open Geospatial Consortium) produce mapas de datos referenciados espacialmente, de forma dinámica a partir de información geográfica. Este estándar internacional define un "mapa" como una representación de la información geográfica en forma de un archivo de imagen digital conveniente para la exhibición en una pantalla de ordenador. Un mapa no consiste en los propios datos. Los mapas producidos por WMS se generan normalmente en un formato de imagen como PNG, GIF o JPEG, y opcionalmente como gráficos vectoriales en formato SVG (Scalable Vector Graphics) o WebCGM (Web Computer Graphics Metafile).

El estándar define tres operaciones:

- Devolver metadatos del nivel de servicio.
- Devolver un mapa cuyos parámetros geográficos y dimensionales han sido bien definidos.
- Devolver información de características particulares mostradas en el mapa (opcionales).

Las operaciones WMS pueden ser invocadas usando un navegador estándar realizando peticiones en la forma de URLs (Uniform Resource Locators). El contenido de tales URLs depende de la operación solicitada. Concretamente, al solicitar un mapa, la URL indica qué información debe ser mostrada en el mapa, qué porción de la tierra debe dibujar, el sistema de coordenadas de referencia, y la

anchura y la altura de la imagen de salida. Cuando dos o más mapas se producen con los mismos parámetros geográficos y tamaño de salida, los resultados se pueden solapar para producir un mapa compuesto. El uso de formatos de imagen que soportan fondos transparentes (e.g., GIF o PNG) permite que los mapas subyacentes sean visibles. Además, se puede solicitar mapas individuales de diversos servidores.

El servicio WMS permite así la creación de una red de servidores distribuidos de mapas, a partir de los cuales los clientes pueden construir mapas a medida. Las operaciones WMS también pueden ser invocadas usando clientes avanzados SIG, realizando igualmente peticiones en la forma de URLs.

WFS

Web Feature Service o WFS del Open Geospatial Consortium u OGC es un servicio estándar, que ofrece una interfaz de comunicación que permite interactuar con los mapas servidos por el estándar WMS, como por ejemplo, editar la imagen que nos ofrece el servicio WMS o analizar la imagen siguiendo criterios geográficos.

Para realizar estas operaciones se utiliza el lenguaje GML que deriva del XML, que es el estándar a través del que se transmiten la ordenes WFS.

WFS no transaccional permite hacer consultas y recuperación de elementos geográficos. Por contra WFS-T (Web Feature Service Transactional) permite además la creación, eliminación y actualización de estos elementos geográficos del mapa.

A pesar de nuestro deseo no se ha podido implementar WFS en worldwind, debido a la falta de experiencia en este campo. Únicamente utiliza WFS la aplicación en un par de capas que ya habían sido definidas en el SDK, una de ellas es la encargada de mostrar los nombres de los países y ciudades, y otra es la encargada de mostrar las fronteras.

Autor: José Vicente Mora Espí
Escuela técnica superior de ingeniería informática

III – Núcleo del trabajo

3.1 – LA SOLUCIÓN

Tras meses de duro trabajo se ha llegado a una solución que cumple sobradamente los requisitos de la aplicación: WorldWind Escenarios Virtuales 1.0

El proyecto se divide en varia partes bien diferenciadas:

DOCUMENTACIÓN

La primera parte del proyecto se basa en documentación, ya que fue una de las partes más largas y complejas de realizar, entender cómo funcionaba el programa, como debían ser usadas las distintas clases, qué estructura seguía worldwind, afianzar conceptos, etcétera.

Existen además una serie de documentos a entregar al cliente:

- Manual de usuario: En este documento, destinado a todos los usuarios del programa se explica cómo instalar y utilizar el programa. Dicho documento se puede encontrar en el apartado 6.8 Apéndices - Manual de usuario.

- Manual de programador: En este documento, se describe cada una de las clases que componen el proyecto, con comentarios internos extraídos de los fuentes y la definición de cada una extraída de los distintos ficheros de cabecera. Dicho documento se puede encontrar en el apartado 6.8 Apéndices - Manual de programador.

- Estructura del CD-ROM: En este documento se introduce la jerarquía de directorios para todos los usuarios que consigan el CD-ROM en el que se presenta este proyecto. Dicho documento se puede encontrar en el apartado 6.10 Apéndices - Estructura del CD-ROM.

Documentación adicional en la memoria del proyecto:

- Bibliografía: El apartado V de esta memoria

- Glosario: El apartado Glosario de esta memoria. Se ha intentado que fuera suficientemente rico como para que con su ayuda toda la memoria fuese entendible por cualquier persona, sea o no ingeniero.

- Desarrollo del proyecto: Se ha entregado a su vez una pequeño cuaderno de bitácora del desarrollo del proyecto a lo largo del tiempo, para que esta experiencia sirva para futuros desarrolladores, especialmente alumnos que estén realizando un Proyecto Fin de Carrera de características similares a éste. Dicho documento se puede encontrar en el apartado 6.10 Apéndices - Estructura del CD-ROM.

CD-ROM

El proyecto se distribuye en un CD-ROM, que contiene no solo la documentación generada, el ejecutable y los códigos fuente del proyecto, sino además una serie de documentación bibliográfica obtenida de Internet y otros sitios, así como herramientas relacionadas y utilizadas durante el desarrollo del proyecto.

LAS IGU

Dado que este es un programa enfocado a mostrar a un usuario final un escenario en 3D sobre un mundo virtual, se ha querido cuidar bien el diseño de las interfaces gráficas de usuario haciéndolas claras y accesibles a todo tipo de usuarios, todas ellas están basadas en los paquetes de Java AWT y sus derivadas.

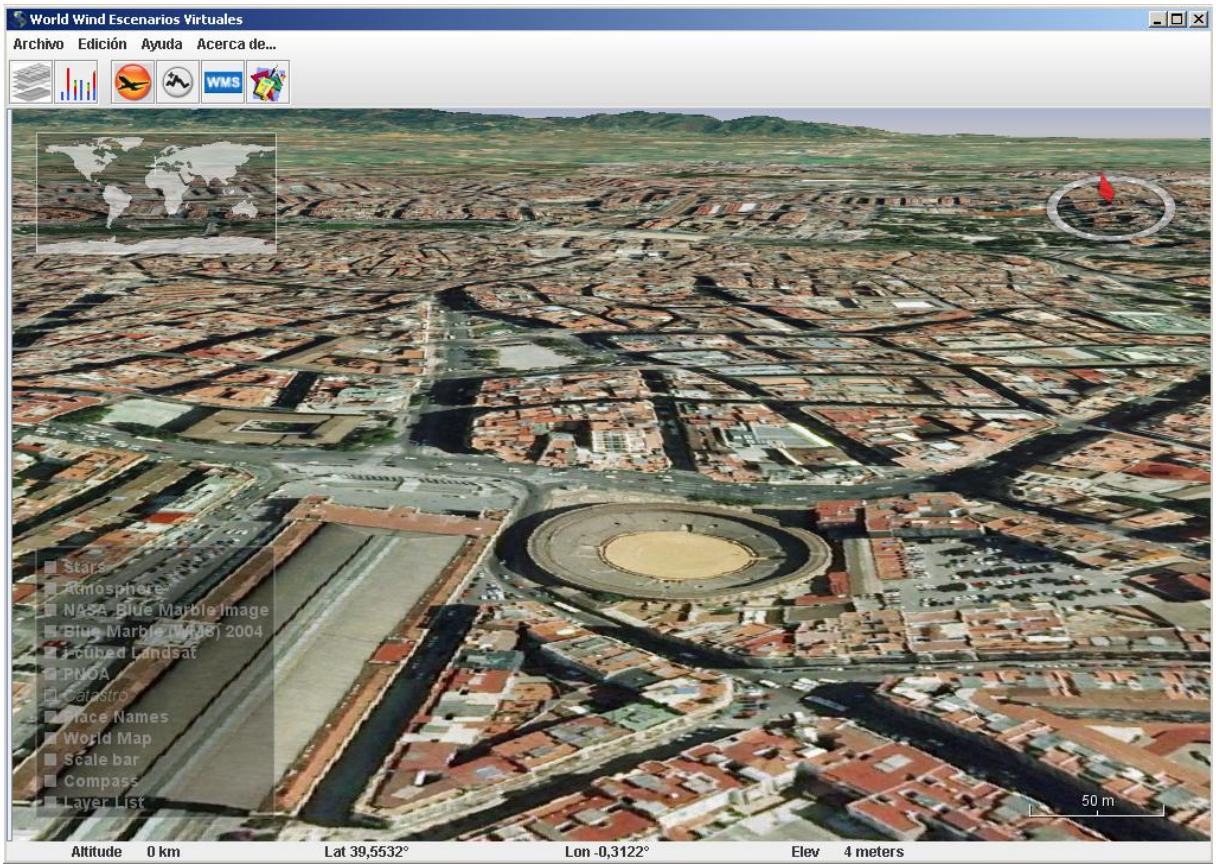

Ilustración 17 – Aspecto de la ventana principal de nuestra aplicación en una vista aérea sobre valencia

3.2 – EL ENFOQUE

Se ha tratado de dar un enfoque práctico a la aplicación, resultado de una metodología de programación extrema formulado por Kent Beck en su libro *Extreme Programming Explained: Embrace Change (1999)*, pues se han ido aplicando soluciones en función de las necesidades. Puesto que gran parte de la aplicación está ya "preparada" en el SDK NASA Worldwind Java mediante un gran número de interfaces, simplemente hemos ido implementando aquellas que nos

eran necesarias paso a paso. Puesto que el inicio del proyecto se desarrolló en una empresa y esta no tenía una planificación clara, las soluciones se iban aportando poco a poco.

Para ello se han intentado aplicar lo mejor posible los paradigmas que definen esta metodología:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.

- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.

- Frecuente integración del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.

- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.

- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.

- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario.

La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.**[Extreme Programming Explained, 1999]**

3.3 – EL DESARROLLO

El proyecto comienza durante el curso 09-10, cuando se inician en Febrero de 2010 unas prácticas en la empresa DIELMO 3D que busca adaptar este software a sus necesidades profesionales, para crear un programa, basado en una licencia gratuita, capaz de sustituir a otros programas de pago. Una vez allí se comenta con el gerente cuales son sus necesidades y objetivos para tratar de clarificar el proyecto.

Una vez que se determinó todo esto se comenzó un proceso de documentación para comprender bien cómo funcionaba la aplicación worldwind, mediante la lectura de infinidad de foros, contactos con gente que hubiese estado trabajando anteriormente con el SDK y realizando multitud de ejemplos de prueba, comprobando nuevas funcionalidades y limitaciones del programa.

El desarrollo del proyecto "a tiempo completo" por parte del alumno comienza en junio de 2010, al término de las prácticas en la empresa, el alumno decide presentar el proyecto a su profesor y determinar que ese sería el proyecto a realizar. Se hizo una planificación usando como métrica el tiempo, dejando un razonable 25% del tiempo disponible periódicamente para recuperar el tiempo perdido por imprevistos y demás retrasos, y basándose en la experiencia adquirida en la realización de diversas prácticas a lo largo de la carrera.

En julio 2010 se presenta ya una aplicación de prueba que cumple perfectamente con los requisitos más básicos, la cual es aprobada por el tutor de prácticas.

A mediados de Julio 2010 se empieza a investigar sobre la posibilidad de embeber la aplicación en web, se empieza a estudiar HTML básico, así como fundamentos de JavaScript y PHP.

En agosto de 2010 con una aplicación final ya presentada al tutor se decide pulir los detalles existentes en cuanto a la codificación y presentación del mismo, se crean escenarios de prueba para ser introducidos en web como applets y ser mostrados en la posterior exposición del proyecto.

A mediados de agosto toda la fase de software está terminada y se incide en mejorar la parte escrita del proyecto, esto es documentación, memoria, bibliografía, consultas y todo aquello que tenga valor documental.

A finales de agosto de 2010 se entrega la memoria y la aplicación final al tutor para que dé sus indicaciones sobre posibles mejoras en cualquier aspecto que considere oportuno.

A principios de septiembre se tiene como objetivo entregar el proyecto en su estado final y preparar la defensa del mismo para su evaluación por parte del tutor.

3.4 – CASOS DE USO

Para el desarrollo de un proyecto es muy importante realizar un modelado tanto de análisis y diseño para obtener un conjunto de modelos que nos permitan especificar los requisitos, la estructura y el comportamiento del sistema. Comenzaremos nuestro análisis del sistema realizando el modelado de casos de uso. Entendemos un caso de uso como una descripción que especifica un comportamiento deseado del sistema. Los casos de uso describen qué hace el sistema representando los requisitos funcionales.

Para describir los casos de uso de WorldWind Escenarios Virtuales, nos ayudaremos de representaciones gráficas, diagramas de casos de uso, y de una descripción del mismo.

Esta descripción incluirá el nombre del caso de uso, actores que intervienen en él, precondiciones, postcondiciones, escenario principal y lo posibles escenarios alternativos. Entendemos por actores del sistema aquellos usuarios, sistemas o tiempo que intervienen en la aplicación. Entendemos por escenario principal, el flujo normal que seguirá la aplicación para ese caso de uso.

3.4.1 Caso de uso: El usuario añade un fichero MDT en modo local

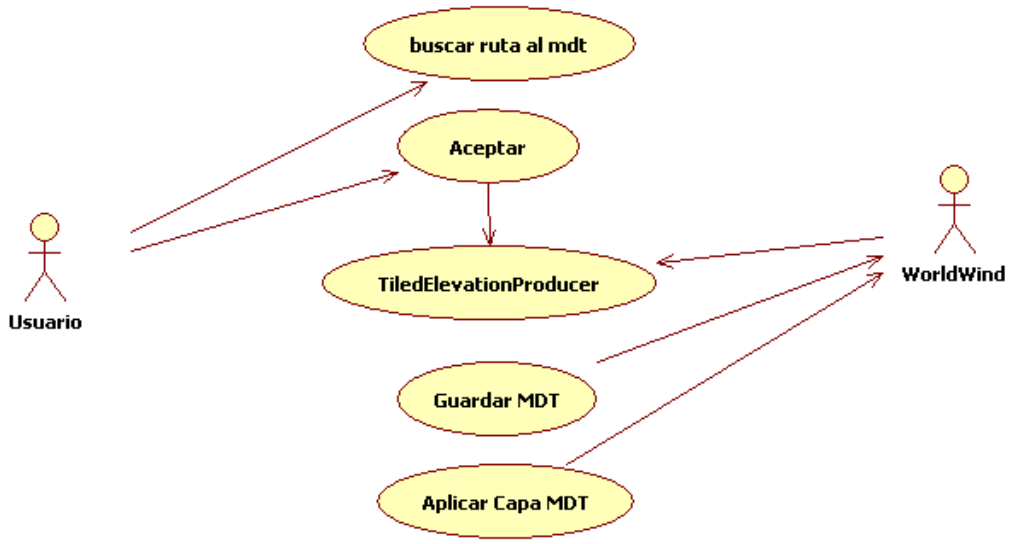

Ilustración 18 – Caso de uso insertar MDT

Caso de Uso: Añadir MDT.

