

Manual de Capaware

Grupo desarrollador de Capaware

Marzo 2009

Índice general

ÍNDICE GENERAL

Prefacio

Este documento pretende ser una introducción al software Capaware, el cual brevemente puede definirse como una herramienta libre que permite la visualización realista y en 3D de terrenos, compatible con capas OGC (Open Geospatial Consortium), dotado además de un sistema de plugins para añadir nuevas funcionalidades. En este manual se cubren los aspectos básicos del software, una guía para su instalación, un manual de uso de la aplicación principal, y una guía de usuario para desarrolladores.

Como todo software libre, uno de sus objetivos principales consiste en permitir que cualquier usuario con conocimientos suficientes de programación pueda desarrollar nuevos módulos, mejorando así la calidad de toda la herramienta. Para ello se ha creado la página web oficial <http://www.capaware.org>, en donde se ubicará el código fuente y la documentación que se genere.

A quien va dirigido este manual

Este manual está orientado por un lado a los usuarios habituales de SIG (Sistemas de Información Geográfica) e IDE (Infraestructura de Datos Espaciales), que quieran usar una herramienta de visualización 3D para manejar la información geográfica. Por otro lado está dirigido a los desarrolladores que quieran construir aplicaciones usando la API de Capaware (CPW), y también a aquellos que quieran añadir nuevos plugins al sistema para mostrar nuevas funcionalidades.

Organización del documento

El documento se divide en dos partes claramente diferenciadas. La primera está orientada al usuario general, en donde se incluye una visión general de Capaware, junto a una guía de instalación y el manual de usuario de las aplicaciones incluidas que les permitirán visualizar un terreno, mostrar capas OGC sobre él, y crear y gestionar elementos 3D insertados en la escena.

La segunda parte está dedicada a los usuarios desarrolladores, y en ella se describen inicialmente las librerías sobre las que Capaware se apoya (OpenSceneGraph [?], libcurl y wxWidgets [?]), para posteriormente introducir una guía de programación utilizando la API desarrollada, partiendo del código de las aplicaciones incluidas, incluyendo el desarrollo de *plugins*.

Sobre los autores

Este software no hubiera sido posible sin un equipo interdisciplinar de desarrollo formado por analistas y programadores, provenientes de la Universidad de Las Palmas de Gran Canaria (ULPGC), y del Instituto Tecnológico de Canarias (ITC), que han colaborado durante 18 meses en el desarrollo del proyecto. A continuación se detallan sus nombres y la entidad a la que pertenecen:

- Agustín Trujillo Pino (ULPGC)
- Izzat Sabbagh Rodríguez (ITC)
- Javier Sánchez Pérez (ULPGC)
- Modesto Castrillón Santana (ULPGC)
- Jose Pablo Suárez Rivero (ULPGC)
- Ignacio José López Rodríguez (ITC)
- Rafael J. Nebot Medina (ITC)
- David Martín Zerpa
- Pedro Jorge González
- Antonio José Sánchez López
- Francisco Manuel Quintana Trujillo (ITC)

Agradecimientos

Agradecemos a los técnicos de la Consejería de Medio Ambiente del Cabildo Insular de la isla de La Palma, por su colaboración en el desarrollo del subsistema de simulación predictiva de incendios forestales, y por prestar las instalaciones de su Centro de Coordinación Insular de Emergencias para la puesta a punto de la aplicación y su prueba en un entorno real. Así mismo, no hubiésemos podido iniciar este proyecto sin la colaboración del Centro de Coordinación de Emergencias del Gobierno de Canarias CECOES-112, así como el apoyo de la empresa pública del Gobierno de Canarias GRAFCAN, Cartográfica de Canarias.

Capítulo 1

Visión general

En este capítulo se ofrece una introducción al software Capaware y sus distintos módulos.

1.1. Historia del proyecto

A finales del año 2004, el Departamento de Ingeniería del Software del Instituto Tecnológico de Canarias (ITC) inicia una línea de trabajo para la creación de proyectos de entornos virtuales 3D para Canarias. Para ello, contacta con investigadores del Departamento de Informática y Sistemas (DIS) de la Universidad de Las Palmas de Gran Canaria (ULPGC) y establece un grupo conjunto de trabajo en este campo. Como primera experiencia se plantea la creación de un navegador de terrenos virtual en una zona concreta de las Islas Canarias, en la isla de El Hierro. Dicho simulador debía permitir una experiencia de vuelo real, con el fin de servir de presentación de la isla con distintos propósitos, tanto turísticos como industriales (la figura ?? muestra una captura de pantalla). Los objetivos que se plantearon al comienzo del desarrollo, y que deberían permitir considerar la aplicación como diferenciadora de otras aportaciones similares, eran los siguientes:

1. Elección de una plataforma de desarrollo potente, flexible y de licencia pública. En este sentido hay que señalar que existen plataformas de desarrollo para realizar vuelos virtuales con costes muy elevados que son prohibitivos de amortizar. En nuestro caso, el motor Crystal Space¹ fue la elección.
2. Suavidad del movimiento. En el vuelo, el usuario debe experimentar la sensación mas suave en el movimiento libre sobre la isla, evitando así, saltos o discontinuidades en el vuelo.
3. Realismo y fidelidad de las ortofotos. Se debe disponer de las mejores ortofotos digitales y convenientemente mapeadas en el sistema para lograr la sensación mas real del vuelo sobre la isla.

Todos los objetivos se cumplieron satisfactoriamente. Como curiosidad cabe destacar que el popular navegador Google Earth², software que incorporaba algunas de las características ya planteadas por nuestro grupo de trabajo, aunque con mucho menor rendimiento al tratarse de una aplicación web, no había aparecido todavía (lo hizo en Julio de 2005).

Tras el éxito obtenido en el visualizador anterior, y aprovechando la experiencia de programación gráfica realizada, se decidió desarrollar un proyecto más ambicioso, que abarcara la totalidad de la

¹<http://www.crytalspace3d.org/main/Main>

²<http://earth.google.es/>



Figura 1.1: Captura de pantalla de El Hierro Virtual

superficie del archipiélago canario, ofreciendo además una utilidad concreta. Como caso de uso inicial nos planteamos la capacidad de realizar simulaciones predictivas de incendios forestales para su uso en un sistema de emergencias, aprovechando los resultados de otro proyecto del ITC de predicción meteorológica en el campo de la eficacia energética de plantas de producción solar fotovoltaica.

Los objetivos que se plantearon inicialmente fueron los siguientes:

- Desarrollar un motor que permitiera la visualización realista de terrenos.
- Poder insertar elementos 3D sobre el terreno, pudiendo ser animados en el tiempo.
- Capaz de conectarse a servidores estándar OGC y mostrar sus capas de información sobre dicho terreno.
- Permitir varios usuarios conectados simultáneamente en tiempo real, de forma que las modificaciones de uno sean vistas inmediatamente por el resto.
- Generar una API libre para que puedan desarrollarse nuevas aplicaciones sobre ella.
- Permitir la creación de plugins para añadir nuevas funcionalidades de forma sencilla.

1.2. Motor gráfico utilizado: OpenSceneGraph

La primera decisión que se tomó fue la de elegir un motor gráfico sobre el que construir Capaware. El motor usado para el proyecto Cavacan (Crystal Space) se quedaba corto, puesto que la extensión de

terreno que queríamos utilizar no podía cargarse totalmente en memoria. Así que se analizaron distintos motores gráficos que existían en ese momento. Las características que se buscaban eran:

- Software libre
- Multiplataforma
- Sistema de cacheado de la escena, para no mantenerla de forma completa en memoria

Los motores seleccionados con las mejores características fueron el Open Scene Graph (OSG) y el Virtual Terrain Project (VTP). A pesar que éste último estaba orientado al uso más específico de visualización de terrenos, finalmente nos decantamos por el OSG, puesto que nos interesaba su uso más generalista. Además, en las pruebas iniciales que hicimos con terrenos extensos, se comportó mucho mejor que su rival en cuanto a tiempo de generación de la escena, y rendimiento durante la visualización.

1.3. Diseño del Capaware

El núcleo de Capaware es en realidad una API multiplataforma, llamada CPW, la cual puede utilizarse para el desarrollo de aplicaciones de visualización de terrenos. Este núcleo se construye sobre el motor gráfico elegido, OSG, el cual está soportado por un lado por la librería gráfica multiplataforma OpenGL, y por otro lado, por una librería de interfaces de usuario, que en nuestro caso puede ser cualquiera, aunque en la implementación final se ha elegido utilizar wxWidgets.

El diagrama final de bloques para el diseño de Capaware se muestra en la figura ??.

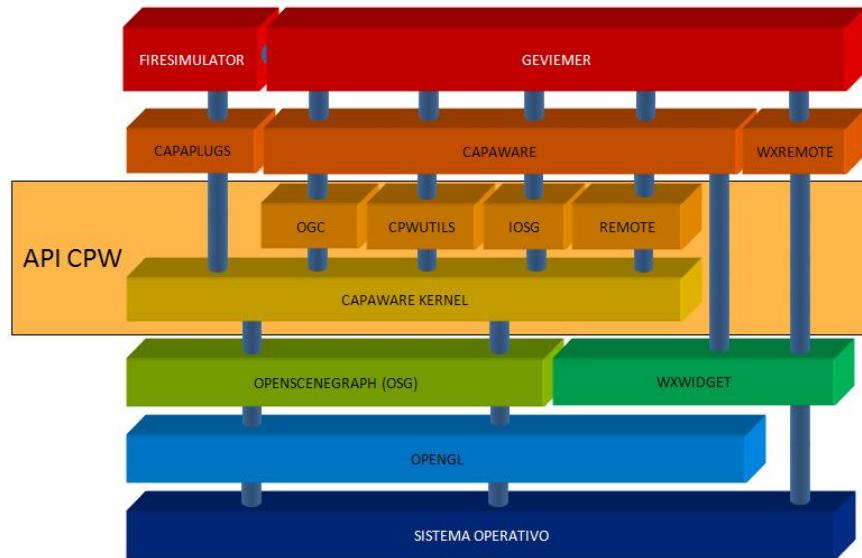


Figura 1.2: Arquitectura en capas de Capaware

La API CPW, está compuesta por un módulo base, llamado Capaware Kernel, que incluye toda la funcionalidad básica, y otros módulos adicionales colocados por encima de éste, encargados de gestionar las conexiones remotas, los protocolos OGC, utilidades varias, etc.

Por encima de la API CPW existe la aplicación general Capaware, que es la que integra la llamada a la mayoría de la funcionalidad de la API. El objetivo de esta nueva aplicación es servir tanto para usuarios desarrolladores de contenido como para usuarios que desarrollen plugins como soporte para añadir nuevas funcionalidades. Por otro lado, también pretende servir como ejemplo de partida para los desarrolladores de aplicaciones.

