INTERFAZ DE USUARIO DE UN SISTEMA MULTI-ROBOT DE RECONOCIMIENTO DE SUPERFICIES CON GOOGLE EARTH

Memòria del projecte

d'Enginyeria en Informàtica

realitzat per

Carles Pagès Rozas

i dirigit per

Lluís Ribas-Xirgo

Bellaterra, 16 de setembre de 2009

ÍNDICE

Capítulo 3: Interfaz de Usuario

Capítulo 4: Caso de Prueba

Capítulo 5: Conclusiones

5.1 REVISIÓN DE LOS OBJETIVOS

5.1.1 OBJETIVOS PRINCIPALES

5.1.1.1 COMUNICACIÓN CON OTROS AGENTES

5.1.1.2 MOSTRAR DATOS SOBRE LA INFORMACIÓN GEOGRÁFICA DE GOOGLE EARTH

5.1.2 OBJETIVOS SECUNDARIOS

5.1.2.1 NAVEGADOR WEB COMO PLATAFORMA PARA LA IU

5.2 LINEAS DE CONTINUACIÓN

5.2.1 SISTEMA DE ESCANEO PARA LA GENERACIÓN DE FICHEROS COLLADA

5.2.2 REDUCCIÓN DE LOS TIEMPOS DE CARGA

Capítulo 1.

INTRODUCCIÓN

1.1 CONTEXTO DEL PROBLEMA. CASO SENCILLO DE EJEMPLO

La información geográfica y de los mapas son elementos que aumentan notablemente las capacidades de muchas aplicaciones hoy casi universalmente accesibles tales como los calculadores de rutas o las guías interactivas basados en GPS. La accesibilidad a la información anterior ha sido posible, entre otros, gracias a Google que la ha puesto a disposición de sus usuarios de forma gratuita. Es posible, pues, pensar en desarrollar aplicaciones que utilicen información geográfica y cartográfica en base a Google Earth y Google Maps [1].

En este trabajo se pretende demostrar la viabilidad de utilizar este tipo de motores geográficos como plataforma para una interfaz capaz de mostrar información espacial generada por un sistema sobre superficies previamente documentadas, cartografiadas y de dominio público. El hecho de mostrar estos mapas, junto a la información generada por el sistema, proporciona al usuario una visión más amplia del problema analizado y otorga un mayor contexto a la situación gracias a las numerosas referencias geográficas de las que se dispone.

Este hecho puede resultar clave a la hora de analizar situaciones ubicadas en grandes extensiones de terreno, como bosques, montañas, ciudades, etcétera... donde el hecho de poder establecer referencias geográficas y cartográficas pueden facilitar las labores de análisis del problema.

Para situar el ámbito de los problemas que se pretenden resolver, se describen a continuación diversos casos prácticos que ejemplifican el dominio de aplicación en el que se trabaja.

Disponemos de un sistema multi-robot [2] encargado del reconocimiento cartográfico de una pequeña área de terreno. El propósito de estos es obtener de una manera precisa y constantemente actualizada los detalles del relieve que conforman este área, ya sea con carácter natural (por ejemplo, un valle, una montaña...) o una estructura creada por el hombre (el interior de un edificio, una infraestructura de transporte de materiales...).

Este sistema tendría multitud de posibles aplicaciones: un ejemplo sería la investigación de sitios inhóspitos aún no documentados, como por ejemplo grutas de interés para espeleólogos, aún pendientes de ser cartografiados.

Otro fin podría ser el análisis del área de un espacio que haya sufrido un desastre natural. Centrémonos en este ejemplo para dar una utilidad práctica real:

Imaginemos un desastre natural, como por ejemplo un terremoto, o un alud de nieve sobre una zona habitada. El relieve del terreno podría haber sufrido notables variaciones respecto a la documentación geográfica de la zona que tendrían los servicios encargados de supervisar la catástrofe. Una mala documentación podría suponer contratiempos importantes a la hora de preparar todo el contingente.

De aquí viene la necesidad de disponer de primera mano, y actualizado en tiempo real, de un plano detallado de la zona analizada, para poder maximizar la calidad del operativo previo de emergencia.

Este plano detallado nos lo podría proporcionar este sistema de robots: Para el ejemplo del terremoto o del alud de nieve, podríamos enviar un grupo de helicópteros no tripulados a la zona del desastre, justo en el momento de conocer la noticia del desastre. Ellos se encargarían de proporcionarnos inmediatamente una visión del estado de la zona. Por ejemplo, podríamos encontrarnos con que la vía de acceso principal a la zona se he visto bloqueada por un montón de rocas. Antes de que el equipo humano

acuda a la zona, ya dispondríamos de esa información, haciendo que los coordinadores de emergencias puedan establecer otros caminos alternativos para llegar al destino con antelación.

Ahora bien, sí, tenemos un sistema de robots capaces de analizar el terreno, pero nos surgen una serie de cuestiones si nos ponemos en la piel del coordinador de emergencias:

- **Deberemos saber de qué forma accederemos a toda esta información.**
- **Necesitaremos un interfaz de usuario lo suficientemente sencillo para que cualquier persona pueda consultar.**
- **Los nuevos datos obtenidos sobre la región analizada deberán poder combinarse con los datos previamente recopilados de los alrededores de dicha región.**

1.2 MOTIVACIONES

A continuación se comentan algunas de las motivaciones que nos impulsan a llevar a cabo este Proyecto de Fin de Carrera [3] para atacar al problema que se acaba de plantear.

1.2.1 GOOGLE EARTH COMO APLICACIÓN PARA VISUALIZACIÓN DE TERRENO

Para responder a las cuestiones que se planteaba en el anterior apartado nuestro coordinador del desastre, y en definitiva, cualquier usuario de nuestra aplicación, debemos proveer un interfaz que resulte accesible, intuitivo, de uso fácil, dinámico y completo.

Para ello elegimos como base donde se apoyará nuestro interfaz la aplicación Google Earth, desarrollado por Keyhole para Google. Google Earth es una conocida aplicación gratuita disponible para los tres sistemas operativos más utilizados para ordenadores personales: Windows, Linux y MacOS.