Precondiciones: El fichero en formato *.bil debe existir y estar georeferenciado. El tamaño debe estar entre 1 y 800MB aproximadamente.

Postcondiciones: El usuario añadirá un MDT al modelo digital 3D en modo local.

Descripción: Añadir un MDT

Resumen: El usuario quiere añadir un MDT propio de una zona concreta.

Actores: Usuario(Principal)

Escenario Principal

1. Usuario selecciona la opción Archivo -> añadir MDT
2. Usuario busca la ruta al fichero
3. Selecciona el fichero en formato bil y le da a aceptar
4. El sistema inicia procedimiento de *"tileado"* y por niveles de resolución del MDT.
5. El sistema guarda en la caché local el MDT
6. El sistema aplica la capa MDT a la visualización 3D.

Escenarios Alternativos

- El fichero bil es demasiado grande y se lanza un error.
- El usuario cancela en mitad del proceso.

3.4.2 Caso de uso: El usuario consulta un servidor WMS

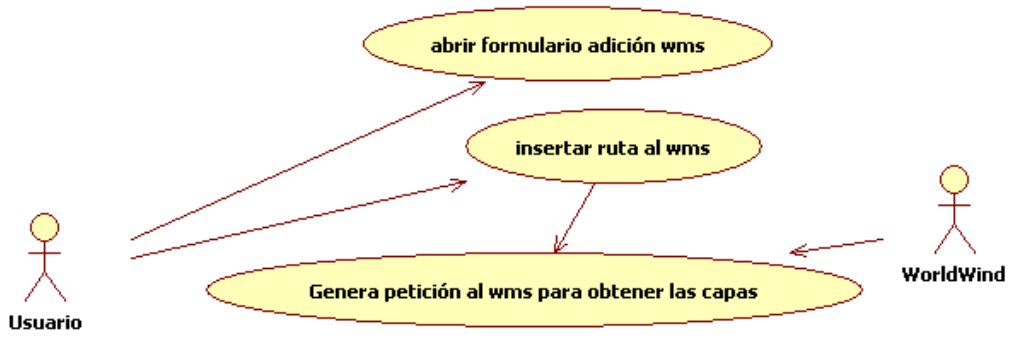

Ilustración 19 – Caso de uso consultar capas en servidor WMS

Caso de Uso: Consultar capas disponibles en un servidor WMS

Precondiciones: La dirección del servidor ha de ser correcta

Postcondiciones: El usuario visualizará una lista de capas que podrá añadir.

Descripción: Obtiene una lista de capas aplicables.

Resumen: El usuario desea consultar las capas ofrecidas por un servidor WMS concreto.

Actores: Usuario(Principal)

Escenario Principal

1. El usuario presiona el botón de añadir WMS
2. El usuario presiona la opción de añadir un nuevo servidor, representado como un "+".
3. El usuario introduce la dirección del servidor WMS
4. El usuario acepta
5. El sistema genera la petición *"GetCapabilities"* y *"GetResource"* al servidor para obtener las capas que ofrece el servicio.

6. El sistema lista las capas ofrecidas.

Escenarios Alternativos

- La dirección del servidor no es correcta o el servicio no está disponible

3.4.3 Caso de uso: El usuario añade un servidor WMS e inserta una capa en el globo worldwindcanvasGL (Bola del mundo)

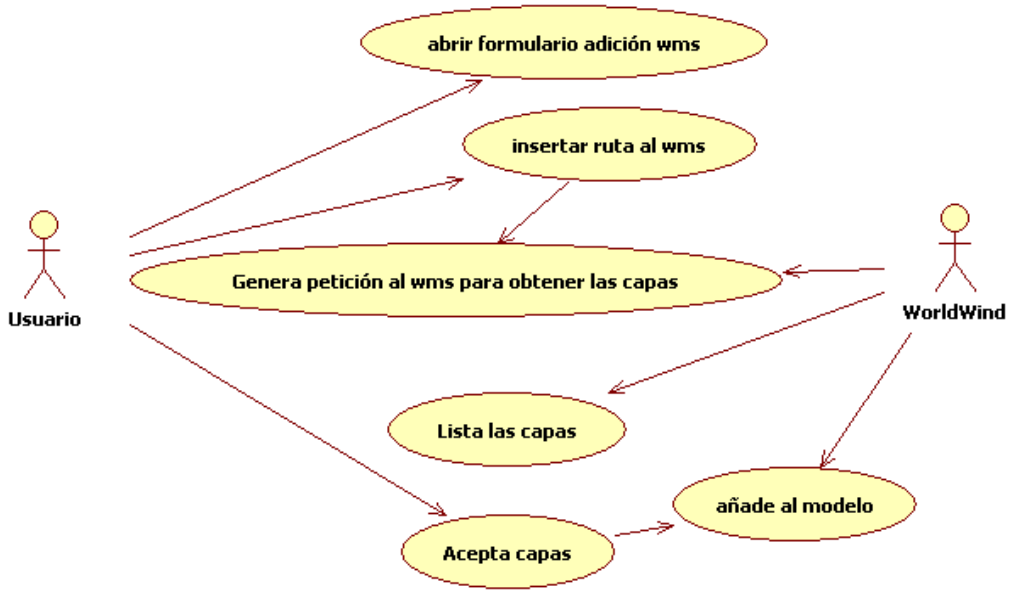

Ilustración 20 – Caso de uso añadir capas WMS

Caso de Uso: Añadir capas disponibles en un servidor WMS

Precondiciones: La dirección del servidor ha de ser correcta

Postcondiciones: El usuario visualizará una lista de capas que podrá añadir y añade una o varias.

Descripción: Obtiene una lista de capas aplicables y añade una o varias.

Resumen: El usuario desea consultar las capas ofrecidas por un servidor WMS concreto y aplicar alguna de ellas en el modelo 3D.

Actores: Usuario(Principal)

Escenario Principal

1. El usuario presiona el botón de añadir WMS
2. El usuario presiona la opción de añadir un nuevo servidor, representado como un "+".
3. El usuario introduce la dirección del servidor WMS
4. El usuario acepta
5. El sistema genera la petición *"GetCapabilities"* y *"GetResource"* al servidor para obtener las capas que ofrece el servicio.
6. El sistema lista las capas ofrecidas.
7. El usuario selecciona una capa de las ofrecidas.
8. El sistema crea en la caché el directorio para esa capa y su fichero de configuración XML.
9. El sistema añade la capa al modelo y ejecuta las peticiones necesarias para ir obteniendo los *"tiles"* presentes en la vista.

Escenarios Alternativos

- La dirección del servidor no es correcta o el servicio no está disponible.
- La capa no es compatible con WorldWind, no está estandarizada.

3.4.4 Caso de uso: Añadir una BBDD con puntos de interés, nombres de empresas o similar.

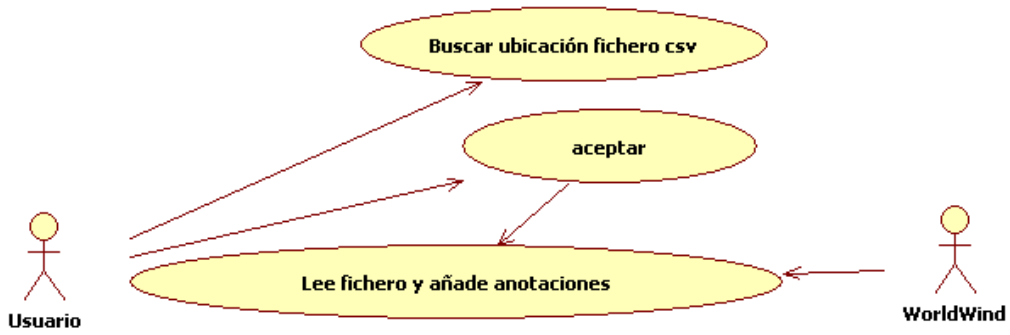

Ilustración 21 – Caso de uso añadir BBDD en formato CSV.

Caso de Uso: Añadir BBDD en formato csv

Precondiciones: El archivo debe estar en el formato correcto

Postcondiciones: El usuario añadirá una lista de topónimos que podrá visualizar.

Descripción: El usuario, mediante un fichero extraído de una base de datos, añadirá un gran número de topónimos sin mucho esfuerzo

Resumen: El usuario añade un gran volumen de información en forma de topónimos.

Actores: Usuario(Principal)

Escenario Principal

1. El usuario selecciona Archivo -> añadir topónimos desde CSV
2. El usuario busca el fichero

3. El usuario acepta el fichero
4. El sistema realiza un recorrido por el fichero y va añadiendo los topónimos a una capa especial creada con ese fin.
5. El usuario visualiza los topónimos en pantalla.

Escenarios Alternativos

- El fichero no está en un formato correcto

En este caso la estructura del fichero debe ser la siguiente:

<Nombre o descripción del POI>, <Latitud campo numérico>,<Longitud campo numérico>;

Más información sobre el formato en el anexo **Fichero CSV**.

3.4.5 Caso de uso: Añadir un topónimo mediante la IGU.

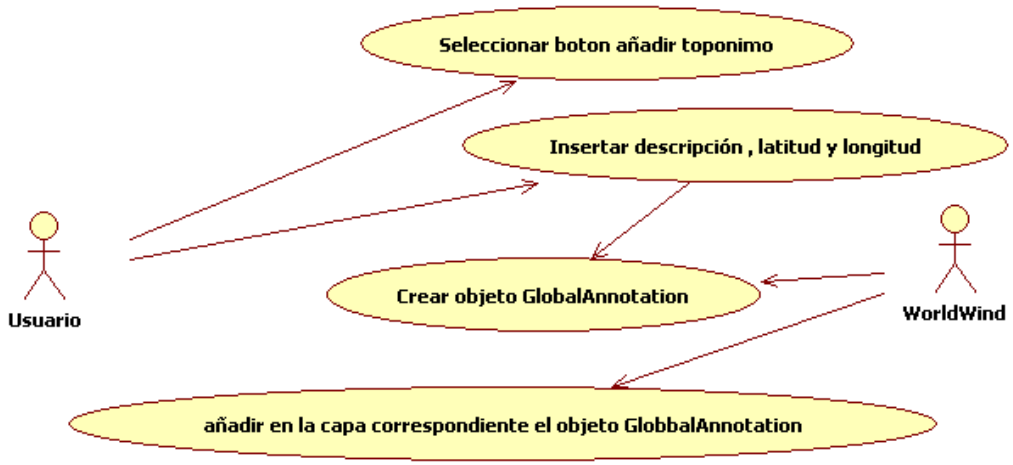

Ilustración 22 – Caso de uso añadir topónimo

Caso de Uso: Añadir un topónimo mediante la IGU

Precondiciones:

Postcondiciones: El usuario añadirá un topónimo en la posición indicada

Descripción: El usuario añade topónimo

Resumen: El usuario desea añadir un topónimo en una ubicación conocida.

Actores: Usuario(Principal)

Escenario Principal

1. El usuario presiona el botón de añadir topónimo.
2. El usuario introduce la nombre del topónimo (Ej. Murcia) y su posición (Latitud, Longitud).
3. El usuario acepta.
4. El sistema crea un objeto GlobalAnnotation con unas características predefinidas
5. El sistema añade este objeto a la capa de topónimos correspondiente.

Escenarios Alternativos

- La posición no es válida

3.4.6 Caso de uso: Aumentar exageración vertical del MDT

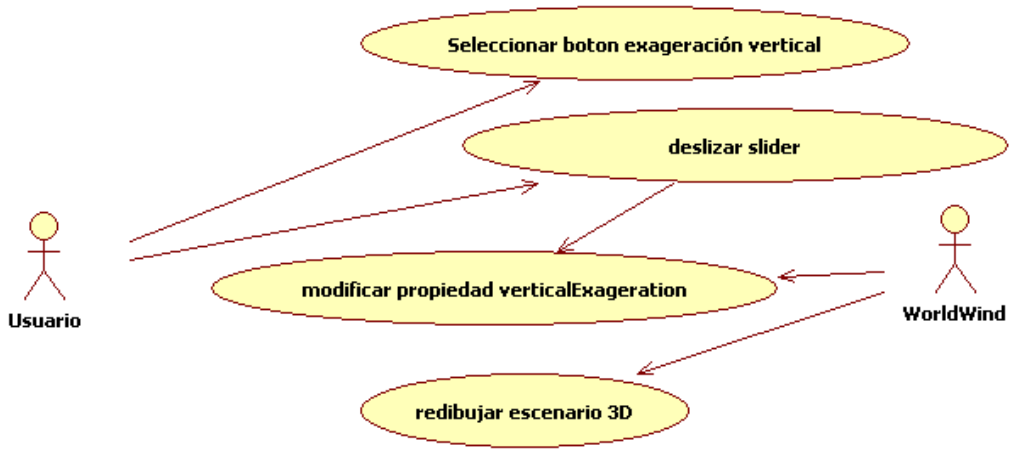

Ilustración 23 – Caso de uso modificar exageración vertical

Caso de Uso: modificar exageración vertical

Precondiciones:

Postcondiciones:

Descripción: El usuario modifica la exageración vertical

Resumen: El usuario modifica la exageración vertical con el fin de tener mejor detalle de algún punto del MDT(Global o propio).