En la distribución también se ha añadido una aplicación más ligera, llamada CapaViewer, que simplemente permite cargar una escena ya generada, y poder navegar por ella. No permite crear nuevos contenidos, sino solamente visualizarlos. Su objetivo es simplemente tener un cliente para que un usuario general pueda visualizar una escena previamente generada, la cual puede componerse de un terreno con ortofotos que se hallen en disco local, junto con elementos 3D y capas OGC que puedan estar tanto en local como en servidores remotos.

Finalmente, en lo más alto del diagrama de la figura ?? existe una aplicación más elaborada, ya con una aplicación específica y con un plugin concreto de simulación de incendios añadido, llamado Geviemer (Gestor Virtual de Emergencias), orientado a la gestión de incendios forestales, desarrollado a partir de Capaware, y que está actualmente funcionando en fase de pruebas en las dependencias del Centro Coordinador Insular de Emergencias del Cabildo de La Palma.

1.4. Requerimientos del sistema

Aunque Capaware se ha diseñado para ser multiplataforma, la presente versión sólo ha sido probada en Windows, tanto XP como Vista. Si se quiere desarrollar nuevos módulos sobre Capaware, se deberá además tener instalada una versión de Microsoft Visual Studio (2005 o posterior). Por otro lado, se puede utilizar CMAKE para generar el proyecto para otros entornos de desarrollo. Actualmente, los archivos CMAKELIST.txt de cada subproyecto permiten generar correctamente el proyecto en Microsoft Visual Studio 2005, pero no se ha probado para otros IDE como el Eclipse, aunque lo tenemos en nuestra 'hoja de ruta'.

En cuanto a requisitos hardware, la única condición es que el sistema disponga de una tarjeta 3D con al menos 256Mb de memoria, y compatible con OpenGL (mucho mejor NVidia que ATI, la cual tiene ciertos problemas de compatibilidad).

También es aconsejable tener bastante memoria RAM y una gran capacidad de disco duro para albergar las escenas.

1.5. Licencia de Capaware

Capaware ha sido desarrollado totalmente con fondos públicos del Gobierno de Canarias en su intención de potenciar y fomentar el uso y desarrollo de aplicaciones de software libre. Además, desde el grupo de desarrollo queremos que la plataforma pueda enriquecerse por medio de las aportaciones realizadas por la comunidad que pueda crearse a su alrededor, de forma libre y sin temor a que otro tipo de licencia pueda conducir a módulos estancos privativos que no puedan ser aprovechados por todos. Así pues, dentro de esta filosofía de trabajo por parte del Gobierno de Canarias, y través de la colaboración ITC-ULPGC, la licencia de un producto como este no podía ser otra que la GPL v3. Ver licencia en <http://www.gnu.org/copyleft/gpl.html>.

Capítulo 2

Primeros pasos con Capaware

En este capítulo se describe la forma de instalar el software, un manual para la herramienta de generación de terrenos, y cómo poder sobrevolarla usando Capaware.

2.1. Instalación de la aplicación

En la web oficial <http://www.capaware.org> se mantendrán actualizadas las diferentes versiones de Capaware. El presente manual fue creado con la primera de ellas, versión 1.0.0.

Tras descargar y ejecutar el fichero de instalación, un asistente nos guiará en el proceso de instalación del software. En esta versión, todas las dependencias requeridas por la aplicación van incluidas en el instalador, incluyendo la versión de OSG que se utilizó.

Tras realizar la instalación, un acceso directo sobre el escritorio permite acceder a la aplicación Capaware. Su ejecución presenta las barras de menú y herramientas, pero no habrá ninguna escena en pantalla. Esto es debido a que la instalación no incluye una escena, es decir datos geográficos para su visualización, por lo que es necesario generar un terreno, o escena en disco antes de poder volar sobre ella.

2.2. Generación de un terreno

Como hemos comentado, antes de poder ejecutar el visualizador realizando un vuelo virtual sobre un terreno, necesitamos generar dicho terreno o escena. Aunque muchos visualizadores actuales no necesitan ninguna escena local en disco, sino que al ejecutarse traen de la red todo lo necesario, el objetivo inicial de Capaware era tener una copia local del terreno a visualizar. El inconveniente de hacerlo así es que se necesita forzosamente disponer de una escena ya generada para poder empezar a trabajar. Sin embargo la gran ventaja es que podemos tener un gran nivel de detalle, tanto en la resolución de las ortofotos como en el grado de definición de la modelo digital del terreno, y su rendimiento en tiempo real será mucho más eficiente que realizando conexiones remotas para su descarga.

2.2.1. Uso de la aplicación CapaBuilder

Para generar la escena utilizaremos la herramienta CapaBuilder, que viene incluida en la distribución. Esta herramienta conectará con el servidor WMS y WCS que indique el usuario, en donde se encuentren las ortofotos y los mapas de altura respectivamente, y la herramienta descargará toda la información

necesaria y generará un fichero con extensión ".ive" que es el que se abrirá desde Capaware. La forma de hacerlo es la siguiente:

1. Una vez que se haya lanzado la aplicación CapaBuilder, en el menú *Build Scene* seleccionamos la opción *Build Scene*.
2. Nos aparece una ventana, ver figura ??, para seleccionar el servidor *WMS* al que queremos conectarnos para descargar las ortofotos.
 - a) En el apartado **Wms Server** indicamos la url del servidor al que queremos conectarnos.
 - b) Si la conexión se ha establecido correctamente en el campo **Description** aparecerá el mensaje de bienvenida del servidor.
 - c) El campo **Select layer** se llenará con las capas disponibles en el servidor.
 - d) Al seleccionar alguna capa, los campos **Format** y **SRS** se llenarán con los formatos de imagen y sistemas de proyección soportados por esa capa y deberemos seleccionar el que nos interese, aunque como veremos a continuación esos datos pueden cambiarse más adelante.
 - e) Una vez seleccionados los datos que queramos, podemos darle al botón *Finish* para pasar a seleccionar el área que nos interese.

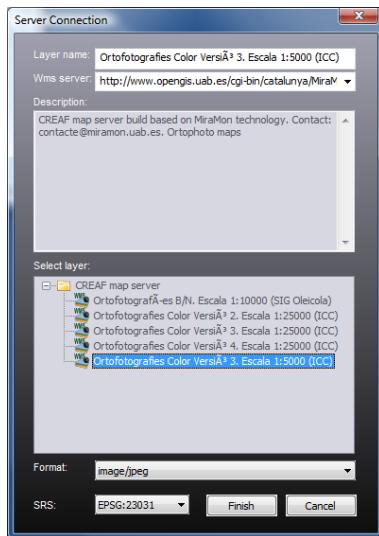


Figura 2.1: Ventana de conexión al servidor WMS.

3. Una vez establecida la conexión con el servidor, nos aparecerá una ventana (figura ??) para seleccionar el área que queremos y si todo ha ido bien, debe aparecer en la ventana principal, la imagen que corresponde a la capa seleccionada en el paso anterior.
4. En esta nueva ventana podemos ver, en la parte superior cuatro campos que corresponden a las coordenadas *UTM* del área que queremos procesar. Las dos primeras coordenadas corresponden a la esquina inferior izquierda y las otras dos a la esquina superior derecha. Estas coordenadas podemos ponerlas a mano, o bien, con el ratón haciendo click y arrastrando aparece un recuadro rojo que indica el área seleccionada (figura ??). Veamos como podemos movernos para seleccionar el área deseada:

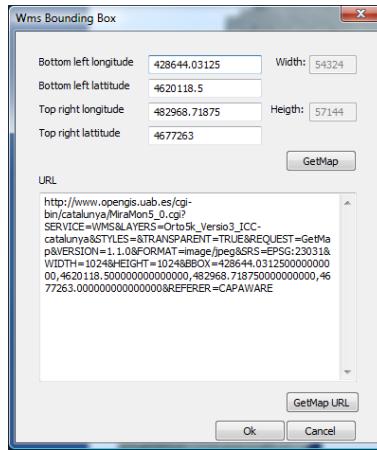


Figura 2.2: Ventana de selección de area y parámetros de la petición WMS.

- Con los cursores del teclado podemos movernos por la escena (arriba, abajo, izquierda y derecha).
- Con la rueda del ratón hacemos zoom en la escena, aunque no descarga nuevas imágenes, sólo es para tener mejor precisión en la selección. Este efecto también podemos conseguirlo pulsando el botón derecho del ratón y moviéndolo hacia arriba o hacia abajo.
- Si seleccionamos un subregión dentro del área visible, podemos aumentar la imagen en esa región al pulsar los botones **GetMap** o **GetMap URL**.
- Si lo que queremos es alejarnos y traer una imagen que abarque un área mayor, sólo debemos alejarnos (con la rueda o con el botón derecho del ratón), seleccionar un área mayor que la imagen visible y pulsar alguno de los botones **GetMap** o **GetMap URL**.

Una vez puestas las coordenadas, podemos hacer una previsualización pulsando el botón **GetMap**.

5. En la parte inferior de la ventana tenemos la *URL* con la petición que se hará al servidor. Esta *URL* podemos cambiarla y ajustarla con los valores que mejor nos convenga e incluso si ya tenemos una petición generada con cualquier otra aplicación, podemos utilizarla siempre que no variemos el servidor, ya que el servidor ya está preestablecido del paso anterior y las solicitudes van a él.
6. Si hacemos algún cambio en la *URL* y queremos ver si está bien y se ajusta a nuestras necesidades, pulsamos el botón **GetMap URL**. Si la petición es correcta aparecerá la imagen pedida, en caso contrario aparecerá un aspa roja para indicarnos que ha habido un error.
7. En cuanto hayamos ajustado el área y la *URL* pulsamos el botón *Ok* y pasaremos a seleccionar el servidor de alturas.
8. La nueva ventana que aparece, es igual que la de conexión al servidor *WMS*, pero ahora la utilizaremos para conectarnos al servidor *WCS* que nos proporcionará las alturas para la región previamente seleccionada.
 - a) En el apartado **Wcs Server** indicamos la url del servidor al que queremos conectarnos.

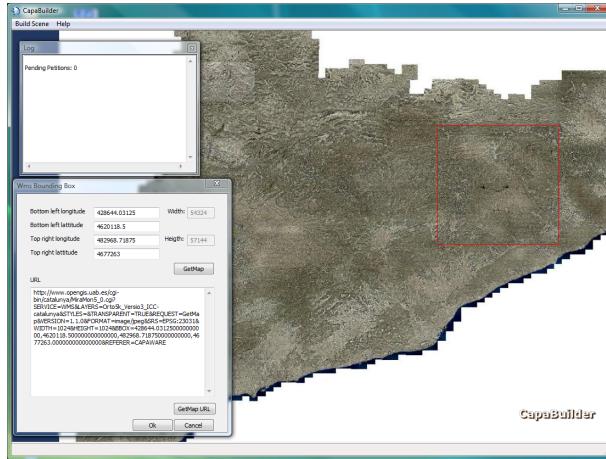


Figura 2.3: Selección de área WMS.