Figura 1.1: Vista de la Universitat Autònoma de Barcelona a través de Google Earth

Este programa nos permite visualizar imágenes por satélite de todo nuestro planeta (y también de otros planetas cercanos de nuestro sistema solar, como Marte), mediante un motor streaming con información geográfica almacenada en los servidores de Google. Simplemente con introducir en el formulario de búsqueda el nombre del lugar que deseamos visualizar, el programa nos muestra una vista aérea por satélite de ese sitio. Una vez cargada, podremos acercar o alejar el zoom, mover la cámara con el ratón, etcétera.....

Además de esta información gráfica de base, Google Earth nos permite superponer elementos a las imágenes visualizadas, permitiéndonos interactuar con el terreno.

Estas capas habitualmente se presentan como marcas en un único punto del mapa con enlaces de interés a los lugares que visitamos. Por ejemplo, si visualizáramos el Taj Mahal, junto al edificio aparecerían marcas con enlaces a web que nos suministrarán información sobre este edificio, por ejemplo el artículo de Wikipedia sobre el Taj Mahal.

Figura 1.2: Superposición de la vista del Taj Mahal con información de Wikipedia

Google Earth incorpora por defecto una serie de marcas vinculadas a servicios web con recursos de información general, como la anteriormente comentada Wikipedia, Panoramio (que suministran fotos añadidas por la comunidad de usuarios tomadas desde el punto donde se sitúa la marca), servicios relacionados con el transporte público, etc... Sin embargo, podemos añadir otros motores web de forma sencilla.

Conviene remarcar que aunque esta información nos está siendo suministrada mediante el protocolo HTTP, protocolo utilizado para mostrar páginas web a través de un navegador como Firefox, Safari, Chrome o Internet Explorer, la información se muestra directamente integrada en Google Earth, haciendo el programa independiente de otros navegadores, e incrementando la interacción entre información, mapa y usuario. Incluso si necesitásemos navegar entre páginas relacionadas, Google Earth nos ofrece un pequeño navegador, basado en Chrome, que se integra en la ventana del programa.

Otro uso de la superposición de capas en el mapa es la superposición de imagenes, líneas, polígonos y modelos en 3D. Esta técnica se basa en añadir directamente sobre el mapa otras imágenes o figuras poliédricas que no se corresponden a la información geográfica almacenada en los servidores de Google. Gracias a esto podríamos superponer imágenes o poliedros obtenidas por otros medios que añaden información a la información recogida por Google más deficiente o desactualizada. También es útil para mostrar estructuras como edificios de forma detallada, o añadir diagramas y colores, por ejemplo para añadir mapas de densidad de población, o vías de comunicación como carreteras, redes ferroviarias, etc...

En cuanto a los poliedros, nos resultan muy útiles para representar estructuras con volumen, evidentemente no representables en un mapa plano, dado que Google Earth es capaz de simular una perspectiva no cenital a partir de los mapas de que dispone. Estas estructuras podrían ser edificios, infraestructuras, accidentes geográficos....

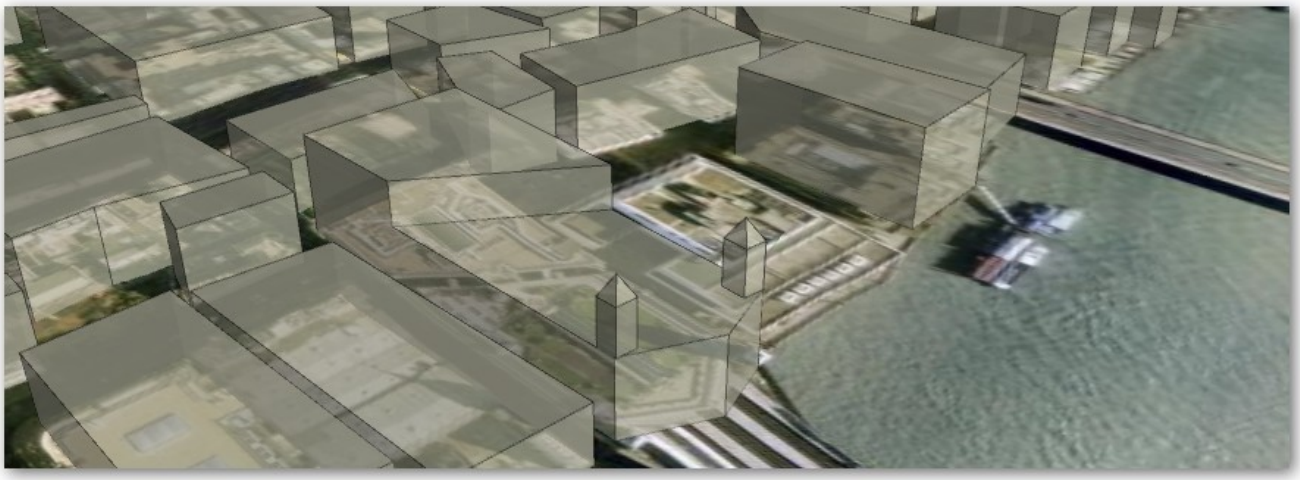

Figura 1.3: Modelo con polígonos en tres dimensiones representando edificios de la ciudad de Londres.

1.2.2 GOOGLE EARTH COMO PLATAFORMA PARA EL SOPORTE DE NUESTRA INTERFAZ

Como vemos, Google Earth brinda un gran abanico de opciones para visualizar y añadir información extra, accesible sobre el mapa, y fácil de obtener y personalizar para el propósito que nos planteemos.

Entonces, y volviendo a nuestras cuestiones con nuestro interfaz de usuario, parece que nuestra pregunta de cómo representar y donde mostrar los planos devueltos por nuestro sistema de robots parece haberse contestado ya:

Insertaremos directamente los elementos encima del mapa, de forma que en nuestra ventana de Google Earth veremos nuestro terreno analizado, superpuesto con la imagen original de Google Earth. De esta forma combinaremos la precisión y actualización del mapa generado por nuestros robots, con los alrededores de este, suministrados por Google.

A pesar de que Google ofrece de forma pública su API para el desarrollo de aplicaciones basadas en Google Earth, hoy en día aún no es habitual ver aplicaciones de carácter cartográfico basados en ella. Este Proyecto de Final de Carrera nos puede resultar útil para comprobar hasta que punto son viables este tipo de aplicaciones.