Actores: Usuario(Principal)

Escenario Principal

1. El usuario presiona el botón Modificar Exageración Vertical
2. El usuario desliza el slider a derecha si desea aumentar, o izquierda si desea disminuir.
3. El sistema modifica la propiedad vertical Exageration.
4. El sistema redibuja el escenario3D

Escenarios Alternativos

3.4.7 Caso de uso: Viajar a un punto

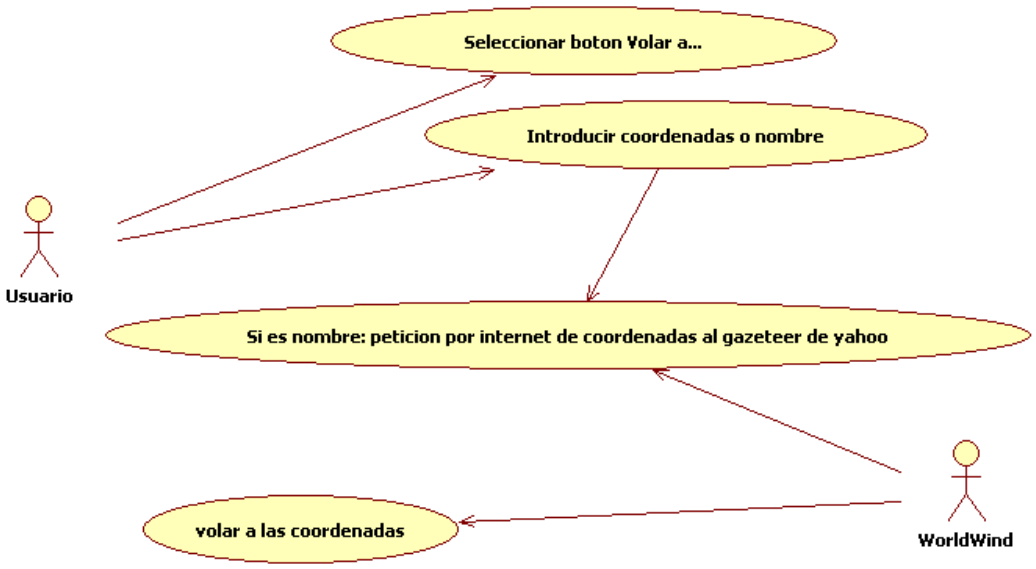

Ilustración 24 – Caso de uso volar a un punto

Caso de Uso: Volar a un punto

Precondiciones:

Postcondiciones: el usuario volará al punto deseado

Descripción: El usuario viaja a un punto

Resumen: El usuario introduce un nombre o coordenadas y el sistema le lleva hasta el lugar indicado

Actores: Usuario(Principal)

Escenario Principal

1. El usuario presiona el botón Volar a…
2. El usuario introduce el nombre del lugar o las coordenadas
3. El usuario acepta.

3.1. Si se ha hecho una búsqueda por nombre el sistema realiza una petición al gazeeter de Yahoo con el fin de que este le devuelva las coordenadas geográficas.

4. El sistema realiza una animación en forma de vuelo hasta el punto.

Escenarios Alternativos

- El nombre del lugar no existe o no se encuentra
- Las coordenadas no son válidas.

3.4.8 Caso de uso: Añadir imagen raster (Jpg, Png …)

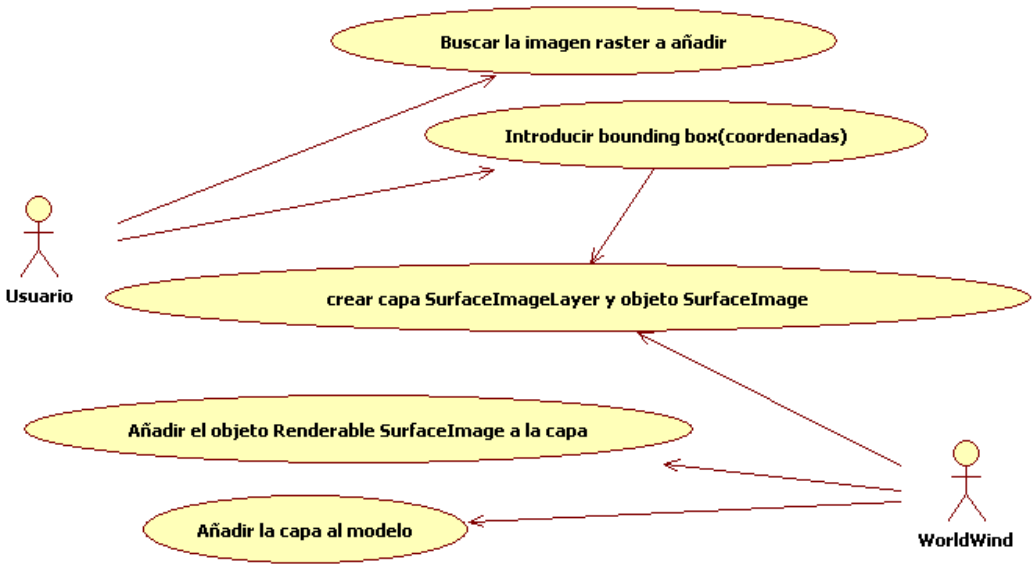

Ilustración 25 – Caso de uso añadir imagen raster

Caso de Uso: Añadir imagen raster

Precondiciones: la imagen debe encontrarse en un formato aceptado

Postcondiciones: El usuario añadirá una imagen raster en el lugar deseado.

Descripción: El usuario añade una imagen raster.

Resumen: El usuario introduce una imagen proporcionando el bounding box la de misma, o sea las coordenadas que ocuparán dos de sus esquinas, con esto se crea un cuadrado que es lo que ocupará la imagen.

Actores: Usuario(Principal)

Escenario Principal

1. El usuario selecciona Archivo -> añadir imagen raster
2. El usuario busca la ruta al archivo y lo selecciona
3. El usuario acepta
4. El usuario introduce las coordenadas del bounding box
5. El sistema crea una nueva capa de la clase SurfaceImageLayer
6. El sistema crea un nuevo objeto de la clase SurfaceImage
7. El sistema introduce el objeto en la capa
8. El sistema añade la capa al modelo
9. El sistema redibuja para mostrar la imagen añadida

Escenarios Alternativos

- El formato de la imagen no es válido
- Las coordenadas no son válidas.

3.4.9 Caso de uso: Instalar Ortofoto referenciada geográficamente

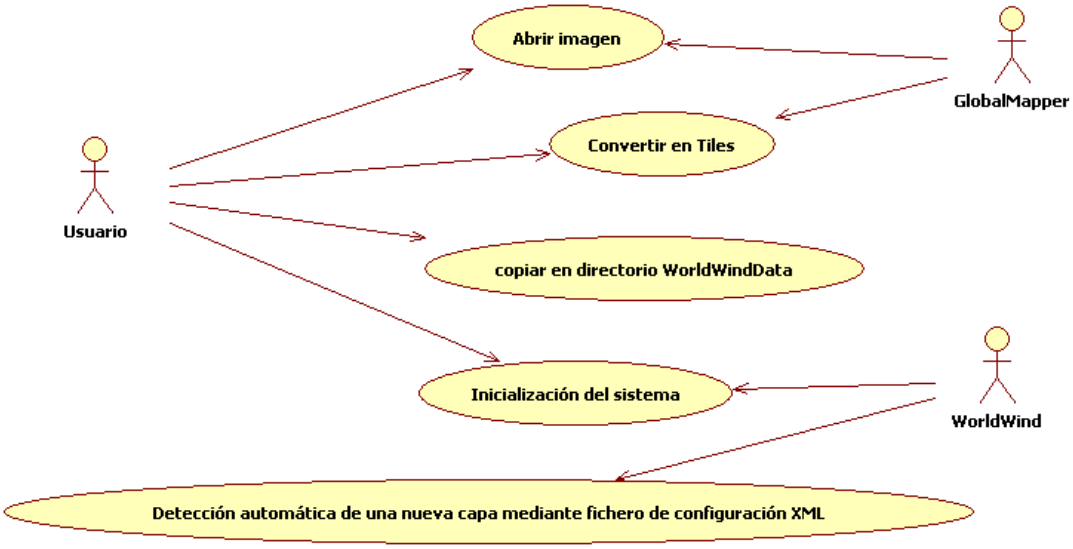

Ilustración 26 - Caso de uso añadir imagen georeferenciada

Caso de Uso: Añadir ortofoto georeferenciada

Precondiciones: la imagen debe encontrarse en un formato aceptado, disponer del programa externo global mapper.

Postcondiciones: El usuario añadirá una ortofoto georeferenciada

Descripción: El usuario añade una ortofoto georeferenciada

Resumen: El usuario construye la estructura por niveles y en tiles de la ortofoto mediante el uso del programa global mapper, luego lo añade al sistema worldwind.

Actores: Usuario(Principal)

Escenario Principal

1. El usuario abre la imagen con el programa global mapper
2. El usuario exporta la imagen a formato worldwind mediante global mapper
3. El usuario copia el fichero de configuración creado por global mapper dentro de la carpeta que contiene los niveles
4. El usuario copia el directorio que contiene los niveles y el fichero de configuración dentro de la carpeta WorldWindInstalled
5. El usuario inicia la aplicación worldwind
6. La aplicación realiza una comprobación de la carpeta WorldWindInstalled para verificar que haya nuevos datos
7. La aplicación detecta el nuevo fichero de configuración y crea la capa correspondiente para añadirla al modelo
8. La aplicación refresca la visualización para visualizar los nuevos datos.

Escenarios Alternativos

3.4.10 Caso de uso: Activar/Desactivar capas en el modelo 3D.

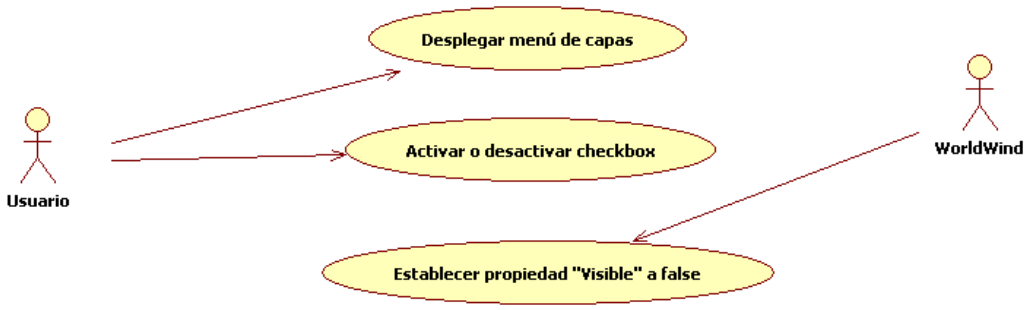

Ilustración 27 – Caso de uso Activar capas

Caso de Uso: Activar/desactivar capas en el modelo

Precondiciones: La capa debe estar incluida en el modelo y mostrarse en el listado

Postcondiciones: El usuario activará o desactivará la capa

Descripción: El usuario activará la visualización o no de una capa

Resumen: El usuario mediante la lista de capas podrá seleccionar aquellas que desea que sean visibles en el modelo 3D o no.

Actores: Usuario(Principal)

Escenario Principal

1. El usuario pulsa el botón de capas en la ventana principal
2. El sistema muestra una lista con las capas que actualmente se muestran en el modelo
3. El usuario selecciona una capa y activa/desactiva la misma
4. El sistema establece la propiedad de esa capa a true/false
5. El sistema refresca el modelo 3D para actualizar la vista.

Escenarios Alternativos

3.4.11 Caso de uso: Medición de áreas, distancias y otros parámetros.

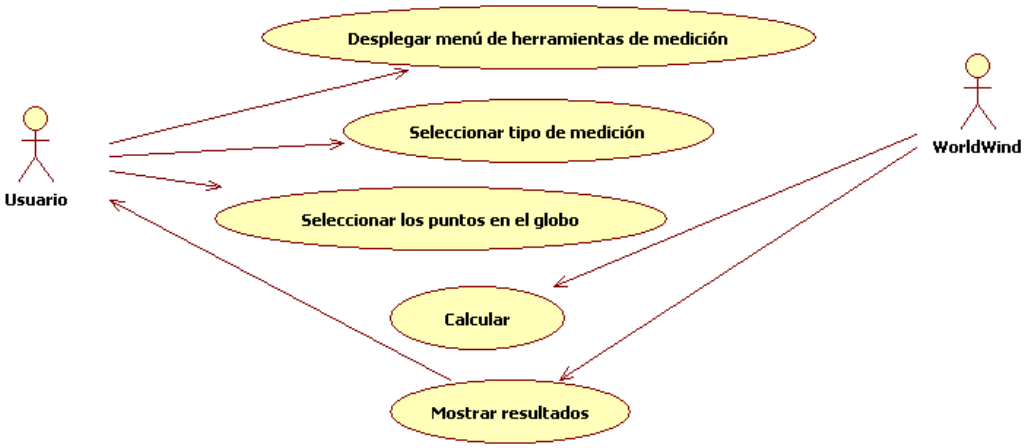

Ilustración 28 – Caso de uso Realizar mediciones en el modelo 3D

Caso de Uso: Mediciones en el modelo 3D

Precondiciones:

Postcondiciones: El usuario obtendrá los datos deseados

Descripción: El usuario mide áreas o distancias

Resumen: El usuario selecciona las herramientas de medición y selecciona la forma de medición, línea, polilínea, polígono regular o polígono irregular.