- b) Si la conexión se ha establecido correctamente en el campo **Description** aparecerá el mensaje de bienvenida del servidor, en caso de tenerlo.
- c) El campo **Select layer** se llenará con las capas disponibles en el servidor.
- d) Al seleccionar alguna capa, los campos **Format** y **SRS** se llenarán con los formatos de imagen y sistemas de proyección soportados por esa capa y deberemos seleccionar el que nos interese. **NOTA:** Se debe tener en cuenta que debemos seleccionar aquí el mismo *SRS* que hemos seleccionado para las *WMS*. Si los *SRS* son diferentes puede que no se ajusten correctamente las ortofotos y las alturas.
- e) Una vez seleccionados los datos que queramos, podemos darle al botón *Finish*.

A partir de ahora, la aplicación se conectará a los servidores seleccionados para descargar las imágenes necesarias para la generación. En la ventana *Log* irán apareciendo mensajes que nos indicarán que está haciendo la aplicación.

El proceso de generación dependiendo del área seleccionada puede ser bastante largo, tardando incluso varias horas.

Datos generados

Si no se ha producido ningún error, la aplicación generará en el directorio donde se encuentre el ejecutable dos carpetas que contendrán todas las imágenes descargadas. En una carpeta encontramos las imágenes de alturas y en la otra las ortofotos. **Importante:** El nombre de estas carpetas no debe cambiarse ya que si se hace, los datos contenidos en ellas no será accesible.

Además, la aplicación una vez acabada la generación del terreno, creará una carpeta en el directorio *data* que contendrá el fichero **.cws** necesario para la carga del terreno desde *Capaware* junto con una carpeta *terrain* que contendrá el fichero **.ive** y el resto de ficheros del terreno.

Un fichero **.cws** es un fichero editable que indica el fichero **ive** que representa al terreno que se quiere mostrar (también pueden ser más de uno), junto con las coordenadas UTM donde se encuentra dicho terreno, así como la altura en metros (por defecto cero para los terrenos). En este fichero también podemos especificar otros ficheros **ive** que representen cualquier otro tipo de objetos, no necesariamente

terrenos (por ejemplo, edificios, vehículos, etc.), y por lo tanto también se ha de incluir una coordenada de altura para cada fichero.

El ejemplo para la isla de El Hierro, que se encuentra en las coordenadas exactas, quedaría de esta forma en el fichero:

```

1 [OSG]
2
3 D:\\\\islasosg\\\\ehive_18\\\\eh.ive
4 188000.0
5 3060000.0
6 0

```

Después de generar un terreno, si falla algo en el proceso de generación o una vez generado el terreno vemos que alguna imagen se ha descargado defectuosa y se ve mal en el terreno, no hace falta que repitamos todo el proceso. En la misma carpeta donde se han descargado las ortofotos tenemos un archivo **.cbp**. Este fichero contiene la información suficiente para volver a lanzar el proceso sin que tengamos que repetir todos y cada uno de sus pasos. Sólo necesitamos iniciar CapaBuilder y en el menú *Build Scene* seleccionar *Load Project* y seleccionar el archivo que queremos cargar. El proceso se iniciará automáticamente.

En caso de que veamos que hay imágenes defectuosas, deberemos borrarlas del disco antes de cargar de nuevo el proyecto.

Hay que tener en cuenta que la carpeta con el fichero **.ive** generado puede ser bastante grande en espacio de almacenamiento. Por ejemplo, la isla de Tenerife, que ocupa una extensión aproximada de $2,000 \text{ Km}^2$, con una resolución de ortofoto de 1 metro por píxel, y una resolución de malla aproximada de 10 metros por vértice, requiere 15Gb en disco. La explicación para dicha cantidad de almacenamiento se debe a que internamente el terreno ha sido generado con diferentes niveles de detalles en una estructura quad-tree, guardando para cada nivel de detalle nuevas versiones de las ortofotos y de la malla (en el caso de Tenerife se han creado 10 niveles de detalle). Esta concepción por niveles permite un gran rendimiento en tiempo real mientras se sobrevuela el terreno, ya que en todo momento el sistema decidirá el nivel de detalle para cada zona, en función de la lejanía a la cámara, como se ve en la figura ??.

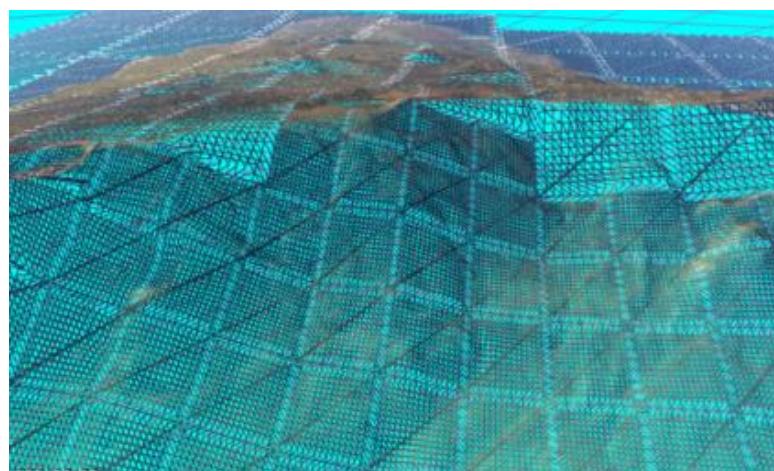


Figura 2.4: El nivel de detalle de cada mosaico se elige automáticamente en función de la distancia a la cámara

2.3. Ejecutando Capaware

Una vez generado el terreno, ya podemos ejecutar el Capaware. Inicialmente aparecerá vacío puesto que aún no hay escena insertada. Para cargar una escena debemos ir a la opción "File ->Open Project", y eligiendo el fichero ".csw" correspondiente nos la mostrará. Normalmente estos ficheros se encuentran en el directorio data.

Junto con todo el terreno aparece la interfaz de usuario de Capaware (ver figura ??). En la esquina superior derecha una brújula nos permite conocer en todo momento la relación del eje vertical de la pantalla con el norte del mapa.

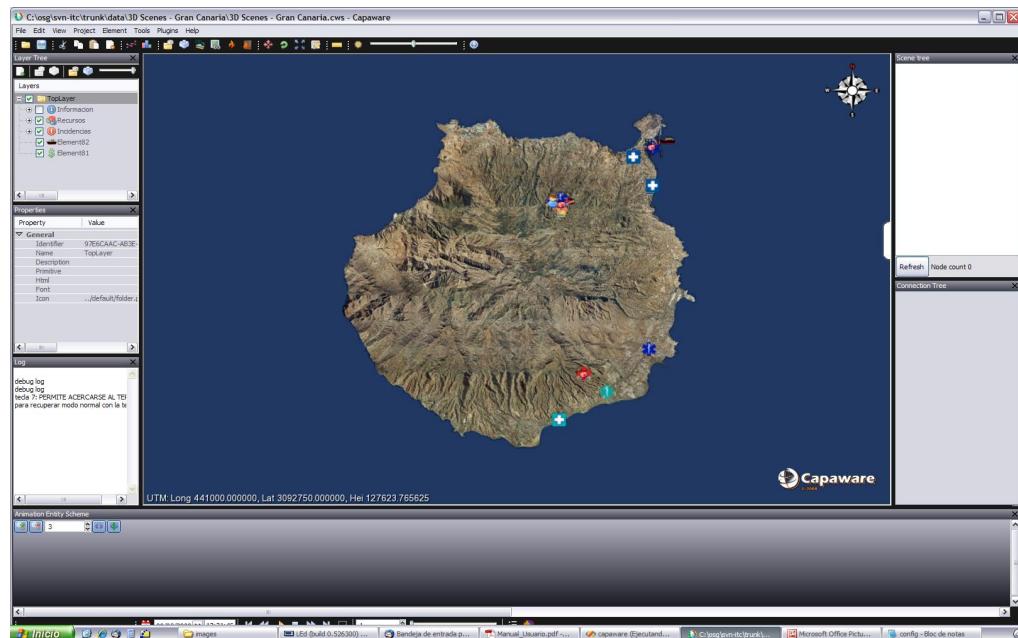


Figura 2.5: Vista de la aplicación con la escena de la isla de Gran Canaria, y diversos elementos insertados

Las distintas acciones de interacción con el ratón son las siguientes:

- Doble clic en un punto del mapa: Hace zoom sobre esa zona del mapa centrándola.
- Clic mantenido con el botón derecho del ratón: Acerca o aleja la zona del mapa.
- Clic mantenido con el botón izquierdo del ratón: Permite desplazar la zona de interés sobre el mapa manteniendo el zoom (ver figura ??).
- Rueda central del ratón: Acerca o aleja el punto sobre el que descansa el puntero (ver figura ??).
- Clic mantenido de ambos botones unido a desplazamiento del puntero: Permite rotar la vista sobre el mapa (ver figura ??).



Figura 2.6: Pulsando el botón izquierdo arrastramos la escena.

2.3.1. Navegación virtual

Capaware ofrece una forma fácil e intuitiva de navegar sobre el terreno. La rueda giratoria del ratón nos permite acercarnos a alejarnos en la escena. Con el botón izquierdo lo que conseguimos es desplazar o arrastrar el terreno a una posición deseada. Para rotar la vista actual y conseguir incluso pasar a la visualización 3D, pulsamos botón izquierdo y derecho del ratón simultáneamente. De esta forma nos aparece un icono verde con forma de cono invertido que permite especificar el sentido de la rotación (si movemos además el ratón a izquierda o derecha rotamos en estos sentidos, y si lo hacemos adelante o atrás, rotamos en el eje tridimensional). Para ayudar a conocer la orientación en la rotación, se dispone en la parte superior derecha de una brújula orientativa de la posición en todo momento. De estas variadas maneras conseguimos adecuar la perspectiva que el usuario elija.



Figura 2.7: Con la rueda del ratón hacemos zoom sobre la zona indicada.



Figura 2.8: Pulsando en ambos botones rotamos la escena alrededor del icono.

Capítulo 3

Uso de la aplicación Capaware

Una vez tenemos el terreno, la cosa no ha hecho más que empezar. En este capítulo analizaremos las posibilidades que permite la aplicación.

3.1. Visión general de la interfaz

A continuación se resumen las distintas opciones de la interfaz gráfica de la aplicación:

- **Barra de menús:**

- **File:** Opciones habituales para abrir, cerrar, almacenar un proyecto, así como para salir de la aplicación.
- **Edit:** Opciones de edición, copiar, cortar, pegar y eliminar.
- **View:** Cambia el modo de visualización entre 2D y 3D. Permite cambiar el modo de ver la escena, como *wireframe* (malla de triángulos) o modo con texturas (solid). Visualiza o no las distintas barras de botones y paneles en la interfaz gráfica. Permite visualizar pantalla completa y volver a la organización estándar de la interfaz.
- **Project:** Acceso a las operaciones básicas que se pueden realizar en un proyecto, como son la importación y creación de primitivas y elementos en la escena.
 - **Import:** Importa elementos de tipo entidad o capa ya existentes en disco, para su incorporación en el proyecto y escena actual.
 - **Create Template:** Crea una plantilla genérica para la creación de un determinado tipo de elemento. Las plantillas se utilizan para definir elementos similares de un mismo tipo para su rápida inserción en la escena.
 - **Container Layer:** Crea una plantilla para capas contenedoras. Las capas contenedoras se usan para organizar capas y elementos.
 - **Element:** Crea una plantilla para elementos de tipo modelos 3D.
 - **Add New:** Añade elementos nuevos a la escena.
 - **Container Layer:** Añade una nueva capa contenedora, basada en una plantilla.
 - **Element:** Añade un nuevo elemento 3D, basado en una plantilla.
 - **Wms Layer:** Permite añadir una capa de información desde un servidor WMS remoto.