De vuelta a nuestro coordinador de emergencias, éste simplemente deberá ejecutar Google Earth en su ordenador y cargar el motor de información relacionado con nuestro sistema multirobot. Acto seguido le aparecerá centrada en pantalla la imagen generada, junto el resto de información que requiera integrada en el programa, ya se trate de localidades cercanas al lugar para desplegar la base de operaciones de un posible dispositivo de emergencia y vías de comunicación, o de contenidos web útiles para la coordinación de la operación, como por ejemplo números de teléfono, formularios para la trasmisión de órdenes vía Internet, etc...

1.2.3 GOOGLE EARTH COMO FUENTE DE INFORMACIÓN CARTOGRÁFICA GRATUITA

Existen numerosas empresas dedicadas a la obtención de información geográfica y cartográfica. Un ejemplo serían las empresas dedicadas a la comercialización de dispositivos de posicionamiento de automóviles sobre la red de carreteras mediante el sistema GPS. Para el uso de este tipo de dispositivos, es necesario previamente disponer de la documentación cartográfica en la que el usuario pueda ubicar su vehículo.

Generalmente estas empresas no ofrecen sus bases de datos públicamente, sino que exigen la contratación de sus servicios. Google es una excepción, ya que ofrece gratuitamente a los usuarios la documentación geográfica de la que dispone en Google Earth. De hecho no solo ofrece públicamente sus mapas, sino también las herramientas necesarias para manipularlos, como la inserción sobre ellos de nuevos elementos.

La elección de Google Earth como fuente de información cartográfica evidentemente supone una gran ventaja de cara a la viabilidad económica de un sistema de este tipo.

1.2.4 EL NAVEGADOR WEB COMO PLATAFORMA DE LA INTERFAZ

Hoy en día el navegador web es una herramienta fundamental, presente en la mayoría de sistemas operativos, y a la cual están acostumbrados prácticamente la totalidad de usuarios de sistemas informáticos. La grandes bazas de los navegadores son la comunicación con sistemas remotos basadas en el protocolo TCP/IP (fundamentalmente HTTP) y la posibilidad de mostrar contenidos multimedia, como imágenes, audio y vídeos en streaming, etcétera...

En los últimos años han aparecido un gran número de aplicaciones basadas en el navegador web que no son solo documentos estáticos, como clientes de correo electrónico, reproductores de vídeo, interfaces para el control para redes sociales, etcétera.. De hecho la inmensa mayoría de los productos desarrollados por Google se basan en el navegador web.

El hecho de basar el desarrollo de una aplicación sobre un navegador web supone un ahorro de esfuerzo por parte del usuario, ya que lo sitúa en un entorno menos desconocido del que sería una aplicación independiente.

Además, muchos desarrolladores están apostando fuerte por esta filosofía, haciendo que los navegadores sean plataformas cada vez más refinadas y con mayores prestaciones.

1.3 DEFINICIÓN DEL PROBLEMA

Una vez tenemos ya claro a grandes rasgos de que forma queremos plasmar las capas que contendrán nuestros elementos, ahora debemos marcar los pasos de cómo vamos a enlazar la información cartográfica que nos suministrará el sistema multi-robot con Google Earth.

La primera consideración que debemos tener en cuenta es que, para que Google Earth pueda correr, necesitaremos un requisito fundamental: disponer de acceso a Internet, ya que la aplicación obtendrá de los servidores de Google toda la información necesaria para desplegar las imágenes por satélite. Sin Google Earth, este proyecto carece de sentido.

Con esta premisa, podemos apoyarnos en un servicio remoto conectado a Internet que enlace Google Earth con el sistema multi-robot.

1.3.1 CÓMO INSERTAR INFORMACIÓN EN GOOGLE EARTH

Como se comentaba anteriormente en el capítulo 1.2, todas las capas generada por nuestro proyecto que queramos desplegar, ya se trate de la información geográfica generada por los robots, o de información extra transmitida por el protocolo HTTP, deberá ser insertada sobre el mapa. La pregunta es, ¿de qué forma le indico a Google Earth que cargue estos elementos?

Google Earth es capaz de mostrar sus mapas a través de un navegador web, mediante un plug-in desarrollado por la propia Google instalable en el navegador. Junto a este plug-in se suministra una API con funciones dirigidas a insertar elementos sobre el mapa. Esta API está implementada en lenguaje Javascript, un lenguaje interpretado capaz de ser ejecutado por los navegadores web.

El primer problema es **poder mostrar la salida que proporciona Google Earth a través del navegador web, y saber cargar en el mapa los elementos que recibiremos.**

1.3.2 CÓMO OBTENER INFORMACIÓN DE OTROS AGENTES

Como sistema para probar la viabilidad de este trabajo, se desarrollará una interfaz para el sistema multi-robot que se comenta en el punto 1.1 de la memoria como demostración.

Este sistema consiste en una serie de robots Exploradores, capaces de explorar áreas del terreno y analizar sus rasgos geográficos, detectando obstáculos, variaciones de la pendiente, etcétera... Cada uno de estos robots constituye un agente del sistema.

Todos los datos del terreno que van recopilando los robots Exploradores son entregados a otro agente del sistema, implementado en Java, llamado Cartógrafo [4]. Este agente es el encargado de construir un mapa a partir de los datos suministrados.

Además este agente Cartógrafo es capaz de comunicarse con otras aplicaciones o agentes del sistema, tal y como hace con los robots Exploradores, a través de JMS (Java Message Service) [5, 6]. Este servicio sirve para administrar colas de mensajes y enviárselas a otras aplicaciones que también tengan la funcionalidad JMS. Dado que la interfaz de usuario será también uno de los agentes, deberá poder comunicarse con el resto de agentes del sistema usando el mismo protocolo que éstos utilizan. Para entregarnos la información, el Cartógrafo enviará mensajes en forma de cadenas de caracteres, codificando los detalles de la información del terreno a representar.