Actores: Usuario(Principal)

Escenario Principal

1. El usuario pincha en el botón de herramientas de medición
2. El usuario selecciona la herramienta de medición que se adecue a sus necesidades
3. El usuario selecciona los puntos en la pantalla formando una forma

4. El sistema realiza los cálculos en tiempo real y va mostrando en pantalla los datos

Escenarios Alternativos

3.4.12 Caso de uso: Buscar topónimo o similar

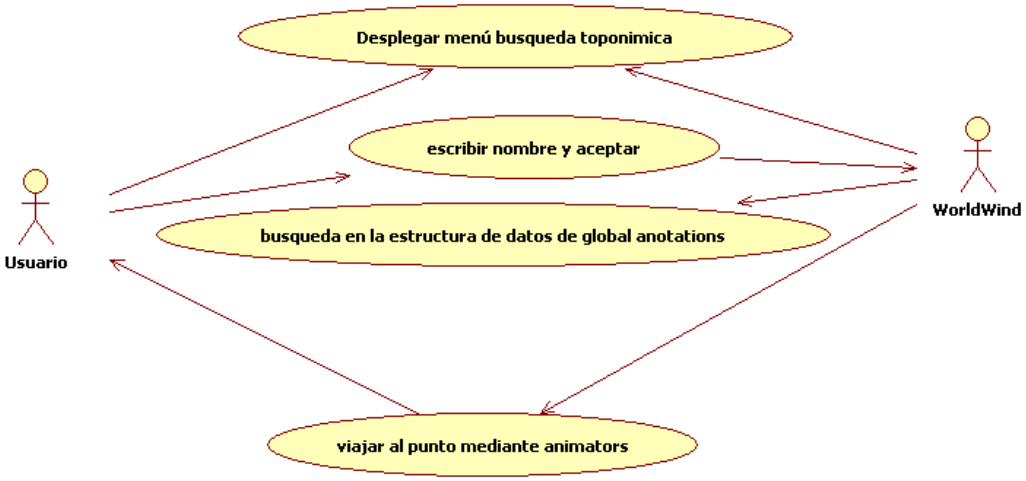

Ilustración 29 – Caso de uso búsqueda por topónimos

Caso de Uso: Búsqueda por topónimos

Precondiciones: el nombre del topónimo debe existir

Postcondiciones: el usuario obtendrá una lista con los resultados de su búsqueda y podrá ir al punto.

Descripción: El usuario realiza una búsqueda de los topónimos existentes.

Resumen: El usuario realiza una búsqueda de los topónimos existentes y viaja al de su interés.

Actores: Usuario(Principal)

Escenario Principal

1. El usuario presiona Edición -> Búsqueda toponímica
2. El usuario introduce el nombre del topónimo a buscar
3. El usuario acepta
4. El sistema realiza una búsqueda en la pila de objetos globalAnnotations de la capa topónimos
5. El sistema devuelve el o los topónimos demandados
6. El usuario pulsa sobre el de su interés y el sistema le lleva al lugar

Escenarios Alternativos

- No existe el topónimo a buscar

3.4.13 Caso de uso: búsquedas en web hacia el applet

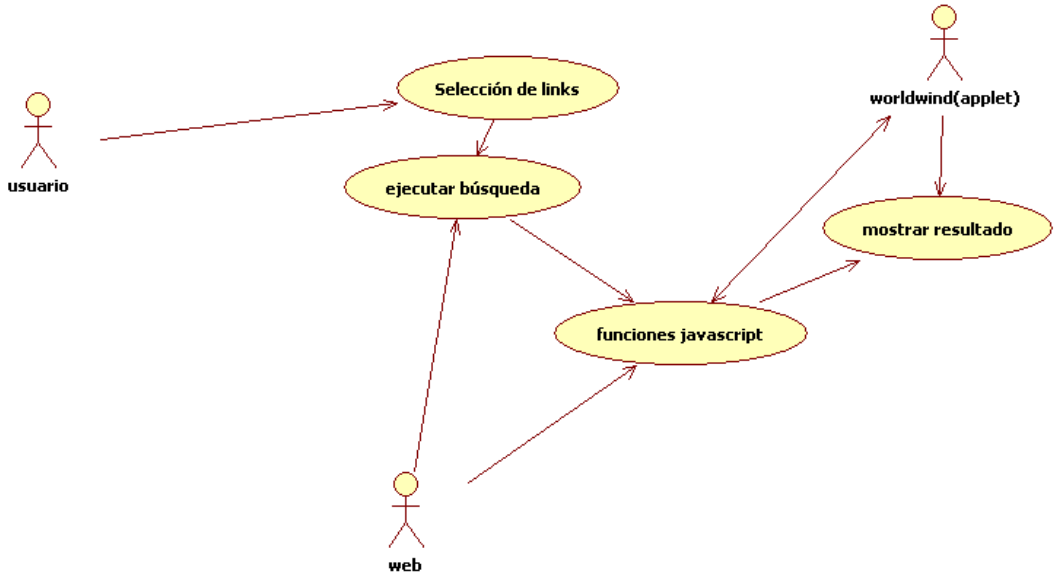

Ilustración 30 – Caso de uso búsquedas en applet

Caso de Uso: Búsquedas en web y comunicación con el applet

Precondiciones:

Postcondiciones: El usuario irá al punto buscado

Descripción: El usuario viaja a un punto que ha buscado en una lista

Resumen: El usuario selecciona un topónimo de una lista o realiza una búsqueda

Actores: Usuario(Principal)

Escenario Principal

1. El usuario en la web selecciona alguna categoría de topónimos o bien realiza una búsqueda

1.1 La web ejecuta la búsqueda mediante un script php

2. Obtiene un listado con los topónimos
3. Selecciona el topónimo de su interés
4. La web ejecuta las funciones de javascript necesarias para acceder a los métodos públicos del applet
5. El sistema, en este caso el applet, realiza la parte propia del programa, viajar al topónimo y refrescar la vista.

Escenarios Alternativos

- El nombre del lugar no existe o no se encuentra

3.5 – CODIFICACIÓN

Una vez definidos los casos de uso deseados, se pasó a la codificación del sistema deseado, introduciendo los ajustes necesarios para adaptarlo a los lenguajes de implementación utilizados (Java).

Método de trabajo

Desde un primer momento se codificaron las clases diseñadas con la intención de generar una librería de clases reutilizables. Para ello hubo que hacer hincapié en la estandarización del código fuente, se utilizaron un formato fijo de secciones en cada uno de los ficheros fuente, y se documentaron las funciones y métodos de una manera estandarizada que facilitara su reutilización y rápida comprensión , así como su extracción automática para generar un manual del programador.

Tras la codificación de cada clase se procedía a un testeo concienzudo mediante la ejecución de numerosos caso de prueba de cada uno de los métodos de las clase, buscando posibles bugs prematuros, trazándole adecuadamente mediante la inserción de breakpoints en los puntos del código a priori conflictivos y llamando a dichos métodos con unos parámetros incorrectos que pudieran hacerlas fallar(valores nulos, negativos, etcétera).

Una vez que dicha biblioteca de clases fue generada y correctamente testada este proyecto ha actuado como un "reutilizador más" de la biblioteca de clases.

En la codificación se optó por codificar de forma legible, para favorecer el mantenimiento, y buscando siempre el estilo adecuado de Java. Se han aprovechado en la medida de lo posible los potenciales de este "lenguaje mejorado". Además, se ha tendido a una programación lo más estandarizada posible, buscando una portabilidad máxima.

La codificación como proceso se ha dividido en tres partes diferenciadas:

- Codificación y testeo de cada una de las clases base del proyecto.
- Pruebas concienzudas
- Integración con el interfaz principal

Estandarización del Código Fuente

Para la estandarización del código fuente se han utilizado los patrones definidos por [Johnson, 1996] y [Hoff's, 1995]. A continuación se detallan las más significativas o características:

- Longitud de las líneas de código inferior a 80 caracteres, o en su defecto separación en varias líneas gracias al metacaracter '\', que escapa el retorno de carro introducido.
- Indentación al mismo nivel de las llaves de apertura y clausura que delimitan los bloques de código, para bloques con más de una línea de código.
- Asteriscos posicionados junto al tipo al que apuntan (en la declaración de un puntero), y no junto al nombre de la variable, especificando así el tipo utilizado, y evitando confusiones. Todas las variables son inicializadas al ser declaradas.
- Declaración de las variables locales a una parte de un método (por ejemplo las que aparecen en un bucle) en el momento de uso.
- Cada fichero llevará una cabecera de unas 20 líneas explicando qué contiene.
- Cada método llevará una cabecera explicando su funcionalidad, valores de retorno y parámetros.
- Todas las clases presentarán (aunque no estén codificados) un constructor por defecto, un constructor copia, una sobrecarga del operador "<<" para poder imprimirlas en flujos y un destructor virtual.

Los Ficheros Fuente de las Clases

Cada clase presenta un fichero *.java, que es la extensión de los ficheros fuente en este lenguaje.

En todos ellos se incluye una cabecera previa indicando a qué desarrollo pertenecen, información de contacto y demás.

Las funciones tienen una explicación previa en la que aparecen los siguientes campos:

- DESCRIPCIÓN: Breve descripción de la tarea realizada por la función. Ej.: Busca por nombre una capa
- PARÁMETROS: Explicación de cada uno de los parámetros que recibe la función. Ej.: visible -> true/false explícita si la capa está visible o no en el modelo.
- VALORES DE RETORNO: Valores de retorno de la función junto a una breve explicación de su significado. En todas ellas se ha utilizado el 0 como un funcionamiento normal de la función. Ej.: 1 -> Error. No hay elementos (vacía)

Se ha cuidado también el mantener en la medida de lo posible la seguridad en los accesos a los datos, estableciendo a los objetos los niveles de acceso adecuados pudiendo ser, públicos, protegidos o privados. En la mayoría de los casos se ha optado por establecerlos privados o protegidos, salvo algún caso especial en el que el acceso tuviese que realizarse específicamente al objeto en cuestión y no a través de algún método público. Se ha diferenciado claramente también entre los métodos internos de cada clase y los métodos públicos, atendiendo siempre a buscar el mayor índice de seguridad posible.

Webs

Para las Webs de prueba, como ya se ha comentado se han realizado utilizando Adobe Dreamweaver, se componen de tres partes divididas en tres frames, una parte de cabecera en la cual incluimos información tal como logos y demás cosas que deseamos que no cambien, un frame en el cual estará embebido el applet y que no debe refrescarse y una parte dinámica que cambiará y se refrescará para comunicarse con el frame que contiene al applet y mediante javascript poder acceder a los datos del applet.

Se ha utilizado código estandarizado de HTML, en su mayor parte producido por el propio programa Dreamweaver utilizando la vista en diseño, pero siguiendo siempre las convenciones de *world wide web consortium.*

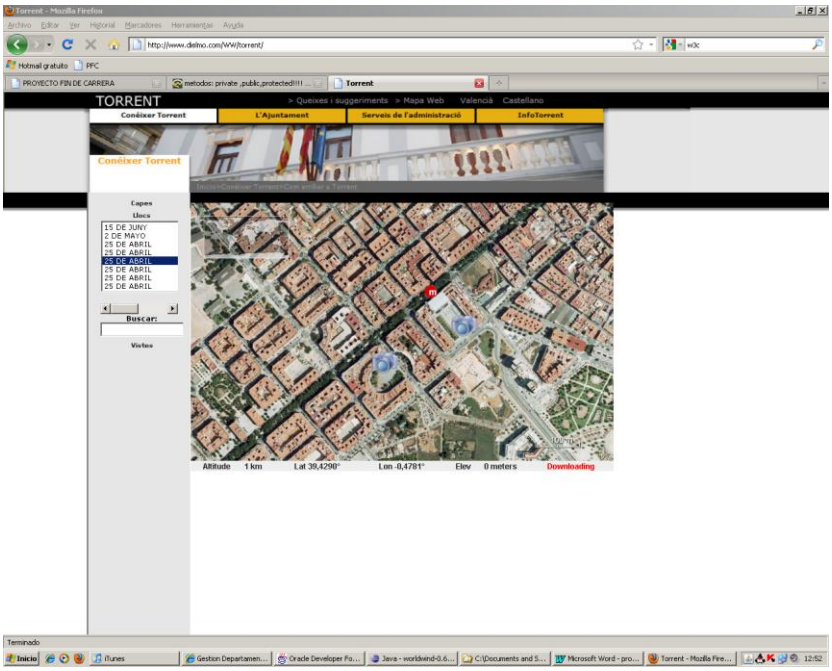

Ilustración 31 – Imagen del applet embebido en una web

Para el desarrollo de las funciones en javascript se ha utilizado un estilo similar a la programación en Java, salvando las distancias entre los dos lenguajes. Se ha incluido en las páginas en las cuales era necesario realizar alguna comunicación con el applet, lo mismo podemos decir de PHP, el cual únicamente se ha utilizado en un par de páginas en las cuales era necesario realizar alguna búsqueda en una base de datos con formato CSV.

Autor: José Vicente Mora Espí
Escuela técnica superior de ingeniería informática

IV – Conclusiones

4.1 – CONCLUSIONES DEL PROYECTO

La primera conclusión que podemos sacar es que WorldWind Escenarios Virtuales ha cumplido con la mayoría de los objetivos propuestos inicialmente.

Cumple con todas las bases de un visor 3D. Proporciona modos de navegación tanto con ratón como mediante teclado, proporciona la adición de nuevos datos y la carga y guardado de los mismos tanto en modo local, como mediante servidores WMS.

También existe la posibilidad de crear mapas con alguno de los programas anteriormente mencionados e insertarlos en el globo 3D. Lo cual proporciona un alto grado de personalización.

Durante todo el proceso de desarrollo de este proyecto, las dificultades encontradas no han sido pocas, por una parte tenemos el desconocimiento total del proyecto, la forma en que estaba hecho y/o estructurado, las estructuras de datos utilizados etc. Lo cual me produjo no pocos dolores de cabeza y muchas semanas de leer código, pruebas inútiles y un lento proceso de aprendizaje a base de prueba y error y preguntar y leer muchos foros, además de consultar numerosos ejemplos e incluso utilizar técnicas de ingeniería inversa como decompiladores para los casos en los que el código no era accesible.

Otra gran dificultad encontrada ha sido el desconocimiento del funcionamiento de diversas tecnologías de servicios de mapas y datos, como los servidores WFS, los cuales no se ha podido hacer funcionar en este proyecto, pero que ofrecen una alternativa mucho más eficiente a la hora de mostrar datos geográficos como pueden ser topónimos o capas en formato shape. Todo esto es mejorable con una mayor formación y conocimiento en este campo.

También se ha tenido problemas a la hora de crear un soporte especifico para mostrar estructuras en 3D, como pudieran ser edificios, de modo que el escenario

quedase de un modo mucho más realista, se encontraron diversas soluciones pero ninguna de ellas lo suficientemente eficiente, en gran parte por el desconocimiento en cuanto a las librerías 3D utilizadas, en este caso JOGL. Se consiguió cargar pequeños grupos de edificios, pero en ningún caso algo lo suficientemente vistoso, sin que ello comprometiera excesivamente el rendimiento de la aplicación.