- **Remote Entity:** Permite conectar a otro equipo para añadir entidades que han sido creadas remotamente y trabajar conjuntamente sobre ellas.
- **Fire:** Permite crear modelos de incendios basados en definición de perímetros y marcas de tiempos (ver figura ??).

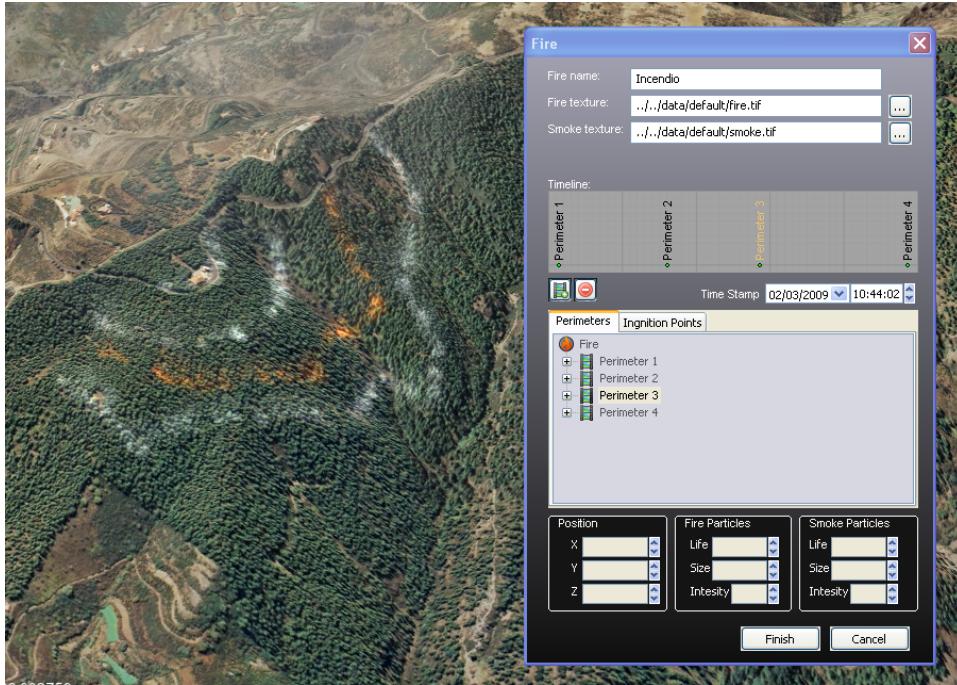


Figura 3.1: Creando un incendio con cuatro perímetros, picando directamente los puntos sobre el terreno

- **Firewall:** Permite crear cortafuegos para su visualización en la escena.
- **Element:** Permite acceder a las transformaciones que se pueden realizar sobre el elemento activo en la escena, así como a la edición de propiedades.
 - **Transform:** Realiza las transformaciones básicas de Traslación, Rotación y Escalado. En función de la transformación seleccionada, aparece un gizmo diferente asociado al elemento para poder realizarla de forma interactiva (ver figuras ??, ?? y ??).
 - **Edit Properties:** Abre el panel de edición de propiedades del elemento activo.
- **Tools:** Acceso a Herramientas como el cálculo de distancias.
- **Plugins:** Acceso los diferentes plugins cargados.
- **Help:** Información acerca de la aplicación.
- **Barra de botones superior:** contienen accesos rápidos a las opciones del menú: File, Edit, Project, Element, Tools, Plugins, Help ya comentados (ver apartado anterior: Barra de menús). También incorpora opciones de visualización de iluminación.
- **Barra de botones inferior:** contiene un indicador que muestra el estado de la aplicación y la barra de control de animación.

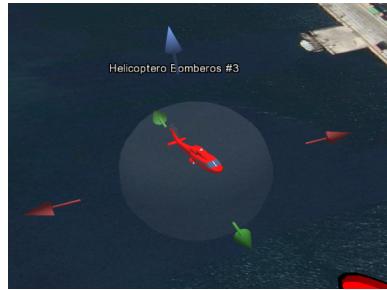


Figura 3.2: Gizmo de translación: pulsando en una de las flechas trasladamos sólo en una dirección. Haciendo click en la esfera trasladamos libremente.

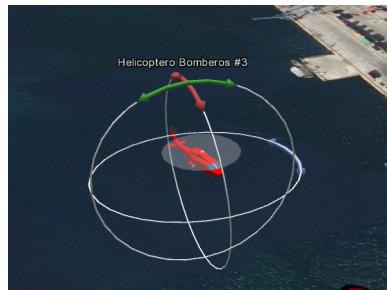


Figura 3.3: Gizmo de rotación. Pulsando en según qué órbita rotamos en los distintos ejes.

- **Treelayer:** Muestra una representación en forma de árbol del contenido de la escena actual. En la parte superior contiene algunas de las opciones de menú más comunes en la edición de elementos: Import, Create container layer template, Create element template, Add layer, Add element ya comentados (ver apartado anterior: Barra de menús, pág xx). También integra una barra que permite ajustar la transparencia de las capas de información WMS existentes. Si se hace clic secundario sobre un elemento también se puede acceder a las operaciones que se pueden realizar con dicho objeto (ver figura ??).
- **Ventana de propiedades:** Muestra información relevante acerca del objeto que se encuentra actualmente seleccionando.

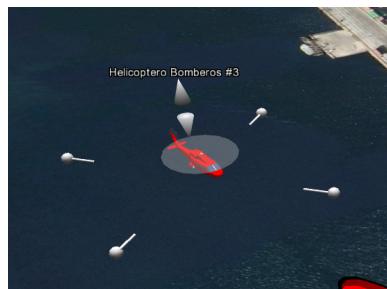


Figura 3.4: Gizmo de escalado. Podemos modificar el tamaño del elemento en las tres dimensiones o sólo en una.

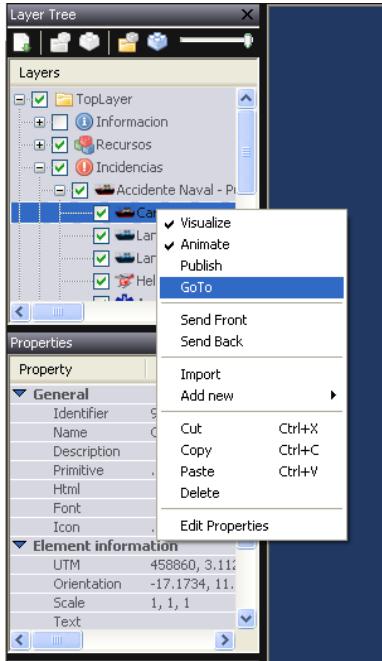


Figura 3.5: Existe un menú popup con las opciones más frecuentes para cada elemento del treelayer

- **Ventana de Navegación:** Muestra la vista 3D de la escena. Contiene información para la posición actual de punto de vista en UTM, así como una brújula y un minimapa (ver figura ??). También contiene una barra de botones desplegable para las opciones más frecuentes:

- Ver/Ocultar Brújula.
- Añadir nuevo Incendio.
- Añadir nuevo Elemento 3D.
- Añadir nueva Capa WMS.
- Ver/Ocultar Minimapas.

- **Animation Entity Scheme:** Muestra un esquema temporal con todas las entidades animadas que contiene la escena (ver figura ??). Permite hacer zoom sobre la vista, ajustar el esquema al ancho de la visualización o ir al momento actual de la animación. También permite navegar temporalmente haciendo clic izquierdo y arrastrando sobre el la barra de tiempo.

- **Connection Tree:** Representa en forma de árbol los elementos de la escena que se están usando de forma compartida entre varios equipos conectados entre sí.

- **Ventana de Log:** Es una ventana que muestra información relevante de la aplicación en tiempo de ejecución. Se puede usar para observar datos de importancia de la ejecución de plugins o para el desarrollo y depuración de herramientas sobre capaware.

- **Accesos Rápidos:**

- CTRL+N FILE-NEW

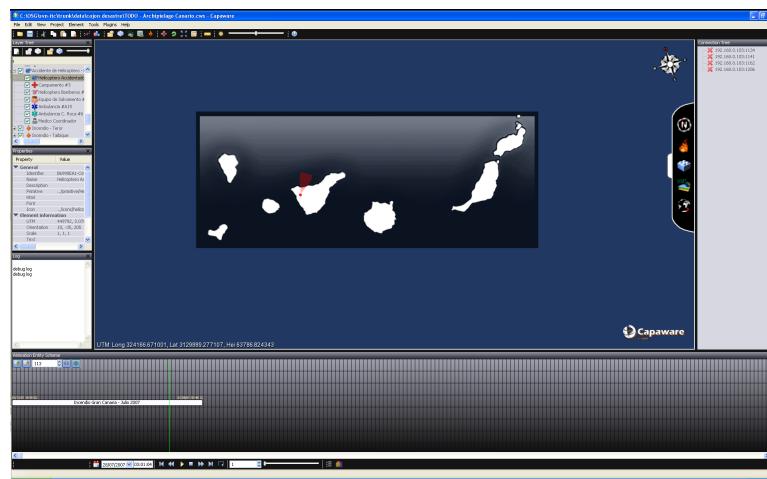


Figura 3.6: Al hacer click sobre el minimapa lo ampliamos en pantalla para poder ver nuestra situación actual, o poder desplazarnos automáticamente hacia una posición determinada.



Figura 3.7: La ventana muestra la lista de elementos animables en la escena

- CTRL+O FILE-OPEN
- CTRL+SHIFT+S FILE-SAVE
- CTRL+X EDIT-CUT
- CTRL+C EDIT-COPY
- CTRL+V EDIT-PASTE
- F9 VIEW-SHOWALL
- F10 VIEW-HIDEALL
- F11 VIEW-FULLSCREEN
- CTRL+T TRANSFORM-TRANSLATION
- CTRL+R TRANSFORM-ROTATION
- CTRL+S TRANSFORM-SCALE
- CTRL+P ELEMENT-PROPERTIES

3.2. Funcionalidades avanzadas

■ **Crear primitivas/plantillas:** Las plantillas se utilizan para definir elementos similares de un mismo tipo para su rápida inserción en el proyecto. La opción para crear nuevas plantillas se encuentra en el menú 'Project':

- **Container Layer:** Crea una plantilla para capas contenedoras. Las capas contenedoras se usan para organizar otras capas y elementos. Se caracterizan por su nombre y un ícono. Como puede verse en la figura ??, también se puede añadir una descripción y atributos.