Nuestro segundo problema es que **necesitaremos un agente interfaz de usuario que sea capaz de recibir esta información del Cartógrafo y con ella generar modelos de representación del terreno, como marcas, imágenes, modelos en 3D, etcétera... que sean mostrados al usuario, y que a su vez el usuario sea capaz de enviar ordenes para que las reciban los otros agentes.**

1.3.3 CÓMO INTERACTÚA EL USUARIO CON EL SISTEMA PARA CONSEGUIR LA INFORMACIÓN

Uno de los principales objetivos de nuestro interfaz es que debe ser extremadamente sencillo para el usuario visualizar en Google Earth toda la información que busque sobre el mapa generado por el sistema multi-robot.

Teniendo en cuenta que uno de los principales requisitos es el acceso a Internet, debemos aprovechar esto para usar herramientas habituales para la navegación por esta red, puesto que el usuario espera no tener que realizar complejas operaciones, ni perder tiempo para configurar la vista. Además, si este sistema es empleado en dispositivos de emergencia, la velocidad para conseguir los datos puede ser crucial.

El tercer problema es **buscar una manera de que el usuario disponga de la información de una forma rápida y sencilla por medios habituales.**

1.4 OBJETIVOS

Para responder a las tres cuestiones propuestas en el último punto, primero las resumiremos en una sola:

Necesitamos un interfaz de usuario (a partir de ahora IU), rápido y sencillo de manejar, que se encargue de recibir la información geográfica del Cartógrafo, y que ésta sea introducida en los mapas de Google Earth, que será visto por el usuario a través de un navegador web. Además, el usuario deberá ser capaz de mandar órdenes al Cartógrafo.

En este Proyecto de Final de Carrera se pretende conseguir los siguientes objetivos:

1.4.1 OBJETIVOS PRINCIPALES

Se fijan como objetivos principales aquellos que se presentan como fundamentales para demostrar la viabilidad del proyecto. A conseguir estos se dedica la principal cantidad de horas de ingeniería.

1.4.1.1 COMUNICACIÓN CON OTROS AGENTES

Dado que la información cartográfica que el usuario visualizará es generada y suministrada por otros agentes del sistema, y a su vez el usuario debe poder solicitar órdenes a estos agentes a través de la IU, ésta debe ser capaz de poder comunicarse de forma bidireccional con ellos.

Los agentes ya existentes en el sistema utilizan el protocolo JMS (Java Message Service). Por lo tanto la IU deberá implementar este protocolo para comunicarse con ellos.

1.4.1.2 MOSTRAR DATOS SOBRE LA INFORMACIÓN GEOGRÁFICA DE GOOGLE EARTH

La información cartográfica recibida por el agente Cartógrafo deberá ser renderizada sobre la información que ofrece Google Earth. A través de la API que Google ofrece, se manipulará el mapa agregando los diferentes elementos que habremos cargado, y se definirán sus atributos para su correcta visualización y ubicación respecto a los mapas de Google Earth. Además, los elementos deberán ser actualizados periódicamente para que el usuario pueda comprobar si ha ocurrido algún cambio, y sin que ello signifique recargar toda la aplicación.

1.4.2 OBJETIVOS SECUNDARIOS

El objetivo secundario también debe ser tenido en cuenta. Sin embargo el cumplimiento de éste no se presenta tan decisivo como el de los principales, por lo que no se le ha dedicado tantas horas de trabajo como los anteriores.

1.4.2.1 NAVEGADOR WEB COMO PLATAFORMA PARA LA IU

En los últimos años se ha visto incrementado el número de aplicaciones que son ejecutados remotamente en un servidor desde el navegador web, sin requerir la instalación de software adicional. Esto comporta mayor conocimiento y comodidad al usuario sobre el entorno de trabajo.

El objetivo será que el usuario interactúe con la IU a través de un navegador web, de forma que pueda acceder a él en cualquier máquina que disponga de él, sin importar el sistema operativo en el que trabaje.

1.4.2.2 MÓDULOS DISTRIBUIDOS REMOTAMENTE

Dado que para acceder a las bases de datos de Google se requerirá acceso a Internet, se podría aprovechar este hecho para distribuir los diferentes módulos que conformen la arquitectura de la IU en diferentes máquinas, y que se comuniquen a través de la red, en función de las necesidades. Igualmente seguiría siendo posible albergar todos en una máquina.

1.5 EVALUACIÓN DE COSTES Y RIESGOS

1.5.1 EVALUACIÓN DE COSTES

Lo deseado en el desarrollo de este Proyecto de Fin de Carrera, si lo independizamos del sistema multi-robot en sí, es que no debe suponer un importante desembolso de puesta en marcha ni requiere un gran mantenimiento. Esto es debido fundamentalmente a que, hoy en día, el uso de las tecnologías basadas en Internet se encuentra en franca expansión.

En base a ésto, y a los puntos expuestos a continuación, podemos atrevernos a asegurar que el Proyecto es técnicamente viable.

1.5.1.1 EVALUACIÓN DE COSTES EN LICENCIAS

A lo anteriormente comentado debemos sumar la filosofía de los desarrolladores de estas tecnologías en brindar al público en general no solo herramientas gratuitas, sino una amplia documentación para implementar aplicaciones basadas en éstas.

Todas las tecnologías que utilizaremos para la implementación del interfaz, como Java y PHP [7, 8] , son gratuitas y se pueden conseguir fácilmente a través de Internet, tanto sus módulos principales como bibliotecas auxiliares. Además, en Internet podemos encontrar amplia documentación para su uso.

Por otra parte, si bien Google es propietaria de todas las tecnologías y aplicaciones que desarrolla, su famosa filosofía de ofrecer gratuitamente sus servicios a los usuarios le ha convertido en uno de los máximos responsables de la importancia de los recursos en Internet hoy en día. Además Google provee una extensa documentación para todas sus aplicaciones, incluida Google Earth, que anima a toda la comunidad de usuarios a desarrollar recursos para sus aplicaciones.

Por lo tanto, y para empezar, debemos tener en cuenta que para poder trabajar con todas las tecnologías que necesitaremos para implementar la aplicación no necesitaremos ningún tipo de desembolso económico. El permiso para utilizar Google Earth o para desarrollar aplicaciones con las tecnologías que utilizaremos supone un gasto de cero euros en licencias.