Otro aspecto a mejorar es la integración en web de las aplicaciones o applets, ya que no se pudo encontrar una forma óptima para la carga de topónimos debido a la imposibilidad de acceder a los archivos que contenían los mismos, la solución propuesta fue la de realizar la carga de los mismos durante la carga de la página web, una solución de compromiso ya que para grandes volúmenes de datos esto podría afectar en forma de rendimiento negativo a la carga de la página.

Aunque las dificultades encontradas y las horas aplicadas a este proyecto han sido muchas, tras la finalización del mismo he comprendido que quizá hubiesen sido necesarias más de una persona para la realización del mismo, debido a la complejidad de alguna función del mismo, como por ejemplo alguien que tuviese conocimientos ya en programación en 3D, o alguien que hubiese trabajado con WFS y WMS, aunque ello conlleve mantener canales de comunicación activos que "aumenten" los costes para evitar problemas de desinformación. Por otra parte aunque mucha gente quiera dejarlo de lado, la potencia que proporciona JAVA es enorme, y muchas veces el problema se encuentra más en el programador que en las herramientas utilizadas.

Otra cosa que he aprendido durante la realización de Este proyecto ha sido que a veces no hay que buscar el equilibrio entre tiempo y espacio, ya que por ejemplo en este tipo de aplicaciones importa mucho más que los datos se muestren correctamente que antes que lo que ocupen en memoria.

Para terminar, sólo comentar que al ser este proyecto software libre espero pueda ayudar a otras personas a desarrollar más aplicaciones y usos para el JAVA SDK de WorldWind y que sirva de ejemplo para aquellos que así lo deseen, personalmente no he encontrado muchas referencias de cómo programar muchos

aspectos necesarios y gracias a comentarios en comunidades como worldwindforums y esfuerzo, he podido solventarlas, otras desgraciadamente no. Con esta aplicación esperamos aportar nuestro granito de arena a la comunidad para allanar el camino a otras personas.

4.2 – TRABAJO FUTURO Y AMPLIACIONES

El presente proyecto puede ser ampliamente retomado por cualquier otra persona que esté interesada en la visualización 3D, son muchos los frentes que quedan abiertos, y en los que futuros desarrollos pueden ahondar ya que queda mucho trabajo por hacer, desde la implementación de estándares de facto como son KML o CityGML, a la implementación y correcta visualización de los servidores WFS, encargados generalmente de mostrar información de tipo vectorial, ahora mismo únicamente representable en modo local.

Se han observado navegando por la web muestras de lo que WorldWind Java es capaz, ya que hemos visto vídeos en los que se cargan multitud de objetos 3D con texturas, esto es algo muy interesante, ya que dota de un mayor realismo si cabe a un escenario 3D, mostrar una ciudad en la cual aparezcan todos sus edificios con sus respectivas texturas es un proyecto ambicioso, lamentablemente el código que probamos no fue suficientemente eficiente y únicamente era capaz de mostrar objetos en 3D sin textura, y el sistema sufría mucho para mostrarlos, se ralentizaba en exceso. Quizá una optimización de las librerías en 3D o la utilización de las mismas seria suficiente.

La optimización de la muestra de topónimos, ya que al mostrar una cantidad elevada, se muestra una ralentización bastante molesta, es otro de los factores que podrían ser analizados en un futuro, si alguien desea retomar este proyecto.

Implementación de calculadores de rutas a modo de GPS, sería muy interesante el poder buscar rutas, a través de caminos, calles y demás, se conoce la alta dificultad de este propósito pero no por ello es desdeñable como tal.

Perfeccionamiento de los métodos de carga de archivos “Shape”, ya que ahora mismo no es 100% compatible con el estándar ESRI Shapefile, hemos tenido algunos problemas al cargar algunos ficheros de este tipo.

V – BIBLIOGRAFÍA

5.1 – BIBLIOGRAFÍA CONSULTADA

• **[Arnold y Gosling, 1997]** Ken Arnold y James Gosling. Addison-Wesley/Domo. "El lenguaje de Programación Java". Wesley Iberoamericana. 1997. 334 páginas. (Muy básico, escrito por el desarrollador del lenguaje).

• **[Eckel, 1997]** Bruce Eckel. "Hands -on Java Seminar". Presindent MindView Inc. 1997. 577 páginas. (Tutorial completo en Inglés en formato PDF).

• **[García et al., 1999]**. Javier Garciá de Jalón, José Ignacio Rodríguez, Iñigo Mingo, Aitor Imaz, Alfonso Brazález, Alberto Larzabal, Jesús Calleja y Jon García. "Aprenda Java como si estuviera en primero". Universidad de Navarra. 1999. 140 páginas. (Tutorial muy básico, en el que se trata la potencia de Java, pero sin profundizar en ningún tema en particular).

• **[García, 1997]** Francisco José García Peñalvo. "Apuntes de teoría de la asignatura Programación Avanzada del tercer curso de Ingeniería Técnica en Informática de Gestión". Universidad de Burgos. 1997. 216 páginas. (Comentan la teoría de la programación orientación al objeto. Han sido resumidos para dar una visión de la programación orientada a objeto en el apartado I.1).

• **[Johnson, 1996]** Jeff Johnson. "Coding Standards for C, C++, and Java". Vision 2000 CCS Package and Application Team. 1996. 14 páginas. (Consejos para el formato de los fuentes Java, C y C++).

• **[Morgan, 1999]** Mike Morgan. "Descubre Java 1.2". Prentice Hall. 1999. 675 páginas. (Sin duda muy interesante, sobre todo por su actualidad al tratar con Java 2, y por su extensión al tratar todas las bibliotecas de Java).

• **[Naughton, 1996]** Patrick Naughton. "Manual de Java". Mc. Graw Hill 1996. 395 páginas. (Introduce todos los aspectos de la programación básica en Java).

• **[van Hoff et al., 1996]** Arthur van Hoff, Sami Shaioi y Orca Starbuck. "Hooked on Java". Addison-Weslet. 1996. (Todo lo que hace falta saber para crear applets, con muchos ejemplos. En inglés).

• **[Zolli, 1997]** Andrew Zolli. "La biblia de Java". Anaya multimedia. 1997. 814 páginas. (Completo en lo a que biliotecas del JDK 1.1 se refiere).

• **[Froufe, 1997]** Agustín Froufe. "Tutorial de Java". Facultad de Informática de Sevilla. 1997. http://www.fie.us.es/info/internet/JAVA/.

• **[Kent Beck, 1999]** "Extreme Programming Explained: Embrance Change". Addison-Wesley Publishers (Explicación completa sobre la programación extrema).

5.2 – BIBLIOGRAFÍA ADICIONAL

http://www.lcc.uma.es/~ppgg/PFC/ - Página enfocada a las buenas prácticas en la realización de un PFC.

http://pisuerga.inf.ubu.es/lsi/Docencia/TFC/ITIG/icruzadn/Memoria/index.htm - Memoria de un PFC que ha sido muy útil de cara a la realización de este.

http://worldwind.arc.nasa.gov/java/ - Página principal del proyecto WorldWind Java.

http://worldwind.arc.nasa.gov/java/demos/ - Página con una serie de demos muy útiles para empezar con el proyecto.

http://worldwind.arc.nasa.gov/java/demos/index_applet_text_and_links.html -Demos acerda de convertir la aplicación de escritorio en un applet.

http://worldwindcentral.com/wiki/Java - Wiki principal de la aplicación

http://en.wikipedia.org/wiki/NASA_World_Wind - Explicación en wikipedia del proyecto WorldWind

http://www.mountaincartography.org/publications/papers/papers_lenk_08/weber.pdf - Interesante artículo sobre la cartografía 3D

http://java.sun.com/developer/technicalArticles/javase/worldwind/ - Guía técnica de desarrollo en java worldwind.

http://java.about.com/b/2008/10/28/nasa-world-wind-java-applet.htm - HOWTO acerca de los applets en NASA WorldWind

http://www.ibm.com/developerworks/java/library/j-wwj/?ca=dgrlnxw97wwjsdkeclipse&S_TACT=105AGX59&S_CMP=GR – Pequeño artículo acerca de la integración de worldWind con Eclipse

http://patmurris.blogspot.com/2008/05/nasa-world-wind-java-sdk-05-released.html - Blog de uno de los principales activos de este proyecto, uno de sus desarrolladores principales.

http://geotux.tuxfamily.org/index.php?option=com_content&task=view&id=16&Itemid=11 – Página orientada a la geolocalización en Linux.

http://weblogs.java.net/blog/2007/05/30/geotagging-nasa-world-wind - Artículo sobre cómo geoetiquetar localizaciones usando nasa worldwind.

http://nasa.atompedia.com/es/nasa-satelite/nasa-world-wind/nasa-world-windwms - Página de consulta de algunos de los principales servidores WMS para worldwind.

http://www.sigte.udg.edu/jornadassiglibre2009/uploads/Articulos/C35.pdf -Portfolio de las jornadas de SIG 2009.

http://www.opensourceworldconference.com/malaga10/sites/default/files/sig3d.pdf - Artículo sobre las jornadas OpenSource 3D de Málaga 2010.

http://es.wikipedia.org/wiki/SIG - Artículo de la Wikipedia sobre SIG.

http://www.gabrielortiz.com/ - Página/Foro muy activo sobre GIS y cartografía

http://eureka-geo.com/wordpress/?p=21 – Foro acerca de cartografía

http://igosoftware.wordpress.com/2010/02/24/girona-2010-comunicaciondesarrollo-de-aplicaciones-sig-3d-personalizadas/ - Página de una empresa española dedicada a la cartografía y visualización 3D.

Autor: José Vicente Mora Espí
Escuela técnica superior de ingeniería informática

VI – ANEXOS

6.1 LICENCIA NASA WORLDWIND

NASA versión de código abierto ACUERDO 1,3

EL PRESENTE ACUERDO DE CÓDIGO ABIERTO ("ACUERDO") define los derechos de uso, reproducción, distribución, modificación y redistribución de los programas informáticos, originalmente publicado por el gobierno estadounidense, representado por la agencia gubernamental("Agencia del Gobierno"). EL GOBIERNO DE LOS ESTADOS UNIDOS, representada por una agencia gubernamental, es beneficiario TERCEROS de todas las distribuciones posteriores o redistribuciones DEL SOFTWARE. Cualquier persona que utilice, reproduzca, distribuya, modifique o redistribuya el software sujeto, como se define, o parte de el, por esa acción, ACEPTA POR COMPLETO las responsabilidades y obligaciones contenidas en este Acuerdo.

Agencia de Gobierno: Nacional de Aeronáutica y del Espacio (NASA)
Gobierno Designación Agencia Software Original: ARC-15166-1
Software Agencias de Gobierno Título Original: Versión 1,3 WorldWind
Registro de usuario requerido. Por favor, visita http://opensource.arc.nasa.gov/
Agencia del Gobierno Punto de Contacto para Software Original: Patrick.Hogan@ nasa.gov

1. DEFINICIONES

A. "Contribuyente" significa Agencia Gubernamental, ya que es el desarrollador del software original, y cualquier otra entidad que realiza una modificación.

B. "Las patentes cubiertas": los derechos de patente permisible por un contribuyente que son necesariamente infringidos por el uso o la venta de las modificaciones, solo o en combinación con el software.

C. "Display", la exhibición de una copia del Software, ya sea directamente o por medio de una imagen, o cualquier otro dispositivo.

D. "distribución", transporte o transferencia del Software en perjuicio, independientemente de los medios, a otro.

E. "Ampliar Obra" significa programas de computación que combina software sujetos o partes de ellos, con el software independiente del Software que no se rige por los términos de este Acuerdo.

F. "Modificación" cualquier alteración de, incluyendo la adición o supresión de la sustancia o la estructura de ya sea el software original o software sujeto, e incluye obras derivadas, como se define dicho término en el Estatuto de Derecho de Autor, 17 USC 101. Sin embargo, el acto de incluir softwares como parte de una obra mayor, no constituye en si misma una modificación.

G. "Original Software" es el primer ordenador de software liberado bajo esta Agencia Acuerdo por el Gobierno con el Gobierno Agencia designación ARC-15.166-1 y con derecho WorldWind, incluyendo el código fuente, código objeto y la documentación adjunta, en su caso.

H. "destinatario": cualquier persona que adquiere el software de reserva en virtud del presente Acuerdo, incluidos todos los contribuyentes.

I. "redistribución" medios de distribución del Software sujeto después de una modificación se ha hecho.

J. "Reproducción", la realización de una contraparte, la imagen o copia del Software.

K. "venta": el intercambio del software de reserva de dinero o de valor equivalente.

L. "Sin perjuicio de Software" es el software original, modificaciones, o de partes respectivas de la misma.

M. "uso" se refiere a la aplicación o el empleo del Software Tema para cualquier propósito.

2. OTORGAMIENTO DE DERECHOS

A. En el marco de los derechos de patente: Sujeto a los términos y condiciones del presente acuerdo, cada contribuyente, con respecto a su propia contribución al Software, otorga a cada persona una licencia no exclusiva, mundial, libre de regalías para participar en las siguientes actividades relacionadas con el Software Asunto:

1. Uso
2. Distribución
3. Reproducción
4. Modificación
5. Redistribución
6. Mostrar

B. En el marco de Patentes de los derechos: Sujeto a los términos y condiciones del presente Acuerdo, cada contribuyente, con respecto a su propia contribución al Software, otorga a cada beneficiario con arreglo a cubierto de Patentes una licencia no exclusiva, mundial, libre de regalías licencia para dedicarse a las siguientes actividades relacionadas con el Software:

1. Uso
2. Distribución
3. Reproducción
4. Venta
5. Oferta para la venta

C. Los derechos concedidos en virtud del artículo B. también se aplican a la combinación de una Modificación del colaborador y el software de reserva si, en el momento de agregar la modificación introducida por el Contribuyente, la adición

de tales causas Modificación de la combinación de ser cubiertos por las patentes objeto. No se aplica a cualquier otra combinación que incluye una modificación.

D. Los derechos reconocidos en los párrafos A. y B. permiten que el destinatario sublicencie los mismos derechos. Sublicenciar: Estos deberán estar bajo los mismos términos y condiciones del presente Acuerdo.