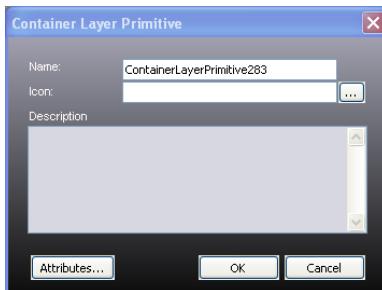


Figura 3.8: Una capa contenedora permite englobar otras capas y elementos.

- **Element:** Crea una plantilla para elementos de tipo modelos 3D. Los modelos 3D representan objetos o entidades en la escena. Se caracterizan por un nombre, un ícono y un modelo 3D. Como puede verse en la figura ??, también se puede añadir una descripción y atributos. Gracias a las plantillas se pueden insertar distintas instancias de un mismo objeto, pe. múltiples ambulancias. Estas instancias son editables independientemente y comparten el modelo 3D, ícono, y estructura de atributos, permitiendo una inserción mucho más rápida.

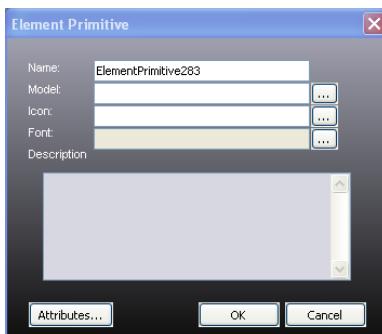


Figura 3.9: Interface para la creación de una primitiva, como paso previo para poder crear elementos a partir de ella.

■ **Crear y modificar capas:** Para insertar una capa en la escena es necesario tener previamente una plantilla en la que está basada. De esta forma, primero se selecciona la plantilla y la capa se insertará en el proyecto. A continuación se puede llenar y modificar los campos que se deseen. Para modificar una capa ya creada se puede hacer seleccionando la capa en cuestión en el Layer

Tree y haciendo click secundario->Edit Properties, apareciendo la ventana de la figura ???. Los campos del formulario se describen a continuación:

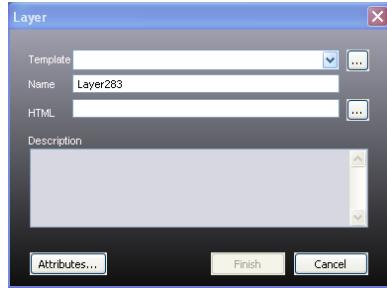


Figura 3.10: Interface para la creación de una capa.

- Template: Plantilla en la que está basada el elemento 3D.
 - Name: Nombre del elemento.
 - HTML: Etiqueta de HTML.
 - Description: Permite añadir un texto descriptivo.
- **Crear y modificar elementos 3D:** Para insertar un elemento 3D en la escena es necesario tener previamente una plantilla en la que está basada. De esta forma, primero se selecciona la plantilla y el objeto se insertará automáticamente en del centro de la vista sobre el terreno. A continuación se puede llenar y modificar los campos que se deseen. Para modificar un elemento ya creado se puede hacer seleccionando el objeto en cuestión en el Layer Tree y haciendo click secundario->Edit Properties o a través de las opciones Element->Transform, para las operaciones básicas, o Element->Edit Properties, para mostar el formulario completo (ver figura ??). Los campos del formulario se describen a continuación:

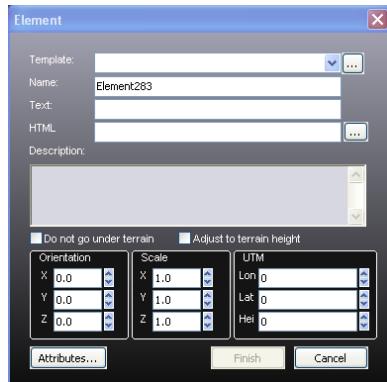


Figura 3.11: Interface para la creación de una capa.

- Template: Plantilla en la que está basada el elemento 3D.
- Name: Nombre del elemento.

- Text: Etiqueta de texto.
- HTML: Etiqueta de HTML.
- Description: Permite añadir un texto descriptivo.
- Do not go under terrain: Activar esta opción impide que el objeto atraviese y se quede por debajo del terreno.
- Adjust to terrain height: Activar esta opción ajusta el objeto para que se situe a nivel del terreno.
- Orientación: Muestra la orientación y permite rotar el objeto. Scale: Muestra la escala y permite agrandar o encoger el objeto. UTM: Muestra la UTM y permite mover el objeto.

■ Conexión a capas WMS:

Para insertar una capa WMS en la escena, hay que acceder a la opción Wms layer, bien a través del menú Project->Add New->Wms Layer o a través de la barra de botones disponible. Lo primero es añadir un nombre para la capa y a continuación se escribe la url del servidor al que se desea acceder o se elige de la lista predefinida. Tras esto sólo es necesario buscar la capa de información que se desea añadir de entre las disponibles, seleccionar el formato en que se desea descargar y que se guarde localmente y finalizar. Para modificar una capa ya creada se puede hacer seleccionando la capa en cuestión en el Layer Tree y haciendo click secundario->Edit Properties (ver figura ??). Los campos del formulario se describen a continuación:

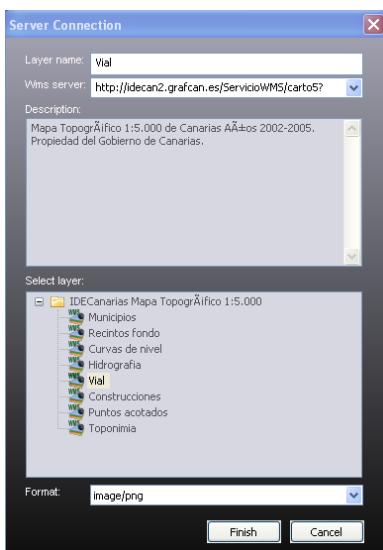


Figura 3.12: Conexión a capas WMS.

- Layer name: Nombre de la capa.
- Wms server: Url del servidor al que se desea conectar.
- Description: Información proveída por el servidor al conectar.
- Select layer: Muestra las capas disponibles.
- Format: Permite elegir el formato de la capa seleccionada.

- Imágenes no disponibles: Cuando al navegar se intenta acceder a una imagen que no está disponible, o el servidor no se encuentra funcionando, aparecerán imágenes señalas con una 'X' roja indicando que no se pueden mostrar.
- Borrado de la caché de imágenes: Las imágenes correspondientes a las capas de información son descargadas al directorio del proyecto. Para borrar la caché sólo es necesario acceder a dicho directorio y borrar las carpetas correspondientes.
- **Manejo del tiempo:** Se permite la creación de elementos animables. Un elemento animable está definido en un espacio de tiempo, declarado a partir de un instante inicial y un instante final. Capaware tomará por defecto los objetos animables y creará un espacio de tiempo donde estarán incluidos estos elementos.
 - **Panel Animation Entity Scheme:** Muestra los elementos animables representados como cajas dentro de un diagrama de tiempo. El panel permite ajustar la vista al total de la animación, al momento actual, ir a un determinado momento haciendo click sobre el gráfico o ajustar el zoom.
 - **Panel Animation Control:** Dispone de los controles clásicos para visualizar una animación, además de campos para mostrar y acceder a un momento concreto, un campo para ajustar la velocidad a la que se visualiza la animación y una barra de tiempo ajustable. También dispone de una opción para visualizar y editar la lista de entidades animables que quieren mostrar y una opción para la visualización de incendios forestales.
- **Conexión con otro usuario:** Los elementos de un proyecto pueden publicarse para permitir su acceso remoto en entornos colaborativos. Para publicar una entidad y convertirla en accesible se hace a través de la opción Publish, que se puede encontrar haciendo click secundario sobre el elemento en cuestión. Para insertar una entidad remota, hecha pública en otro equipo, hay que acceder a la opción Remote Entity, a través del menú Project->Add New->Remote Entity (ver figura ??). Los campos del formulario se describen a continuación:



Figura 3.13: Para conectar a un servidor remoto se hace mediante la dirección IP.

- Server IP: Dirección IP de la máquina que tiene la entidad publicada.
- Server Port: Puerto que se utiliza para las comunicaciones. Por defecto es el 3000.

Al conectar aparecerá otro formulario donde se expondrán las entidades publicadas disponibles (ver figura ??). A continuación se pueden seleccionar aquéllas que deseen, y añadirlas a la lista. Los campos del formulario se describen a continuación:

- Entities tree: Lista de entidades publicadas disponibles.
- Added Entities: Lista de entidades que se van a añadir.



Figura 3.14: Esta ventana nos indicaría que el equipo remoto al que nos queremos conectar tiene actualmente 6 elementos publicados, y que sólo queremos compartir tres de ellos

- **Utilidades:** Capaware dispone de una herramienta para medir distancias sobre el terreno. Puede accederse a través de la opción Distance, a través del menú Tools->Calculate->Distance (ver figura ??). Haciendo click sobre el terreno se irá dibujando una trayectoria que se ajustará al mismo y se mostrará el cálculo de la distancia del recorrido. El formulario muestra la posición UTM del último punto de referencia, la distancia total y las unidades en las que se muestra.



Figura 3.15: Con la herramienta distancia se pueden tomar mediciones directamente sobre el terreno.

- **Creación de un incendio:** Para insertar un incendio en la escena, hay que acceder a la opción Fire, bien a través del menú Project->Add New->Fire o a través de la barra de botones disponible. Lo primero es añadir un nombre para el incendio y a continuación se puede insertar perímetros asociados a instantes de tiempo y ajustar las propiedades del sistema de partículas que se deseé. Para modificar un incendio ya existente se puede hacer seleccionándolo en el Layer Tree y haciendo click secundario->Edit Properties (ver figura ??). Los campos del formulario para crear incendios se describen a continuación:

- Fire name: Nombre del Incendio.
- Fire texture: Textura a usar para las partículas que representan las llamas.
- Smoke texture: Textura a usar para las partículas que representan el humo.
- Timeline: Diagrama de tiempo que donde se mostrarán los perímetros que se crean en la simulación. Los perímetros deben crearse en orden. Éstos pueden seleccionarse para volverlos activos en el diagrama, haciendo click sobre ellos; y si se hace click y se arrastra se podrá mover el perímetro en el tiempo.



Figura 3.16: Interface para la creación de un incendio.