1.5.1.2 EVALUACIÓN DE COSTES EN INFRAESTRUCTURA

Para analizar el coste del soporte a la aplicación, debemos tomar como comparación cualquier otro modelo de aplicación cliente-servidor:

Para el cliente, simplemente necesitaremos un ordenador con acceso a Internet y que sea capaz de hacer correr Google Earth. No es necesario adivinar que la inmensa mayoría de ordenadores personales domésticos cumplen sobradamente con estos requisitos.

En cuanto al servidor, evidentemente nuestro script deberá ser alojado en una máquina conectada a Internet capaz de servir la información a los clientes que se lo soliciten. Si bien puede ser clave el rendimiento del servidor en cuanto a respuesta: primero porque aparte de enviar información a los clientes, también se debe realizar el vínculo con el Cartógrafo. Este vínculo es más sensible, ya que sin él no obtendremos información alguna de los robots, y por tanto no podremos mostrar la información.

El vínculo con el Cartógrafo se puede realizar remotamente a través de Internet sin problemas, pero nos ahorraríamos problemas de conexión o tiempos de respuesta si tanto la IU como el resto de agentes del Cartógrafo residen en la misma máquina, o en la misma red local.

En cuanto a la tasa de información que debe proporcionarnos el servidor para los clientes, tampoco necesitamos un servidor privilegiado: las estructuras de datos de tamaño máximo que moveremos por el sistema serán imágenes, cuyo tamaño aproximado será de alrededor de 200 KB. Cualquier conexión a Internet doméstica habitual es capaz de descargar esta cantidad de información en pocos segundos.

1.5.1.3 EVALUACIÓN DE COSTES EN TIEMPO DE TRABAJO

Debemos tener en cuenta las horas de trabajo dedicadas a la realización de este Proyecto, tanto por parte del estudiante que lo presenta (que se estiman en unas 450 horas en total para un Proyecto de Final de Carrera de Ingeniería Informática), como por parte del Director de Proyecto.

1.5.2 EVALUACIÓN DE RIESGOS

Dado que nuestra aplicación no deja de tener un carácter meramente de enlace entre otras dos aplicaciones, es necesario entender que cualquier error que ocurra estará basado en un fallo del soporte de la IU, es decir, de las aplicaciones que permiten correr PHP, Java o el navegador web.

Dada la popularidad e implantación de este estándar en Internet, afrontar un fallo del sistema de este tipo puede ser solventado por cualquier técnico con conocimientos en mantenimiento de estas máquinas.

Otro posible error ajeno a nuestro enlace sería el fallo de los dos sistemas enlazados: Google Earth y Cartógrafo. En el caso del primero evidentemente se trataría de un fallo de programación en sí, o de la máquina cliente. También podría darse el caso que los servidores de Google se encontraran inactivos por alguna razón, con lo cual no podríamos obtener los datos geográficos que Google Earth proporciona, o incluso podría darse el caso de ser incapaces de ejecutar la aplicación. En cuanto a un posible fallo del Cartógrafo, deberíamos ponernos en contacto con los responsables de éste para averiguar que problemas han sucedido.

Podríamos aceptar como riesgo contratiempos durante la implementación del script, que podrían demorar la obtención de resultados. También cabría la posibilidad de tener que afrontar cambios en cuanto a la filosofía de los desarrolladores de las tecnologías que usa nuestro Proyecto. Imaginemos por ejemplo que en el futuro Google decide que el uso de sus aplicaciones deja de ser gratuito. Esto nos llevaría a replantearnos la viabilidad del Proyecto si supone un desembolso económico mayor.

Resumiendo, nuestra sistema es determinista (para una misma información suministrada por el Cartógrafo generará el mismo resultado) y acotado (no recibiremos información inesperada que el sistema no sepa tratar). Cualquier error que se pueda producir con la ejecución del interfaz no estaría relacionado con la implementación del script, una vez ésta ha sido completada con éxito.

1.6 TAREAS Y PLANIFICACIÓN

El desarrollo de este Proyecto de Fin de Carrera comportará las siguientes tareas, ordenadas cronológicamente y con la estimación del tiempo dedicado a cada una, que en total suman 450 horas. Agruparemos estas tareas por fases, y haremos una breve explicación de lo que comportará cada una de ellas.

También indicaremos la planificación prevista inicialmente por fechas. Evidentemente esto es una estimación inicial, es posible que los tiempos previstos aumenten o se reduzcan durante el transcurso del desarrollo del Proyecto.

1.6.1 PRIMERA FASE: PREVIO

Los primeros pasos del Proyecto evidentemente consistirán en entender el problema, y realizar una estimación de todos los requerimientos que necesitaremos para desarrollar el interfaz. Ésto conlleva una documentación sobre las tecnologías que se usarán para llevar a cabo nuestro proyecto.

Esta fase nos introducirá en el objetivo del Proyecto, y nos dará una idea de lo que se pretende conseguir, así como evaluar costes y riesgos, establecer la planificación, etcétera... Finalmente se hará un breve resumen de las tecnologías con las que trabajaremos.

Esta fase, de 61 horas de duración, se empieza a desarrollar el 1 de diciembre de 2008 y finaliza el 10 de enero de 2009, si bien a este punto de la memoria ya estará finalizado el primer capítulo.

Recopilación de requerimientos (10 horas)

Recopilación de información sobre las funcionalidades de Google Earth y su plug-in para navegador web (10 horas)

Recopilación de información sobre otras aplicaciones basándose en Google Earth como interfaz (8 horas)

Recopilación de documentación sobre el protocolo JMS (3 horas)

Recopilación de información acerca del Estado del Arte: Google Earth (5 horas)

Elaboración del primer capítulo de la memoria: "Introducción" (25 horas)

1.6.2 SEGUNDA FASE: INTERFAZ EN BASE AL NAVEGADOR

En esta fase se implementará el documento web que servirá para mostrar el mapa de Google Earth, y para ubicar los controles que servirán para que el usuario interactúe con el interfaz.

Aún no se tocará ningún tema relacionado con el Cartógrafo, ya que en principio esta clase pretende ser de propósito general, si bien las funcionalidades que presentará se limitarán a los aspectos que conciernen a este Proyecto. Por tanto, para probar esta clase usaremos imágenes y coordenadas simuladas por nosotros para poder mostrarlas en el mapa de Google Earth, y no generadas por el Cartógrafo.