3. OBLIGACIONES DEL BENEFICIARIO

A. Distribución o redistribución del Software debe hacerse en virtud del presente Acuerdo, a excepción de las adiciones.

1. Cada vez que un destinatario distribuya o redistribuya el software, una copia de este Acuerdo debe ser incluido con cada copia del software de reserva, y

2. Si el o los destinatarios distribuyen o redistribuyen el software en perjuicio de cualquier otra forma de código fuente, el receptor también debe poner el código fuente disponible libremente, y deben adjuntar con cada copia del software con la información sobre cómo obtener el código fuente de manera razonable, en o a través de un medio habitualmente utilizado para el intercambio de software.

B. Cada beneficiario debe asegurarse de que el aviso de copyright siguiente aparece prominentemente en el Software Asunto:

Copyright (C) 2001 Gobierno de Estados Unidos
representada por el Administrador de la
Nacional de Aeronáutica y del Espacio.
Todos los derechos reservados.

C. Cada colaborador debe caracterizar su alteración del Software Asunto como una modificación y debe identificarse a sí mismo como el creador de las modificaciones, de manera que razonablemente permite a los destinatarios subsiguientes para identificar el origen de la modificación. En cumplimiento de

estos requisitos, colaborador debe incluir un archivo (por ejemplo, un archivo de registro de cambios) que describe las modificaciones introducidas y la fecha de las modificaciones, identifica Contribuyente como originador de las alteraciones, y consiente a la caracterización de las alteraciones como una modificación, por ejemplo, mediante la inclusión de una declaración de que la modificación se deriva, directa o indirectamente, del original software proporcionado por el Gobierno de la Agencia. Una vez concedida la autorización, éste ya no puede ser revocada.

D. Un colaborador puede añadir su propio aviso de copyright del Software Asunto. Una vez que un aviso de copyright se ha añadido al Software Asunto, un destinatario no se puede extirpar sin el permiso expreso del Contribuyente, que dio el aviso.

E. El Destinatario no puede hacer ninguna representación en el software de reserva o en cualquier promoción, publicidad o cualquier otro material que pueda interpretarse como una aprobación por el Gobierno o por cualquier Agencia de destinatarios antes de cualquier producto o servicio prestado por destinatario, o que busquen puede para obtener una ventaja comercial por el hecho de Gobierno Agencia o de la participación de un previo del destinatario en el presente Acuerdo.

F. En un esfuerzo por rastrear el uso y mantenimiento de registros precisos del Software Asunto, cada receptor, al recibir el Software Asunto, se le pide que se registra en el organismo gubernamental, visite el siguiente sitio web: http://opensource.arc.nasa. gov. Nombre del destinatario y la información personal se utilizará para fines estadísticos. Una vez que el beneficiario realiza una modificación disponible, se solicita que el destinatario informará a la Agencia del Gobierno en el sitio web proporcionado por encima de cómo acceder a la modificación.

G. Cada colaborador que representa que su modificación se cree que es creación original de colaborador y no viola los acuerdos, reglamentos, estatutos o reglamentos, y además que Contribuyente tiene derechos suficientes para conceder los derechos transmitidos por el presente Acuerdo.

H. El Destinatario podrá optar por ofrecer, y para cobrar una tasa a favor, garantía, soporte, indemnización y / u obligaciones de responsabilidad a uno o más destinatarios del Software Asunto. El Destinatario podrá hacerlo, sin embargo, sólo en su propio nombre y no en nombre de la agencia gubernamental o cualquier otro destinatario. Tal beneficiario debe dejar absolutamente claro que cualquier de tal garantía, soporte, indemnización y / o obligación de responsabilidad que se ofrece a través de destinatarios solo. Además, el beneficiario ésta acuerda indemnizar a la Agencia del Gobierno y cada destinatario por cualquier responsabilidad incurrida por ellos como consecuencia de la garantía, soporte, indemnización y / o responsabilidad ofrecidos por destinatarios tales.

I. El Destinatario puede crear una obra más amplia mediante la combinación de software con el software por separado no se rige por los términos de este acuerdo y distribuir la obra más grande como un único producto. En tal caso, el beneficiario debe asegurarse de Software Asunto, o partes de ellos, incluido en el trabajo de mayor tamaño son sujetos a este Acuerdo.

J. No obstante las disposiciones contenidas en este documento, el beneficiario queda poner sobre aviso de que la exportación de cualquier mercancía o datos técnicos de los Estados Unidos pueden requerir algún tipo de licencia de exportación del Gobierno de los EE.UU.. El no obtener las licencias necesarias de exportación podrá dar lugar a responsabilidad penal en virtud de leyes de los EE.UU.. Agencia del Gobierno no representa que la licencia no se exigirá ni que, en caso necesario, se publicará en breve. Nada de lo que aquí se ofrece una licencia de importación tales.

4. RENUNCIA DE GARANTÍAS Y RESPONSABILIDAD; RENUNCIA E INDEMNIZACIÓN

R: No Garantía: OBJETO DEL SOFTWARE SE ENTREGA "TAL CUAL" SIN GARANTÍA DE NINGÚN TIPO, ya sea expresa, implícita o legal, incluyendo, pero no limitado a, cualquier garantía de que el software sujeto se ajusten a las especificaciones, CUALQUIER GARANTÍA IMPLÍCITA DE COMERCIALIZACIÓN, IDONEIDAD PARA UN PROPÓSITO PARTICULAR, O LA LIBERTAD DE INFRACCIÓN, CUALQUIER GARANTÍA DE QUE EL SOFTWARE tema será libre de errores, o cualquier otra garantía que la documentación, si se proporciona, se ajustará a las SOFTWARE TEMA. El presente Acuerdo no, de ninguna manera, constituye un aval por una agencia gubernamental o cualquier otro receptor ANTES de cualquier resultado, RESULTANTES DISEÑOS, hardware, productos de software o cualquier otra aplicación, RESULTANTE DE USO DEL SOFTWARE TEMA. ADEMÁS, LA AGENCIA DEL GOBIERNO RECHAZA TODAS LAS GARANTÍAS Y RESPONSABILIDAD SOBRE SOFTWARE DE TERCEROS, si está presente en el software original, y la distribuye "TAL CUAL".

Renuncia y B. Indemnización: receptor se compromete a RENUNCIO Y CUALQUIER RECLAMO CONTRA EL GOBIERNO DE LOS ESTADOS UNIDOS, sus contratistas y subcontratistas, así como cualquier receptor antes. SI USO RECEPTOR de los resultados SOFTWARE TEMA EN cualquier responsabilidad, demanda, daños, gastos o pérdidas derivados de su uso, incluyendo cualquier daño DE PRODUCTOS BASADOS EN, o resultantes de USO destinatario DEL SOFTWARE TEMA, BENEFICIARIO deberá indemnizar y mantener indemne a los gobierno de Estados Unidos, sus contratistas y subcontratistas, así como cualquier receptor antes, en la medida PERMITIDO POR LA LEY. El único recurso del destinatario Para cualquier asunto que deberá ser el despido inmediato, UNILATERALES DE ESTE ACUERDO.

5. CONDICIONES GENERALES

A. Terminación: El presente Acuerdo y de los derechos aquí concedidos finalizarán automáticamente en caso de un beneficiario no cumpla con estos

términos y condiciones, y no ponga remedio a dicho incumplimiento dentro de los treinta (30) días de haber tenido conocimiento de dicho incumplimiento. Sobre la terminación, un destinatario se compromete a cesar inmediatamente el uso y distribución del Software Asunto. Todas las sublicencias del Software Asunto legalmente otorgada por el Beneficiario no desaparecen con la violación de dicha terminación del presente Acuerdo.

B. Divisibilidad: Si alguna disposición de este Acuerdo es inválida o no exigible bajo la ley aplicable, no afectará a la validez o aplicabilidad del resto de los términos del presente Acuerdo.

C. Ley aplicable: El presente Acuerdo estará sujeto a los Estados Unidos la ley federal sólo para todos los efectos, incluyendo, pero no limitado a, la determinación de la validez del presente Acuerdo, el significado de sus disposiciones y los derechos, obligaciones y recursos de las partes.

D. Todo el Entendimiento: Este Acuerdo constituye el completo entendimiento y acuerdo de las partes relativos a la entrega del Software sujeto y no puede ser sustituida, modificada o modificado con el consentimiento de otros escritos debidamente cumplimentada por las partes.

E. Autoridad Encuadernación: Al aceptar y utilizar el Software Asunto virtud del presente Acuerdo, un destinatario afirma su autoridad para obligar a la del destinatario de todos los términos y condiciones del presente Acuerdo, y que cuando el Destinatario se compromete a todos los términos y condiciones.

F. Punto de Contacto: Cualquier contacto del destinatario con el Gobierno de la Agencia debe ser dirigida al representante designado de la siguiente manera: Patrick.Hogan @ nasa.gov

6.2 – Manual de usuario

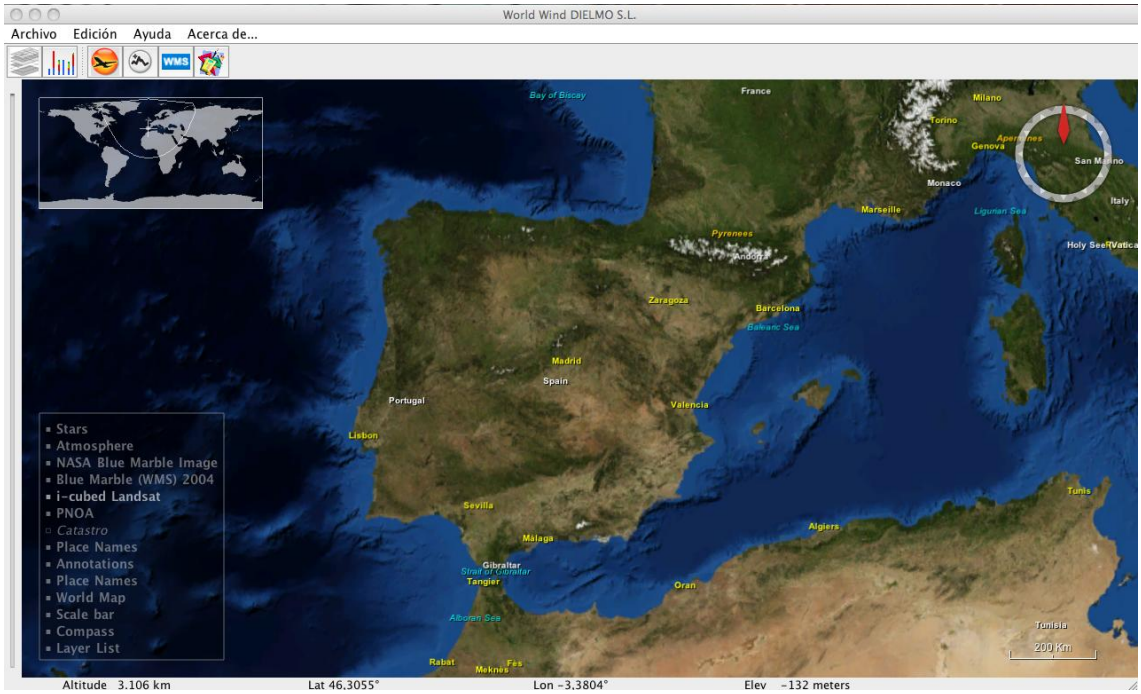

Ilustración 32 – Imagen de la interfaz principal

La interfaz de usuario es muy sencilla, el ella podemos ver una barra de herramientas con las siguientes opciones, *Archivo Edición Ayuda y Acerca de…* además podemos observar una serie de botones mediante los cuales podemos acceder a las distintas opciones disponibles.

También podemos observar una ventana semi transparente la cual nos representa las diferentes capas que estamos visualizando en el momento y si estas se encuentran activadas o desactivadas.

Ilustración 33 – menú de capas en la aplicación

Cabe destacar que la propia ventana, es en sí misma una capa y en cualquier momento podemos habilitarla o deshabilitarla.

6.2.1 – Opciones de la aplicación

Manejo con Ratón

Las opciones con raton son las que siguen:

Acercar imagen: Para acercar la imagen podemos utilizar el Scroll del ratón "la ruedecita". De modo que rodándola hacia delante acercamos la imagen y rodándola hacia detrás la alejamos. Esta función también está disponible presionando el botón del scroll y desplazando el ratón hacia delante o atrás.

Ajustar inclinación y giro: Por defecto la aplicación nos muestra una vista perpendicular de la zona que estamos viendo y una orientación norte. Para cambiar esto simplemente hemos de presionar el botón derecho del ratón y desplazándonos hacia arriba/abajo ajustar la inclinación mientras que desplazando a derecha o izquierda podemos ajustar la orientación o giro, para obtener una orientación sur, sureste, etc.

Desplazar el mapa: para movernos por el mapa sin ningún efecto sobre la inclinación, distancia u otros parámetros, simplemente podemos pulsar con el botón izquierdo del ratón y sin dejar de presionar, mover el ratón a nuestro antojo para movernos libremente.

Barra de herramientas

Los distintos botones, representan las funciones más habituales que podemos llevar a cabo en nuestro aplicativo, por tanto pasaremos a describirlos.

Ilustración 34 – Barra de herramientas de la aplicación

El primero de ellos (de izquierda a derecha), nos mostrará del mismo modo que la ventana del aplicativo, las distintas capas que tenemos habilitadas en ese momento.