- **Añadir perímetro:** El botón creará un nuevo perímetro en el incendio al que se le asociará un tiempo. A este perímetro se le puede definir un contorno haciendo click sobre el terreno para añadir los puntos que lo forman.
- **Borrar:** El botón de borrar permite eliminar la llama o perímetro seleccionado si la pestaña activa es Perimeters. También permite eliminar un punto de ignición si la pestaña activa es la de Ignition Points.
- **Time Stamp:** Muestra el momento de tiempo asociado al perímetro activo. También permite modificarlo cambiando su valor. Hay que tener en cuenta que los perímetros deben insertarse en orden y que no se permite su intercambio.
- **Pestaña Perimeters:** Muestra una representación del incendio que se está creando.
- **Pestaña Ignition Points:** Permite ver y añadir puntos de ignición al incendio. Los puntos de ignición son marcadores visuales que indican dónde se origina el incendio. Para añadir un punto de ignición se sigue el procedimiento habitual de hacer click sobre el terreno.
- **Position:** Muestra y permite modificar las coordenadas de un punto de ignición o de un punto que define un perímetro.
- **Fire Particles:** Muestra la configuración del sistema de partículas de las llamas seleccionadas. Esta configuración es común para todos los puntos de un perímetro. Para modificarlo debe seleccionarse la primera llama del perímetro y editar los campos correspondientes:
 - **Life:** Tiempo de vida de las partículas. Cuanto mayor sea este valor más tiempo durarán y más alto llegarán las llamas creadas.
 - **Size:** Tamaño de las partículas. Cuanto más alto sea este valor mayor serán las llamas.
 - **Intensity:** Intensidad. Incrementar este valor aumentará el número y densidad de las llamas. Se recomienda un valor moderado para mantener un buen rendimiento gráfico.

- Smoke Particles: De forma análoga define las propiedades del sistema de partículas que representa la columna de humo del incendio. Se basa en los mismos parámetros de Life, Size e Intensity.

- **Creación de un cortafuegos:** Para insertar un cortafuegos en la escena, hay que acceder a la opción Firewall, bien a través del menú Project->Add New->Firewall o a través de la barra de botones disponible (ver figura ??). Lo primero es añadir un nombre para el cortafuegos y a continuación se puede insertar los puntos que definen el perímetro y ajustar las propiedades que se desee. Para modificar un cortafuegos ya existente se puede hacer seleccionándolo en el Layer Tree y haciendo click secundario->Edit Properties. Los campos del formulario para crear incendios se describen a continuación:

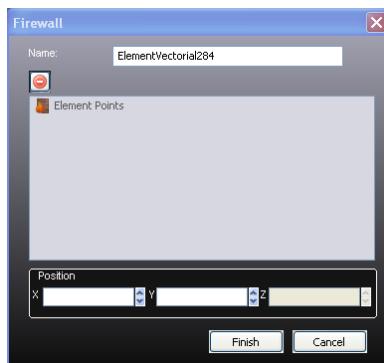


Figura 3.17: Desde la ventana firewall se puede ir haciendo click sobre el terreno para generar una polilínea que represente un cortafuegos.

- Name: Nombre del cortafuegos.
- Delete: Botón que permite borrar un punto.
- Element Points: Muestra los puntos que definen el perímetro del cortafuegos y permite seleccionarlos.
- Position: Muestra y permite modificar la posición del punto seleccionado.

- **Crear un incidente:** Crear una incidente es un proceso que requiere trabajar en distintos niveles. Se aconseja seguir una pautas para mantener un proyecto ordenado y con el que sea fácil de trabajar. A continuación se detallan los pasos recomendados para la creación de incidentes de forma organizada:

- Disponer de un repositorio de modelos 3D adecuado a las necesidades. Capaware dispone de una colección de modelos 3D amplio, enfocado a la creación de incidentes, que se encuentra en el directorio /data/models. Si se necesita añadir nuevos modelos pueden crearse en cualquiera de los formatos admitidos por OSG y añadirse a este directorio.
- Tener claro el tipo de información que se quiere representar y su organización.

- **Crear primitivas:** Si es la primera vez que se va a crear un tipo de objeto 3D o capa de información es necesario crear la primitiva en la que está basada. Es recomendable utilizar un nombre genérico que represente el tipo de entidad que se va a crear, pe. crear la plantilla para capas contenedoras de tipo Información, con nombre "Capas de Información".

- **Crear la jerarquía de capas:** Es importante organizar bien el proyecto sobre el que se trabaja creando capas contenedoras para estructurar el contenido. Hay que tener en cuenta que un incidente puede contener gran cantidad de elementos y seguir una buena organización siempre ayuda a tener un entorno con el que sea más fácil trabajar. Para evitar confusiones se recomienda utilizar nombres concretos para cada elemento, pe. "Toponimia" basada en la plantilla "Capas de Información". Las capas pueden reorganizarse haciendo click y arrastrando en el Layer Tree. También se permiten las operaciones de Copiar/Cortar/Pegar.
- **Crear e insertar los elementos:** Se aconsejan las mismas pautas para la creación de elementos 3D que para el caso anterior. También se recomienda organizar los elementos después de insertarlos en la escena y guardar el proyecto con frecuencia.

3.3. ¿Y a partir de aquí?

La aplicación Capaware es una aplicación bastante genérica, con lo cual el usuario puede personalizarla para lo que necesite. Puede ver cualquier tipo de capa remota WMS que se encuentre en servidores públicos externos, y en las próximas versiones se añadirán nuevos estándares OGC. Puede insertar cualquier tipo de objeto tridimensional, y colocarlo sobre el terreno. Sin embargo, si lo que desea es implementar alguna función nueva sobre el entorno, o desarrollar su propia aplicación geográfica, entonces también dispone de varias alternativas, que se mencionan en los siguientes capítulos.

A modo de ejemplo, se ha desarrollado sobre Capaware una aplicación más avanzada, a petición del Cabildo de La Palma, para poder gestionar todos sus efectivos en una situación de incendio, así como realizar simulaciones. Con ese objetivo se desarrolló la aplicación Geviemer (Gestor Virtual de Emergencias), construida a partir de Capaware, que incorpora un plugin de simulación de incendios para que el usuario pueda realizar estudios sobre el terreno y predecir el comportamiento de un incendio forestal en una zona concreta. Además, incluye capas de contenidos específicos en el área de gestión de emergencias. En la figura ?? se muestra una captura de pantalla.

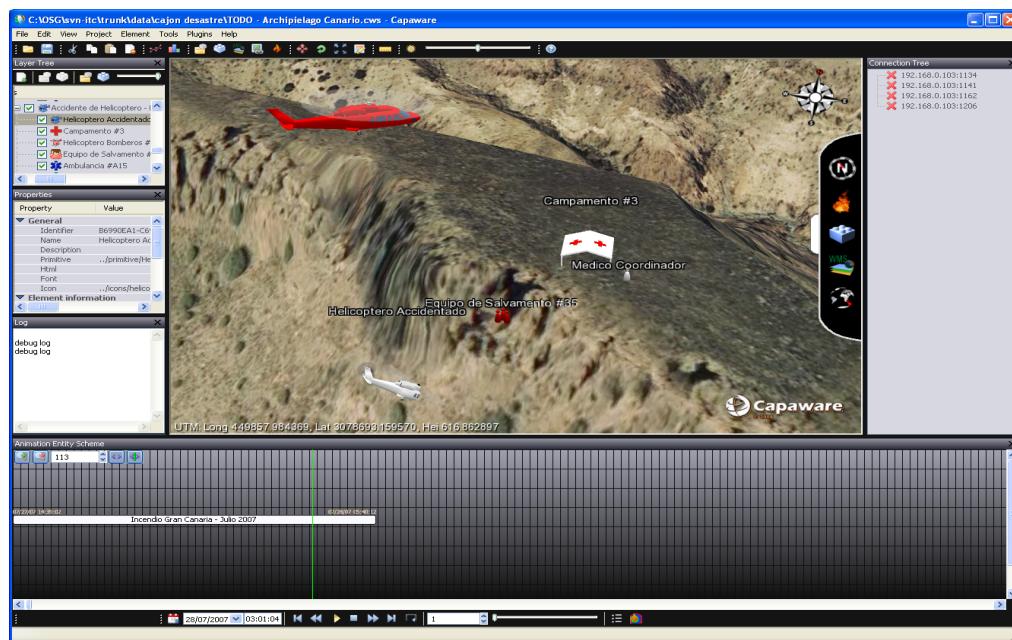


Figura 3.18: Gestor Virtual de Emergencias (GEVIEMER)

Capítulo 4

Programación de plugins

La forma más sencilla de introducir una nueva funcionalidad dentro de Capaware es desarrollando un plugin. El sistema de plugins que tiene integrado Capaware permite que desarrollemos un proyecto independiente C++, en donde hagamos uso de la API CPW, el cual será llamado desde dentro del Capaware cuando el usuario lo requiera, a través de una opción adicional que se añadirá en el menú Plugin de la interfaz general de Capaware. Veamos los pasos para lograr esto, así como un par de ejemplos.

4.1. Creación de un plugin básico

La forma más fácil de empezar es utilizar un plugin de ejemplo básico ya creado, y modificarlo. En la carpeta src/plugins existe un ejemplo llamado "Ejemplo_plugin_1" que es el que comenzaremos a utilizar. Al abrirlo nos aparecen tres módulos. Los dos primeros (cpw y cpwutils) son parte de Capaware, y el tercero (Ejemplo_plugin) es sobre el que trabajaremos.

Dentro de este módulo, el único fichero que nos importa inicialmente es exports.cpp. A su vez, dentro de este fichero, las dos funciones del final son en las que realmente vamos a trabajar: *RegisterMenu* y *ExecPlugin*.

En la función *RegisterMenu* sólo hay que indicar en la string menu cuál es el texto que deseamos que aparezca en la opción del menú.

La función *ExecPlugin* es la que se va llamar cuando el usuario pulse en la opción. En este primer ejemplo, vamos a añadir un elemento 3D (una excavadora) en el centro de la pantalla, sobre el terreno. Para ello, lo primero que debemos es hacer es hallar las coordenadas del terreno que corresponden a esa posición, y de eso se encargan las primeras tres líneas de código:

```
1 cpw::INavigator *navigator = navigator_manager->GetFocusedOrFirstNavigator()
2 ;
3 navigator->GetCenterScreenCoords (cx, cy);
4 navigator->IntersectMouseWithScene (cx, cy, x, y, z);
```

El objeto *navigator_manager* es el que maneja los navegadores, y nos llega como parámetro a la función *ExecPlugin*. Con la primera línea obtenemos el objeto *navigator*, que apunta a la vista actual. Con la segunda línea, utilizamos un método de este objeto para obtener las coordenadas del píxel del centro de la pantalla. Finalmente, con la tercera línea, lanzamos un rayo desde la posición del

observador hacia ese píxel, y obtenemos la intersección de dicho rayo sobre el terreno. Es en ese punto donde insertaremos el elemento.

A continuación debemos crear el elemento, añadirle un callback a Capaware para que este lo pueda manejar desde su interfaz, y añadir un segundo callback para que Capaware lo guarde en disco:

```
1 cpw::Element3D *el = new cpw::Element3D;
2 el->AddCallBack(new cpw::Model3DCALLBACK);
3 el->SetPersistentCallBack(new cpw::PersistentFileCallBack);
```

Una vez creado, hay que indicar las propiedades del elemento, tales como nombre, coordenadas XYZ, orientación, escala, ícono a mostrar cuando el observador esté muy lejos, y modelo geométrico del elemento:

```
1 el->SetName("Elemento3D_plugin");
2 el->SetUtm(x, y, z);
3 el->SetOrientation(0, 0, 180);
4 el->SetScale(1,1,1);
5 el->SetIcon("../data/icons/excavadora.png");
6 el->SetModelUrl("../data/models/excavator/excavator.osg");
```

El último paso consiste en añadirlo a la lista de entidades del tree-layer, para que nos aparezca en el árbol de la izquierda, que representa todo lo que se encuentra contenido en la escena:

```
1 ventity.push_back(el);
```

El objeto *ventity* que nos viene por parámetros es una lista de entidades, inicialmente vacía, que es donde añadiremos todos las entidades que nuestro plugin va a crear.