Esta fase, de 127 horas de duración, se empezará a desarrollar el 10 de enero y concluirá el 20 de marzo de 2009.

Preparación del entorno de trabajo de la segunda fase (2 horas)

Implementación de la estructura del documento web (30 horas)

Implementación las funciones para la situación de elementos sobre el mapa (35 horas)

Implementación del módulo PHP con información predefinida que sirve información al documento web (25 horas)

Implementación de funcionalidades extra, como configuración y ubicación manual de elementos (20 horas)

Elaboración del tercer capítulo de la memoria: "Interacción entre usuario e IU" (15 horas)

1.6.3 TERCERA FASE: CAPTACIÓN DE INFORMACION DESDE UN MODULO EXTERNO

En esta fase se desarrollará el módulo encargado de recibir información desde un módulo externo simulado para que interprete la información recibida por este último y la mande al documento web, que cargará los elementos en el mapa en función de esta información.

Está previsto que esta fase, de 47 horas de duración, se inicie el 20 de marzo de 2009 y concluya el 10 de abril.

Preparación del entorno de trabajo de la tercera fase (2 horas)

Implementación de módulo Java que genera información simulada (15 horas).

Implementación de la comunicación entre el módulo Java y el módulo PHP mediante otros módulos (30 horas)

1.6.4 CUARTA FASE: CONEXIÓN CON EL SISTEMA MULTI-ROBOT

En esta fase se conectará el módulo conseguido previamente a un entorno JMS, simulando la comunicación con otros agentes del sistema multi-robot, tanto para enviar como para recibir datos.

Concluida esta fase, deberíamos disponer ya de todo el interfaz y que éste sea funcional. Para ello quizás sea necesario refinar la aplicación y pulir ciertos detalles.

Está previsto que esta fase, que dura 122 horas, se inicie el 10 de abril de 2009 y concluya el 1 de junio de 2009.

Preparación del entorno de trabajo para la cuarta fase (10 horas)

Añadir funcionalidad de comunicación JMS al módulo Java (20 horas)

Implementación de agente simulado que genere información (20 horas)

Conexión entre agentes mediante JMS (10 horas)

Añadir funcionalidad de escucha continua hacia el agente simulado por parte del documento web (30 horas)

Añadir funcionalidad de envío de ordenes hacia el agente simulado por parte del documento web (15 horas)

Elaboración del tercer capítulo de la memoria: "Arquitectura de la IU" (15 horas)

1.6.5 QUINTA FASE: EJEMPLO, CONCLUSIONES Y PRESENTACIÓN

Para acabar, se analizará todo el conjunto del interfaz y los resultados obtenidos, y se mostrarán pruebas elaboradas que apoyen la viabilidad y fiabilidad del sistema.

Por último, se elaborará una presentación para poder defender este Proyecto de Fin de Carrera ante un Tribunal.

Esta fase dura 95 horas y se iniciará el 20 de julio de 2009 y finalizará el 10 de septiembre de 2009.

Recopilación de requerimientos para realizar un ejemplo elaborado de aplicación (5 horas)

Implementación de módulo auxiliar para el ejemplo (15 horas)

Elaboración del ejemplo (15 horas)

Elaboración del sexto capítulo de la memoria: "Un ejemplo práctico" (15 horas)

Elaboración del quinto capítulo de la memoria: "Conclusiones" (15 horas)

Elaboración de la presentación (30 horas)

1.7 ORGANIZACIÓN DE LA MEMORIA

La memoria de este Proyecto de Fin de Carrera estará organizada en seis capítulos, cuyo contenido se resume a continuación:

Capítulo 1: Introducción

El primer capítulo de la memoria nos ha introducido a los objetivos de este proyecto, que nos servirá de referencia para comprender las motivaciones de su desarrollo. También nos dará una idea de las funcionalidades que ofrece esta aplicación, y de dar una visión general de los detalles de la implementación, así como de analizar los costes y riesgos de ésta. Por último conoceremos diversas tecnologías que se ayudan de Google Earth como interfaz para ofrecer al usuario la información que quieren transmitir sobre el mapa.

Capítulo 2: Arquitectura de la aplicación

En este capítulo se analizarán los diferentes módulos que componen la IU, su funcionamiento, la comunicación entre ellos, y cómo se relacionan con el usuario y con otros agentes del sistema.

Capítulo 3: Interfaz de usuario

Veremos como el usuario se relaciona con la IU: ver la información, manipularla, y mandar órdenes a otros agentes del sistema.

Capítulo 4: Caso de uso

Se mostrará un ejemplo elaborado que mostrará las capacidades del interfaz. Además se mostrarán los pasos que sigue la IU al ejecutar este ejemplo, ayudando a reforzar la comprensión del funcionamiento del sistema.

Capítulo 5: Conclusiones

Para finalizar, mostraremos las conclusiones del Proyecto, mostrando y discutiendo los resultados obtenidos. También analizaremos posibles vías de continuación del Proyecto en el futuro.

1.8 ESTADO DEL ARTE

El uso de la información geográfica que ofrecen Google Earth y Google Maps como soporte al desarrollo de aplicaciones aún no está demasiado extendido. Sin embargo, se pueden encontrar aplicaciones realizados bajo estos entornos. A continuación se exponen unos cuantos ejemplos que se unen a los expuestos en el punto 1.2.1, todos ellos extraídos del foro de usuarios oficial de Google Earth. Estos nos darán un nuevo enfoque en cuanto a lo que es posible realizar con estas herramientas.

1.8.1 EJEMPLO DE SUPERPOSICIÓN DE MODELO EN TRES DIMENSIONES

Esta aplicación simplemente es un modelo en tres dimensiones de la Iglesia de San Nicolás, situada en Dnepropetrovsk, Ucrania. Si bien no aporta ningún otro tipo de información, es interesante observar como se superpone encima del mapa original, donde realmente se ubica la iglesia.