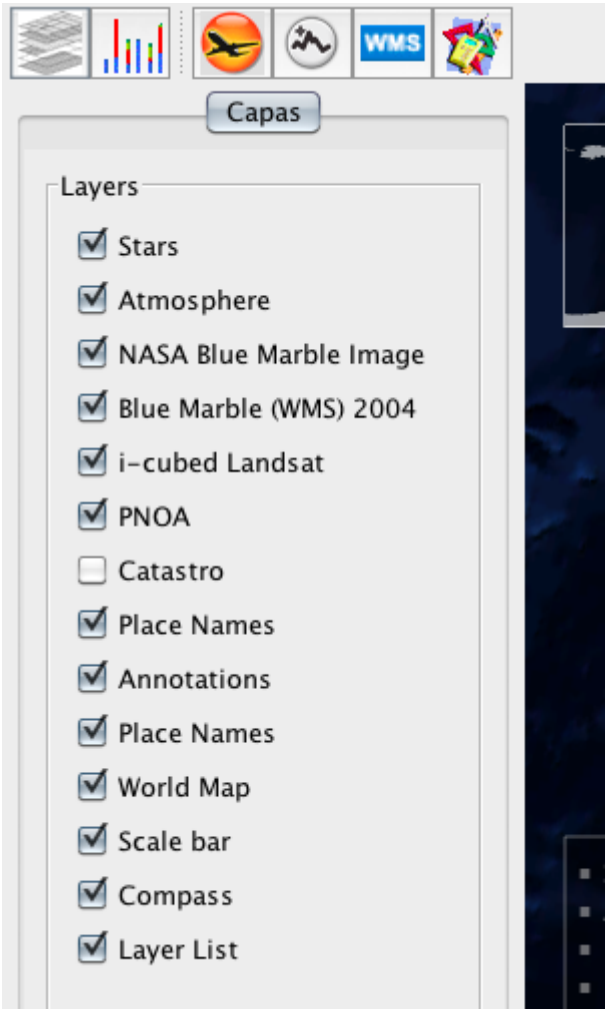

Ilustración 35 – Menú de capas

El segundo de ellos simplemente nos mostrará las estadísticas de utilización de la JVM en nuestro PC/MAC.

El tercero, representado por un globo que contiene un avión, nos mostrará un cuadro de diálogo para viajar a algún punto del globo terráqueo, en él podremos indicar, tanto el nombre del país, población, provincia… como sus coordenadas en formato UTF.

Ilustración 36 – Menú diálogo para volar hacia unas coordenadas

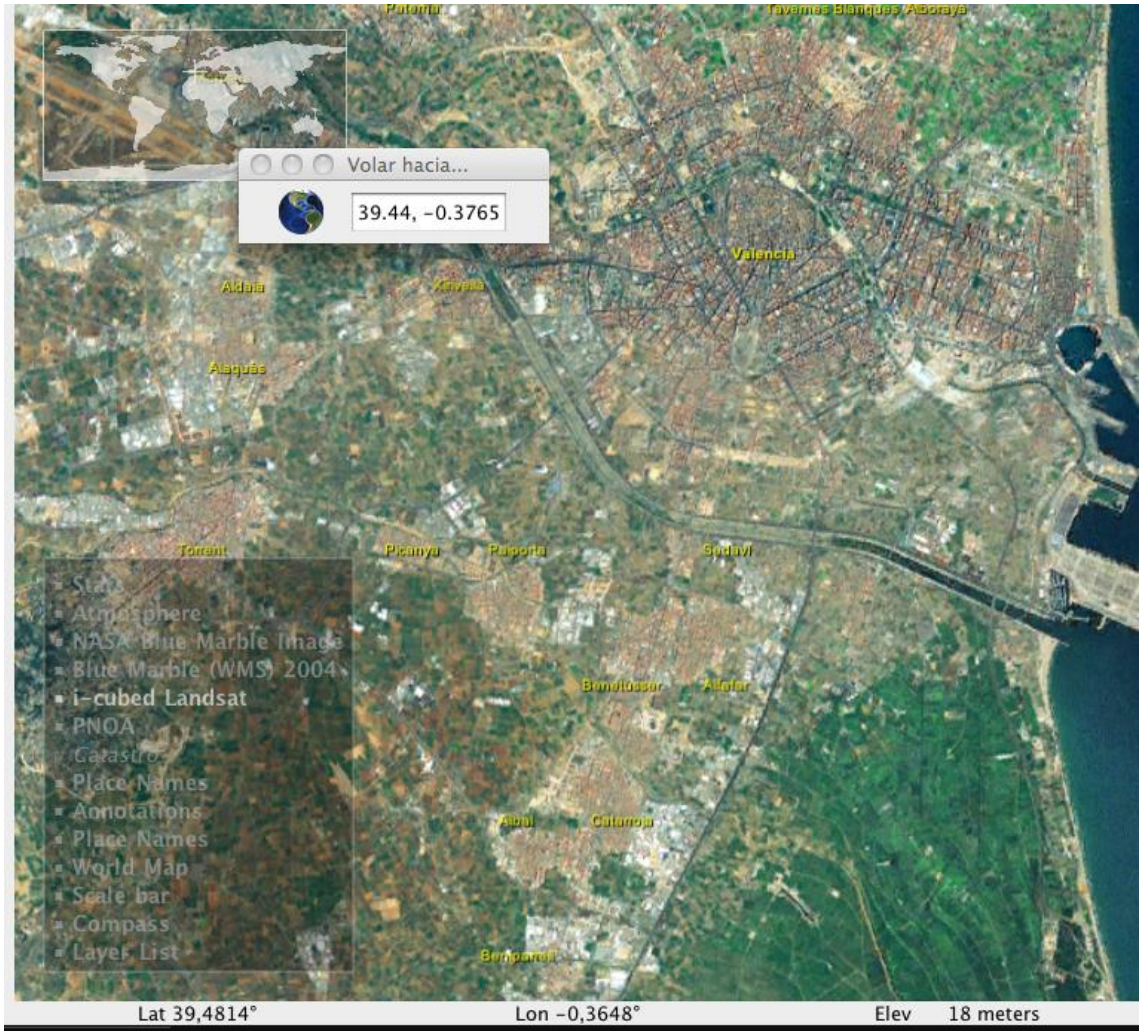

Ilustración 37 – Resultado de volar hacia un punto

Autor: José Vicente Mora Espí
Escuela técnica superior de ingeniería informática

Además en todo momento podremos ver la Latitud y Longitud en la cuál nos encontramos, simplemente echando un vistazo a la barra inferior de la aplicación.

Con el cuarto botón, podemos controlar la exageración vertical del modelo 3D, haciendo más agudos los vértices y observando con mucha mayor claridad pequeños desniveles.

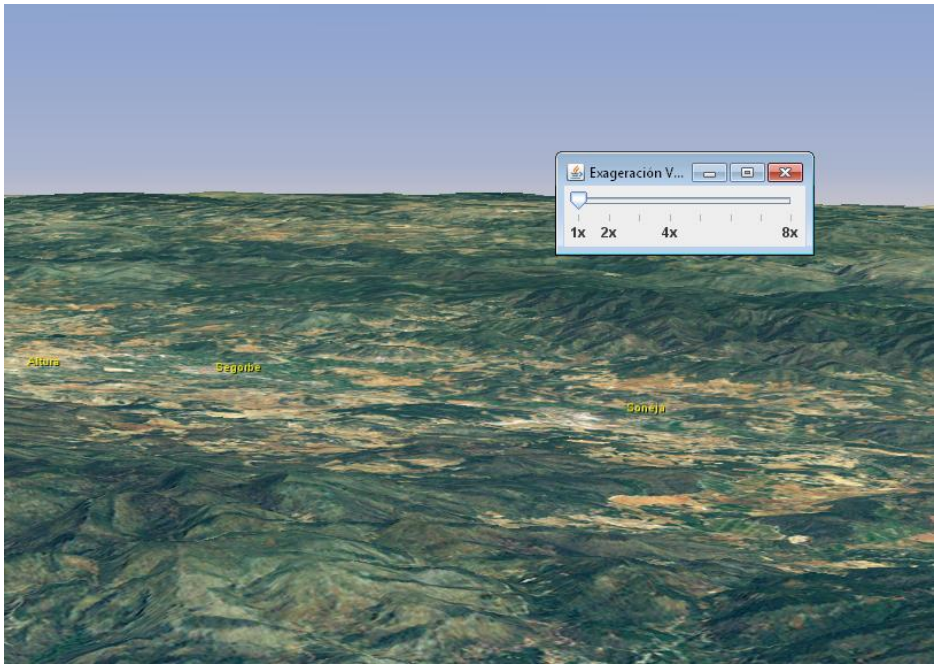

Ilustración 38 – Exageración vertical normal

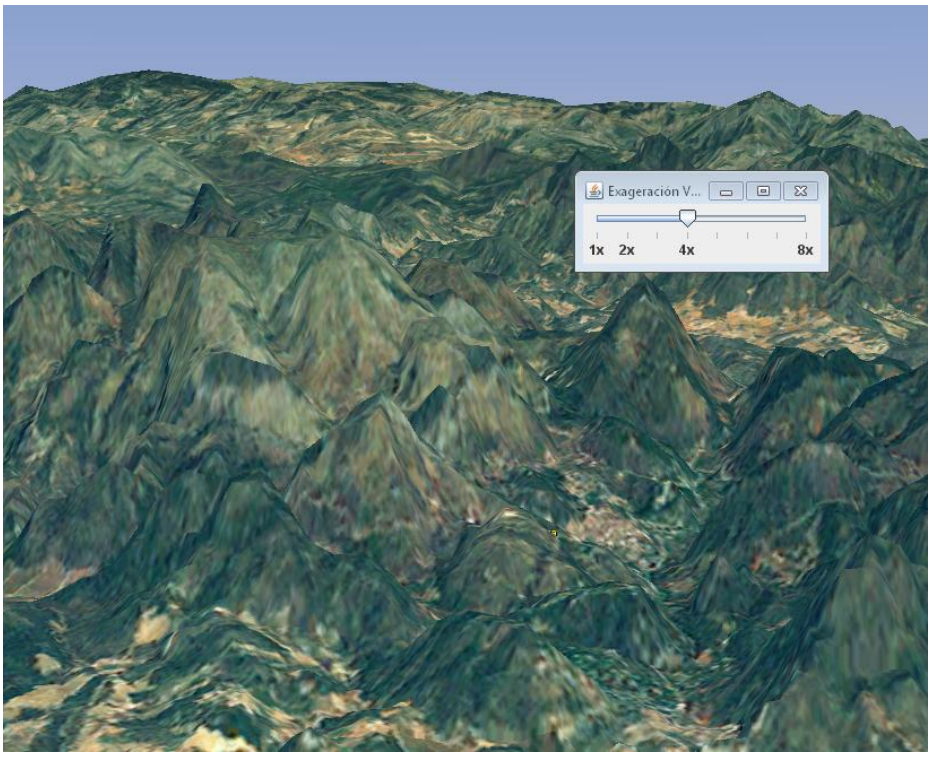

Ilustración 39 – Exageración vertical aumentada hasta 4x

El quinto botón, indicado por WMS nos permitirá añadir direcciones URL de servidores WMS para poder consultar mediante este servicio los mapas que nos ofrezcan. Al pulsar sobre él , nos aparecerá un nuevo cuadro de diálogo en el cual veremos un "+" con el cuál podremos añadir nuevos servidores.

<

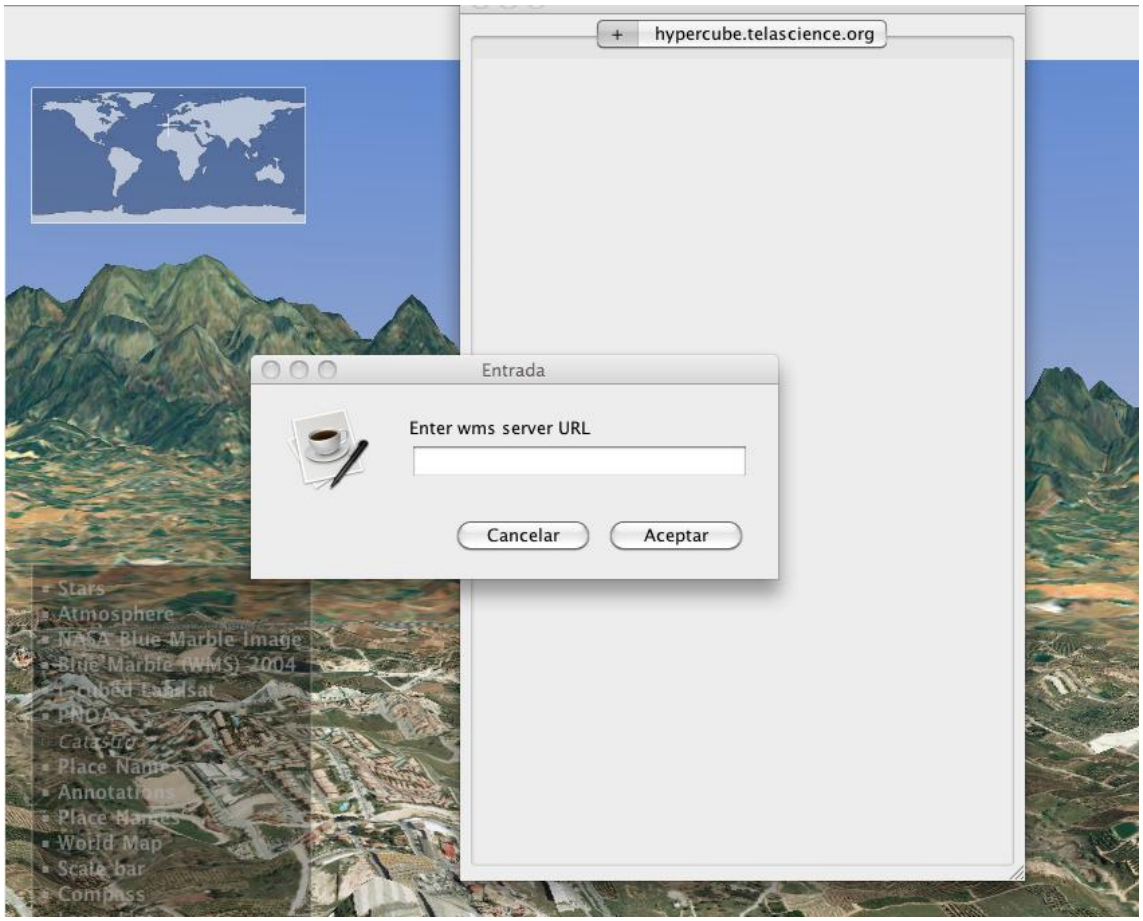

Ilustración 40 – Menú para añadir url's de servidores WMS

Y por último el botón situado más a la derecha nos abrirá la utilidad para realizar mediciones sobre las superficie 3D.