Una vez terminado el código, hay que compilar el proyecto, lo cual generará una dll (en debug o en release dependiendo de la configuración que hayamos tomado). A continuación hay que informar a Capaware de que hay una nueva dll que añadir. Esto se hace editando el fichero *pluginsfile.dat* que se encuentra en la carpeta de Capaware, y añadiendo la dll junto con la ruta que queremos añadir. Realmente esto sólo debe hacerse la primera vez que creamos un nuevo plugin. Como este ejemplo ya estaba creado, ya esta línea aparece en el fichero, y no hace falta añadir nada. Podemos hacer todos los cambios que queramos al código de ese plugin y generar la nueva dll, que la aplicación Capaware, sin necesidad de recompilarse, los cogerá.

Ahora ya podemos ejecutar Capaware, y veremos que en el menú plugin se ha añadido una opción que es la que hemos añadido. Si nos acercamos a ras del suelo, y ejecutamos el plugin, veremos cómo aparece la excavadora sobre el terreno, como se ve en la figura ???. Dependiendo de si estamos lejos o cerca del terreno, veremos el modelo 3D sobre el terreno, o bien un ícono. Como se puede apreciar, también aparece el elemento en el tree-layer de la izquierda, desde el cual podemos ocultarla o volar hacia ella.

En el panel inferior pueden verse las propiedades del objeto, que son la que hemos introducido por código. Ahora podríamos cambiarlas, bien clicando sobre la excavadora y utilizando el gizmo, o bien a través de la opción "Edit Properties" del menú popup al hacer click sobre la excavadora en el tree-layer.

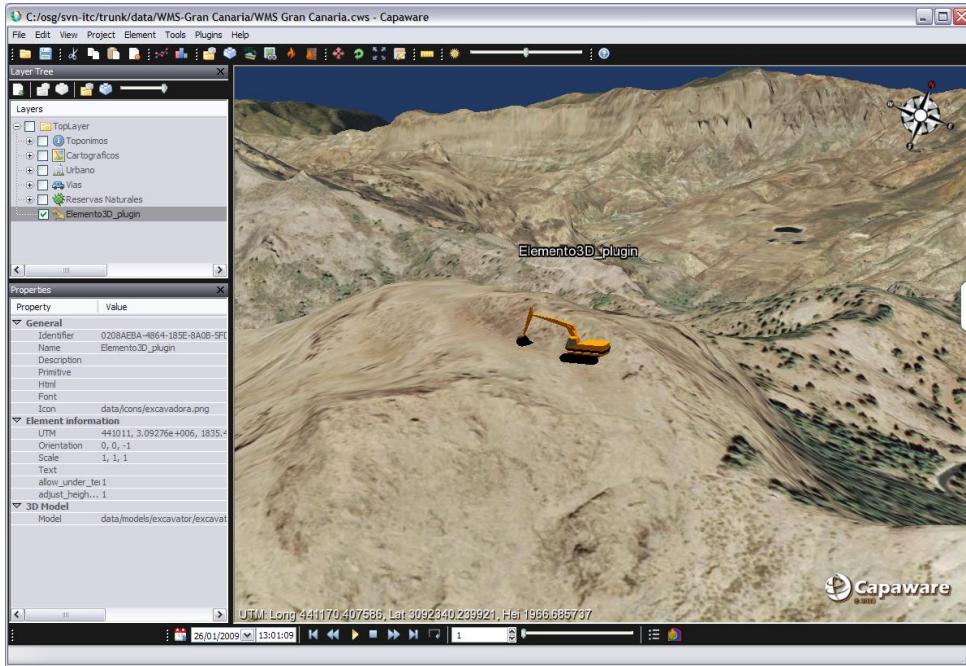


Figura 4.1: La excavadora tal y como la insertamos en el plugin

4.2. Animación de elementos

Ahora que ya sabemos hacer un plugin mínimo para insertar un elemento, veamos cómo podríamos hacer para animarlo. Para ello, abramos el segundo ejemplo que se encuentra en la carpeta `src/plugins`.

Para poder animar debemos crear una clase que herede de la clase `Model3DCallback`, y que sobreesciba un método virtual llamado `Animate`, que es donde escribiremos el código que deseamos ejecutar en cada frame, donde iremos animando el elemento. Lo primero que haremos será obtener el puntero al objeto animado:

```
1 cpw::Element3D *e3d = (cpw::Element3D *) callable;
```

A continuación obtenemos su posición XYZ:

```
1 double *pos = e3d->GetUtmd();
```

En este ejemplo simplemente vamos a ir incrementando su posición X en una unidad. Hay que tener en cuenta que esta función se llamará en cada fotograma, con lo cual, a una tasa de refresco media de 60 fps, nos estaremos moviendo a una velocidad media de 60 unidades. Como las unidades de la escena están en coordenadas UTM, y éstas representan metros, la velocidad sería de 60 m/s.

```
1 pos[0] += 1;
```

Una vez incrementada la posición, debemos asegurarnos que el elemento se mantenga pegado al terreno. Para ello, haremos uso del objeto `scene` (más adelante veremos cómo acceder a él desde el

método *Animate*) y de un método que me devuelve el punto intersección entre un rayo y el terreno. De esta manera, la coordenada Z vendrá dada siempre por la altura del terreno, justo en la posición XY en la que se encuentra el elemento:

```

1 cpw::Point3d<double> i_point;
2 if (scene->IntersectRayWithTerrain(
3     cpw::Point3d<double>(pos[0], pos[1], 10000.0f),
4     cpw::Point3d<double>(pos[0], pos[1], -10000.0f),
5     i_point, true))
6 {
7     pos[2] = i_point.z;
8 }
```

Finalmente, actualizamos la nueva posición en la clase del elemento, y avisamos a Capaware del cambio:

```

1 e3d->SetUtmd(pos);
2 e3d->GraphicUpdate();
```

En cuanto a la función principal *ExecPlugin*, el único cambio que debemos hacer con respecto al ejemplo anterior es registrar el callback para la animación, y pasarle como parámetro el objeto *scene*:

```
1 el->SetAnimateCallBack(new GraphicE3DCallBack(appscene->GetScene()));
```

Si probamos este segundo plugin sobre Capaware, veremos cómo nada más introducir la excavadora, ésta empieza a desplazarse por el terreno indefinidamente.

4.3. Uso del panel de animación

En el ejemplo anterior, hemos animado un objeto, pero no teníamos control alguno sobre el eje temporal. Es decir, no había forma de pararla, ni de ir hacia atrás. Para este tipo de cosas existe el panel de animación, que ya se comentó en la guía de usuario. Veamos cómo podemos insertar nuestro elemento como un bloque dentro del panel de animación. Abramos ahora el ejemplo 3.

Fijándonos en el código de la función *ExecPlugin*, hemos añadido un nuevo bloque, en el que vamos a indicarle un tiempo de comienzo y uno de final asociado al elemento. De esta forma, ya podrá tener cabida en el panel de animación, porque tiene un intervalo concreto donde mostrarse. El código es el siguiente:

```

1 cpwTime tim;
2 tim.seconds = 0;
3 tim.milliseconds = 0;
4 el->SetStartTime(tim);
5 tim.seconds = 1200;
6 el->SetEndTime(tim);
```

En este ejemplo estamos indicando que el tiempo de comienzo es el instante actual¹, y la duración

¹La clase *cpwTime* representa el número de segundos y milisegundos transcurridos desde el 1 de Enero de 1970. En el caso particular cuando indicamos un valor cero, Capaware sustituye el valor por el instante actual.

de la animación va a ser de 20 minutos. De esta forma, aparecerá en el panel de animación justo con ese intervalo temporal.

La función *Animate* también debe ser modificada, ya que ahora el paso del tiempo puede no ser continuo siempre hacia adelante. Por ejemplo, el usuario puede usar los controles de pausa, adelante o atrás de la barra inferior de la aplicación, e incluso puede mover el cursor temporal (línea vertical de color verde en el panel de animación) para desplazarse en el tiempo. Esto significa que a nuestra función *Animate* nos llegará el instante que debe mostrarse en cada momento, y por lo tanto, el cálculo de la animación no puede ser progresivo con respecto a los valores de la anterior iteración, como hacíamos antes, sino calcularse directamente a partir del instante actual.

Para ello debemos primero obtener el valor inicial de la posición, usando una variable estática

```
1 static double x0 = e3d->GetUtmd(0);
```

A continuación calculamos el tiempo transcurrido entre el instante actual y el inicial. El tiempo inicial es el que indicamos en el momento de insertar el objeto, y el tiempo final nos llega a la función en el parámetro *time_stamp*.

```
1 long int duracion = diff_time (e3d->GetStartTime(), time_stamp);
```

Y por último, actualizamos la posición en X en función de la duración:

```
1 pos[0] = x0 + 0.01*duracion;
```

Si probamos a ejecutar el nuevo plugin, veremos que al insertar la excavadora, ésta no se mueve. Sin embargo en el panel de animación nos aparece un bloque de 20 minutos, empezando en el instante actual, que representa toda la animación. Por defecto, la barra temporal está en pausa, y por eso la excavadora está fija. Si pulsamos play, la veremos mover. También podemos mover el cursor temporal y ver la posición de la excavadora en cada instante de la animación, como se ve en la figura ***.

4.4. Cálculo de simulaciones

En lugar de hacer caminar al elemento siempre en la misma dirección, podemos hacer que la dirección se calcule en tiempo real en función de la máxima pendiente en cada posición, tal y como haría un objeto redondo que pusiéramos sobre un terreno inclinado. Tomemos el ejemplo 4.

En este caso hemos cambiado el modelo, de una excavadora a una pequeña bola azul. La función *ExecPlugin* prácticamente sigue igual. En la función *Animate*, la hemos hecho similar al ejemplo 2 (sin usar el panel de animación), en la que vamos actualizando la posición con respecto a la posición anterior.