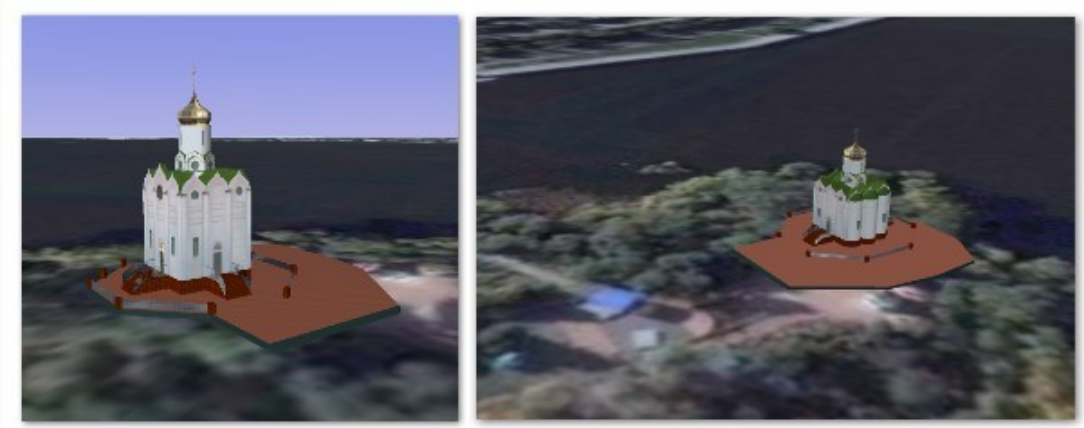

Figura 1.4: Superposición sobre el mapa de un modelo en tres dimensiones representando la Iglesia de San Nicolás.

1.8.2 EJEMPLO DE SUPERPOSICIÓN DE IMÁGENES

El siguiente ejemplo es análogo al anterior, con la diferencia que aquí se superpone una imagen sobre el mapa, y no un modelo en tres dimensiones. Se trata de una fotografía tomada por la NASA de la Isla de Bouvet, situada a medio camino entre el sur de África y la Antártida.

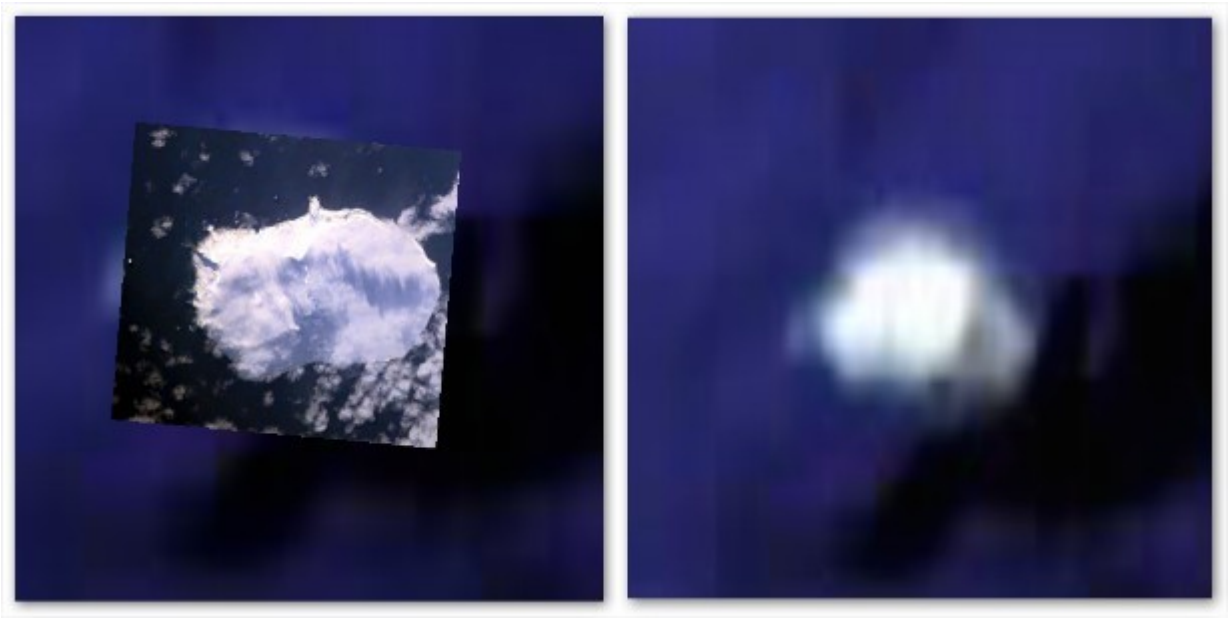

Figura 1.5: A la izquierda, superposición sobre el mapa de una imagen con la fotografía de la Isla de Bouvet. A la derecha, sin superposición

Como podemos observar, la fotografía añade un nivel de detalle superior al que Google Earth ofrece por defecto.

1.8.3 EJEMPLO DE SUPERPOSICIÓN DE MARCAS E INFORMACIÓN HTML

Aquí se nos presenta un buen ejemplo de la superposición de marcas e información HTML. Se trata de un pequeño recorrido por lugares de Bosnia-Herzegovina. Sobre el mapa de despliegan marcas marcando la ubicación de estos lugares. Al clicar sobre ellos, nos aparece un diálogo con información de ese lugar en formato HTML, con texto, imágenes, enlaces, etcétera...