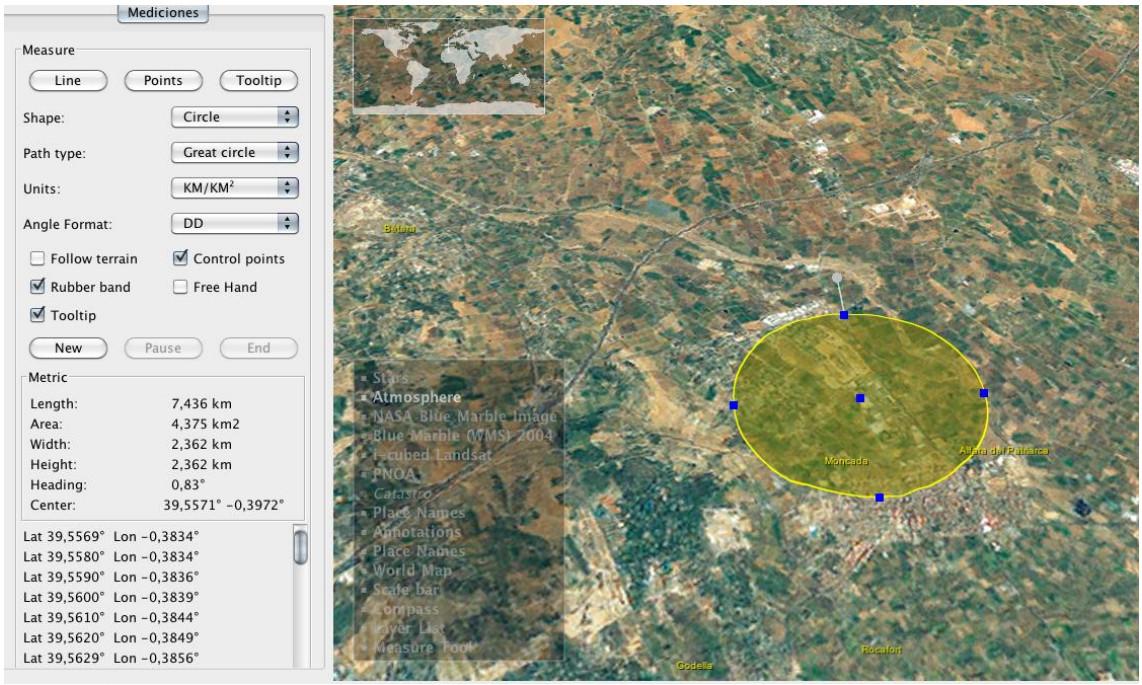

Ilustración 41 – Medición del área de una circunferencia sobre una zona

Barra de menús

Terminado con los botones, pasamos a describir las opciones que hay en la barra de menús.

En Archivo encontramos las opciones para añadir una imagen o un MDT a voluntad, las imágenes pueden ser en formato *JPG, PNG, BMP, GIF y JPEG*. Siendo la opción más recomendable los de tipo PNG. En cuanto a los MDT únicamente se aceptan MDT's en formato BIL.

Ilustración 42 – Inserción de una imagen PNG en una zona del globo donde se desarrolla ciclismo

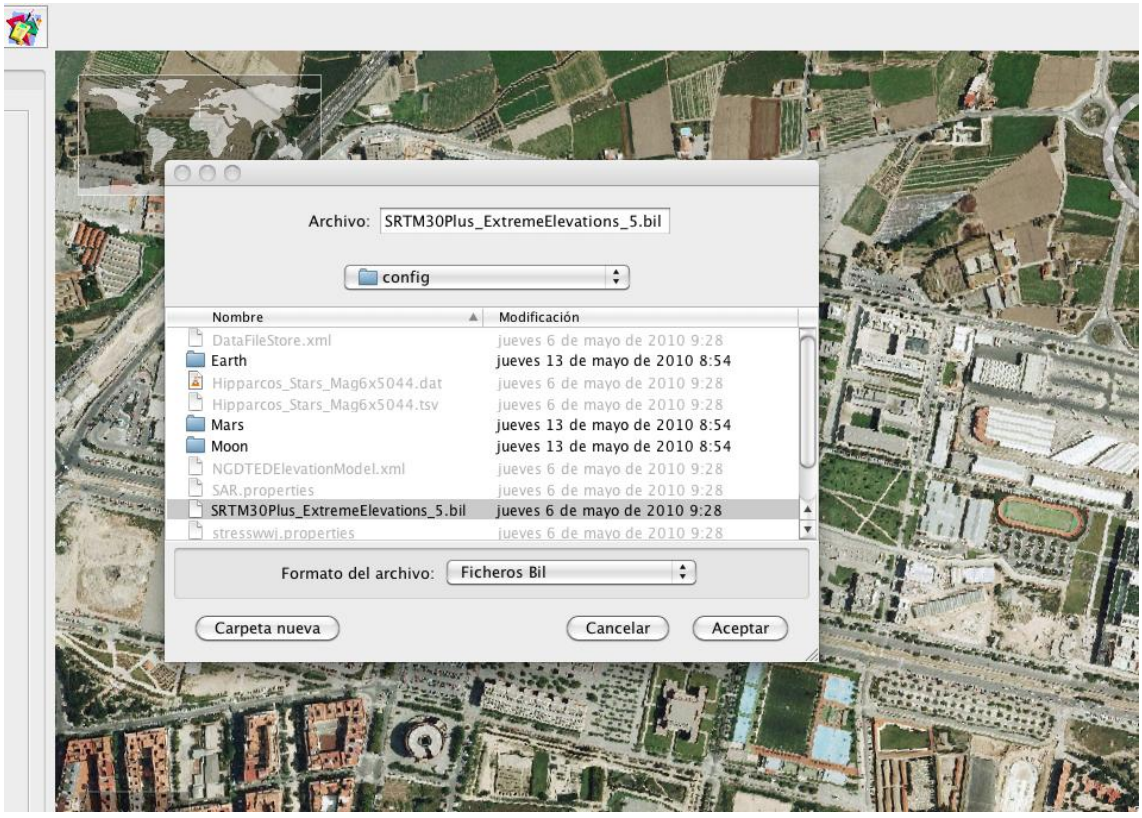

Ilustración 43 – Menú para seleccionar un MDT en formato BIL

Autor: José Vicente Mora Espí
Escuela técnica superior de ingeniería informática

GLOSARIO DE TÉRMINOS

CAD – El diseño asistido por computadora, más conocido por sus siglas inglesas CAD (computer-aided design), es el uso de un amplio rango de herramientas computacionales que asisten a ingenieros, arquitectos y a otros profesionales del diseño en sus respectivas actividades. El CAD es también utilizado en el marco de procesos de administración del ciclo de vida de productos (en inglés product lifecycle management).

SIG – Sistema de información geográfica. Es una integración organizada de hardware, software y datos geográficos diseñado con el fin de resolver problemas complejos de planificación y gestión geográfica.

3D – Tridimensional. Comprende las dimensiones ancho, largo y profundo, creando una visión cercana a la realidad.

API – Interfaz de programación de aplicaciones, es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

SRTM – Shuttle Radar Topography Mission. Es una mission para obtener un modelo digital de elevación del globo terráqueo.

JPL – Jet propulsion laboratory. Es un centro de investigación y desarrollo financiado federalmente por EEUU.

OGC – Open Geospatial Consortium (OGC) fue creado en 1994 y agrupa (en febrero de 2009) a 372 organizaciones públicas y privadas. Las raíces del OGC se encuentran en la software fuente libre GRASS y la subsiguiente fundación OGF (Open GIS Foundation) fundada en 1992. Su fin es la definición de estándares abiertos e interoperables dentro de los Sistemas de Información Geográfica y de la World Wide Web.

WFS – Web Feature Service es un servicio estándar, que ofrece una interfaz de comunicación que permite interactuar con los mapas servidos por el estándar WMS, como por ejemplo, editar la imagen que nos ofrece el servicio WMS o analizar la imagen siguiendo criterios geográficos.

WMS – Web Map Service es un servicio que produce mapas de datos referenciados espacialmente, de forma dinámica a partir de información geográfica. Este estándar internacional define un "mapa" como una representación de la información geográfica en forma de un archivo de imagen digital conveniente para la exhibición en una pantalla de ordenador. Un mapa no consiste en los propios datos. Los mapas producidos por WMS se generan normalmente en un formato de imagen como PNG, GIF o JPEG, y opcionalmente como gráficos vectoriales en formato SVG (Scalable Vector Graphics) o WebCGM (Web Computer Graphics Metafile).

GEOTIFF – es un estándar de metadatos de domino público que permite que información georreferenciada sea encajada en un archivo de imagen de formato TIFF.

RAW – El formato de imágenes RAW ("crudo" en inglés; en el caso de las imágenes, entiéndase como "Formato de Imagen sin modificaciones") es un formato de archivo digital de imágenes que contiene la totalidad de los datos de la imagen tal y como ha sido captada por el sensor digital de la cámara fotográfica.

PNOA – Plan Nacional de Ortofotografía Aérea es un proyecto que tiene como objetivo la obtención de ortofotografías aéreas digitales con resolución de 25 ó 50 cm y modelos digitales de elevaciones (MDE) de alta precisión de todo el territorio español, con un período de actualización de 2 ó 3 años, según las zonas. Se trata de un proyecto cooperativo y cofinanciado entre la Administración General del Estado y las Comunidades Autónomas.

GML – acrónimo inglés de Geography Markup Language (Lenguaje de Marcado Geográfico). Es un sublenguaje de XML descrito como una gramática en XML

Schema para el modelaje, transporte y almacenamiento de información geográfica. Su importancia radica en que a nivel informático se constituye como unalengua franca para el manejo y trasvase de información entre los diferentes software que hacen uso de este tipo de datos, como los Sistemas de Información Geográfica.

KML – Keyhole Markup Language, es un lenguaje de marcado basado en XML para representar datos geográficos en tres dimensiones. Fue desarrollado para ser manejado con Keyhole LT, precursor de Google Earth (Google adquirió Keyhole LT en octubre de 2004 tras lanzar su versión LT 2). Sugramática contiene muchas similitudes con la de GML.

COLLADA – COLLADA son las siglas de COLLaborative Design Activity, para establecer un tipo de formato de fichero enfocado al intercambio de información de aplicaciones 3D. COLLADA está administrado por el consorcio Khronos Group, organizacion sin ánimo de lucro.

IDEE – Infraestructura de Datos Espaciales de España, es una iniciativa para integrar datos, metadatos e información geográfica producida en España en Internet, la cual provee localización, identificación y acceso a esa información a los potenciales usuarios.

JOGL – Java OpenGL (JOGL) es una biblioteca que permite acceder a OpenGL mediante programación en Java. Actualmente está siendo desarrollado por el Game Technology Group de Sun Microsystems, y es la implementación de referencia para JSR-231 (Java Bindigs for OpenGL).

HTML – HTML, siglas de HyperText Markup Language («lenguaje de marcado de hipertexto»), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. El HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un

documento, y puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

POI – Point Of Interest, punto de interés, en geografía se utiliza para marcar localizaciones relevantes o de interés artístico, histórico, lúdico, etc.

ICC – Instituto Geografico de Cataluña.

LANDSAT – son una serie de satélites construidos y puestos en órbita por EE. UU. para la observación en alta resolución de la superficie terrestre.

GPX – GPS eXchange Format (Formato de Intercambio GPS) es un esquema XML pensado para transferir datos GPS entre aplicaciones. Se puede usar para describir puntos (waypoints), recorridos (tracks), y rutas (routes).

KDE – es un proyecto de software libre para la creación de un entorno de escritorio e infraestructura de desarrollo para diversos sistemas operativos como GNU/Linux, Mac OS X, Windows, etc.

JVM – es un máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el bytecode Java), el cual es generado por el compilador del lenguaje Java.

NASA – son las siglas, en Idioma inglés, para la Administración Nacional de Aeronáutica y del Espacio (National Aeronautics and Space Administration) de los Estados Unidos, que es la agencia gubernamental responsable de los programas espaciales.

PHP – es un lenguaje de programación interpretado (Lenguaje de alto rendimiento), diseñado originalmente para la creación de páginas web dinámicas. Se usa principalmente para la interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de

comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

QT – es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con una interfaz gráfica de usuario así como también para el desarrollo de programas sin interfaz gráfica como herramientas para la línea de comandos y consolas para servidores.

GTK+ – es un conjunto de bibliotecas multiplataforma para desarrollar interfaces gráficas de usuario (GUI), principalmente para los entornos gráficos GNOME, XFCE y ROX aunque también se puede usar en el escritorio de Windows, MacOS y otros.

PNG – es un formato gráfico basado en un algoritmo de compresión sin pérdida para bitmaps no sujeto apatentes. Este formato fue desarrollado en buena parte para solventar las deficiencias del formato GIF y permite almacenar imágenes con una mayor profundidad de contraste y otros importantes datos.

GIF – es un formato gráfico utilizado ampliamente en la World Wide Web, tanto para imágenes como para animaciones.

JPG/JPEG – Joint Photographic Experts Group, es el nombre de un comité de expertos que creó un estándar de compresión y codificación de archivos de imágenes fijas. Este comité fue integrado desde sus inicios por la fusión de varias agrupaciones en un intento de compartir y desarrollar su experiencia en la digitalización de imágenes. La ISO, tres años antes (abril de 1983), había iniciado sus investigaciones en el área. Además de ser un método de compresión, es a menudo considerado como un formato de archivo. JPEG/Exif es el formato de imagen más común utilizado por las cámaras fotográficas digitales y otros dispositivos de captura de imagen, junto con JPG/JFIF, que también es otro formato para el almacenamiento y la transmisión de imágenes fotográficas en la World Wide Web. Estas variaciones de formatos a menudo no se distinguen, y se llaman JPEG. Los archivos de este tipo se suelen nombrar con la extensión .jpg.

URL – URL (sigla en inglés de uniform resource locator), es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, vídeos, presentaciones digitales, etc.

SVG – Gráficos Vectoriales Escalables (del inglés Scalable Vector Graphics) o SVG es una especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados (estos últimos con ayuda de SMIL), en formato XML.

XML – eXtensible Markup Language ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades, de ahí que se le denomine metalenguaje. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

MDT – modelo digital de terreno (MDT) es una representación de la topografía (altimetría y/o batimetría) de una zona terrestre (o de un planeta telúrico) en una forma adaptada a su utilización mediante un ordenador digital (ordenador).

CSV – Los ficheros CSV (del inglés comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador decimal: España, Francia, Italia...) y las filas por saltos de línea. Los campos que contengan una coma, un salto de línea o una comilla doble deben ser encerrados entre comillas dobles.