Para ello calculamos el vector gradiente sobre el terreno en la posición actual:

```
1 double grad[2];
2 ComputeGradient (scene, pos, grad);
```

Y a continuación avanzamos la bola en la dirección contraria al gradiente. De esta forma, el objeto bajará por la dirección de la máxima pendiente.

```
1 pos[0] -= grad[0];
```

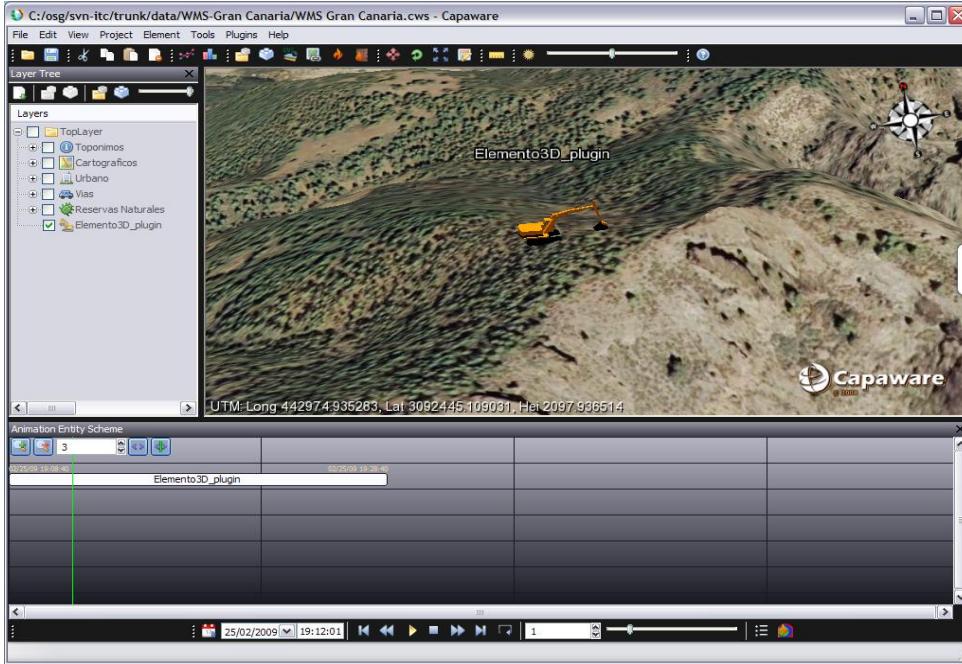


Figura 4.2: La animación de la excavadora aparece en el panel de animación

```
2 pos[1] -= grad[1];
```

Si lo ejecutamos, veremos cómo la bola va disminuyendo siempre en altura por el camino más corto. De hecho, podemos seleccionar el objeto con el gizmo de traslación, llevarlo a cualquier sitio, y al soltarlo seguirá rodando pendiente abajo desde esa posición, como se ve en la figura ??

También hemos añadido un *print* de pantalla en cada frame para ir mostrando la posición del elemento, y el vector gradiente calculado.

```
1 printf ("Estoy en (%.4f, %.4f, %.4f). Gradiente = <%.4f, %.4f>. ", pos[0],  
    pos[1], pos[2], grad[0], grad[1]);
```

El texto que volquemos con la función *print* se verá en la ventana de *log*. Si dicha ventana no se está mostrando, la podemos hacer visible desde el menú *View*. Este sistema es muy útil para ir mostrando mensajes de información durante la ejecución de un plugin.

4.4.1. Añadir simulaciones al panel de animación

Si quisieramos añadir la simulación anterior para poder verla en el panel de animación, entonces el funcionamiento sería diferente. Habría que hacer todo el cálculo de la trayectoria en la función ExecPlugin, almacenarlo en un vector de coordenadas en función de cada tiempo, y fijar un intervalo total de duración para la animación hasta que la bola quedase fija en un sitio de pendiente cero. De esa forma, en la función Animate sólo habría que mostrar la posición correspondiente al instante temporal marcado en el parámetro time_stamp. De esta forma podríamos ver la simulación hacia adelante o atrás, más rápida o más lenta.

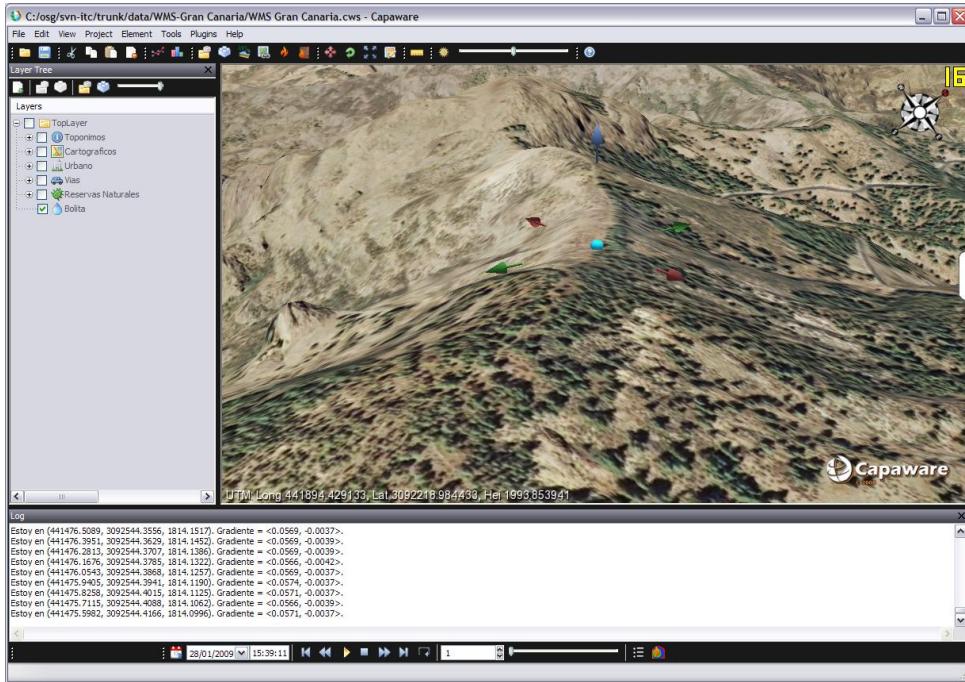


Figura 4.3: Según en que posición soltemos la bolita, ésta caerá de un lado o del otro de la montaña

En la sección siguiente veremos cómo hacer esta simulación, y además se mostrará la forma de crear una interfaz de usuario.

4.5. Plugin con interfaz de usuario

Nuestro último ejemplo nos permitirá mostrar cómo podemos crear una interfaz de usuario para nuestro plugin. Teóricamente podría usarse cualquier librería de interfaz gráfica, pero en nuestro ejemplo usaremos wxWidgets, que es la que se ha usado en toda la aplicación.

Abramos el ejemplo 5. En este proyecto hemos creado un fichero cpp nuevo, dentro de la carpeta *Source Files*, llamado *UITest.cpp*. Es en este fichero donde escribiremos el código relacionado con la interfaz.

Dentro de este fichero, la función importante se encuentra al final, y se llama *CreateGUIControls*. En ella estamos creando una ventana con dos cajas de texto para indicar la posición inicial de la bola, una tercera caja de texto para indicar el número de segundos que durará la simulación, un botón de aceptar, y otro de cancelar. Todos estos componentes se crean en la forma habitual de trabajar con wxWidgets. La ventana final se muestra en la figura ???.

Vayamos ahora con la función *ExecPlugin*. Los dos primeros bloques para calcular el centro de la pantalla y para crear un nuevo elemento van igual que antes. A continuación, insertamos manualmente de forma temporal el elemento 3D para poder verlo sobre el terreno antes de cerrar la ventana.

```
1 | el->GraphicInsertAndRegister();
```

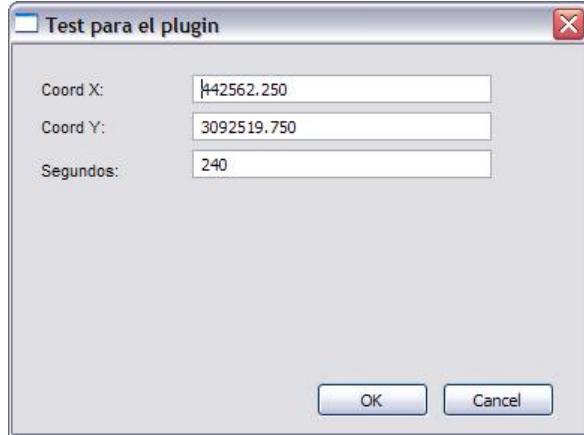


Figura 4.4: Interfaz hecha en WxWidgets

Esto nos permite que una vez hemos ejecutado el plugin, y con la interfaz todavía mostrándose en pantalla, podemos hacer doble clic y mover la bola por donde queramos, actualizándose sus coordenadas en las cajas de texto.

Para poder actualizar estos valores en la interfaz, hemos escrito código dentro de la función *MouseEvent* de la clase *UITest*, que obtiene la intersección del ratón con el terreno, y coloca los valores de la posición obtenido en las cajas de texto. También a modo de ejemplo, se muestra en la ventana de log la posición en 3D donde se ha picado con el ratón.

```

1 char Texto_x[100], Texto_y[100];
2 navigator_manager->GetFocusedOrFirstNavigator()->IntersectMouseWithScene(x, y
   , ipx, ipy, ipz);
3 printf ("estoy picando en la escena!! Punto UTM: (%.2f, %.2f, %.2f) Texto:
   s", ipx, ipy, ipz, tc_name->GetValue().c_str());
4 sprintf (Texto_x, "% .2f", ipx);
5 tc_name->SetValue (Texto_x);
6 sprintf (Texto_y, "% .2f", ipy);
7 tc_name2->SetValue (Texto_y);

```

Seguimos con el código de la función *ExecPlugin*. Lo siguiente es crear la ventana y mostrarla en pantalla:

```

1 UITest *ui = new UITest (ventity, navigator_manager, el->GetUtmd(0), el->
   GetUtmd(1), parent, wxID_ANY, wxString(_T("Element")), wxDefaultPosition
   , wxSize(390, 360));
2 ui->>ShowModal();

```

A continuación eliminamos el elemento temporal:

```

1 el->GraphicDeleteAndUnregister();

```

Ahora preguntamos si el usuario pulsó aceptar:

```
1 | if (ui->GetReturnCode() == wxID_OK) {
```

En caso afirmativo, calculamos la simulación. Para ello, crearemos una tabla de valores, donde en cada fila iremos almacenando el instante y la posición de la bola en cada momento. La tabla tendrá tantas filas como segundos haya indicado el usuario en la interfaz. Toda esta tabla es calculada en este momento.

Finalmente, procedemos igual que en el ejemplo 3, fijando un callback de animación, rellenando los parámetros para el panel de animación, y añadiendo el elemento a la lista de entidades de salida del plugin.

Si nos fijamos ahora en la función *Animate*, veremos que ahora lo único que hay que hacer es, en función del parámetro *time-stamp*, decidir el instante que debe mostrarse, ir a la tabla de simulación, y mostrar la posición correspondiente.

Apéndice A

Trabajo futuro

La presente versión 1.0 de Capaware ha cumplido los requisitos establecidos inicialmente antes de su desarrollo. Sin embargo, por el camino, numerosas ampliaciones y proyectos futuros han ido surgiendo a medida que avanzaba el desarrollo. Las principales líneas que se pretenden continuar son:

- Poder generar la escena al vuelo, al estilo de otras aplicaciones como Google Earth
- Permitir conectar a más tipos de capas remotas: KML, WFS, WCS, OSM, etc.
- Permitir mostrar información GPS asociada a elementos 3D
- Desarrollar nuevos plugins y aplicativos sobre el SDK