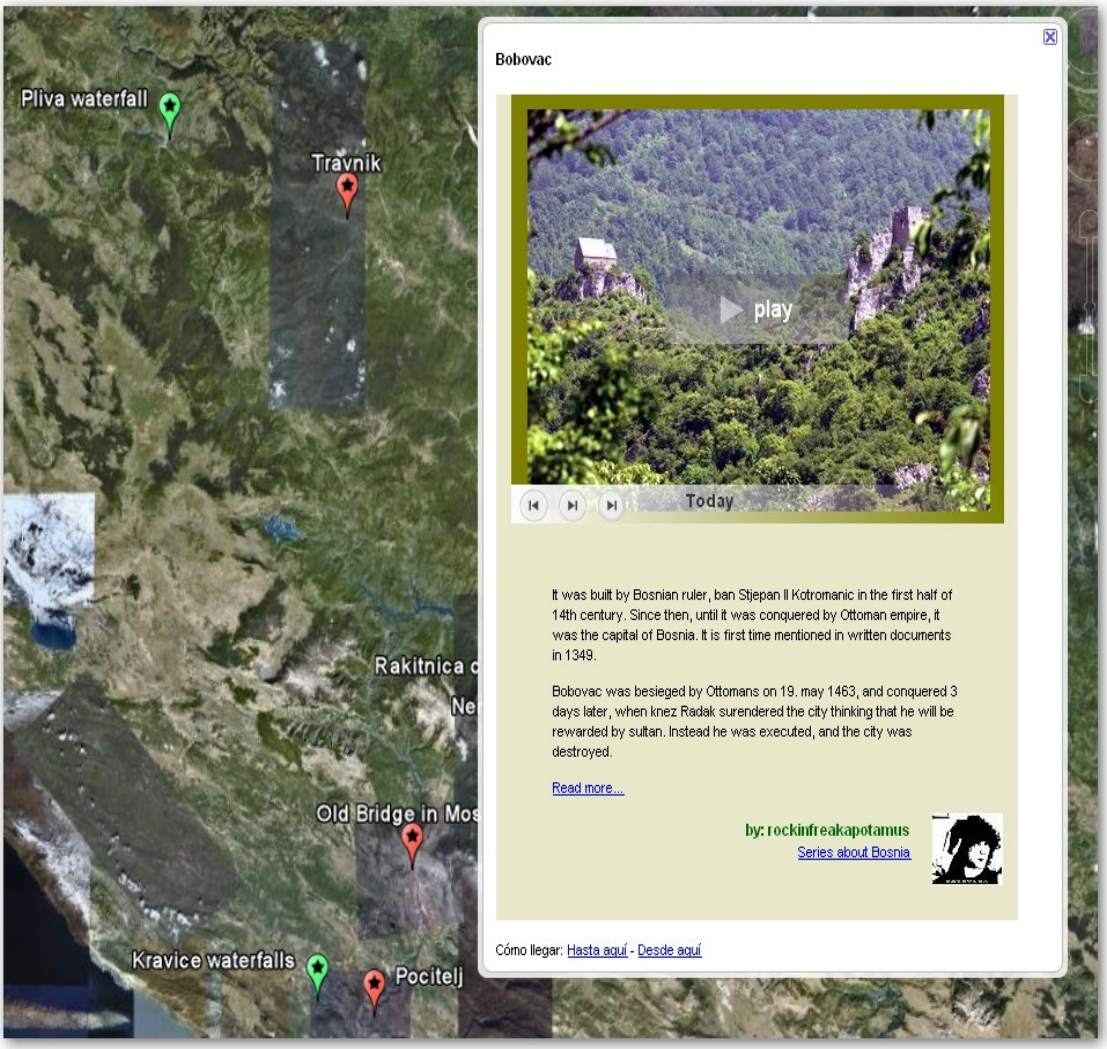

Figura 1.6: Superposición de marcas y contenido HTML con información sobre lugares de Bosnia-Herzegovina

Capítulo 2.

ARQUITECTURA DE LA APLICACIÓN

A la hora de implementar la IU, ésta ha seguido un patrón de diseño de tres capas. Este patrón de diseño se caracteriza por separar la lógica de la presentación, la del negocio y la de datos.

La **capa de presentación** consistirá en la vista que tiene el usuario de la aplicación. A través de ésta el usuario es capaz de interactuar con la IU, ya que se le muestra toda la información geográfica, además de los menús.

La **capa de datos** es la encargada de obtener la información que necesitaremos de otros agentes.

La **capa de negocio** sirve de nexo entre las anteriores: recoge la información de la capa de datos, la procesa y la entrega a la capa de presentación para que sea mostrada.

Los diferentes módulos que conforman nuestra aplicación pueden ser fácilmente catalogados dentro de estas capas. A continuación se explica más detalladamente la función de cada uno de estos módulos, y su analogía con las capas.

Figura 2.1: Programación en tres capas

2.1 MÓDULOS PRINCIPALES

Los módulos principales, como hemos visto, toman el papel de las capas de presentación y datos. Estos módulos se relacionan con el resto de agentes del sistema, y con el usuario. La comunicación entre ellos se realiza mediante los módulos de comunicación, que toman el papel de capa de negocio, y que serán descritos en el punto 2.2.

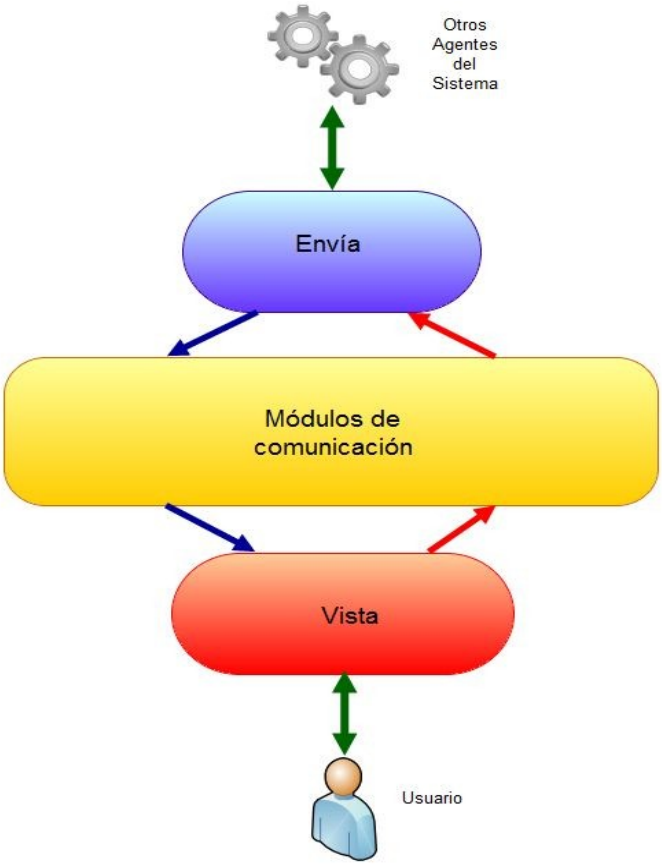

Figura 2.2: Los dos módulos principales comunicándose a través de los módulos de comunicación

La figura 2.2, además de mostrarnos la arquitectura de la IU a grandes rasgos, nos sirve para identificar las capas de las que consta la aplicación: el módulo Vista (en rojo) constituye la capa de presentación, los módulos de comunicación (en amarillo) constituyen la capa de negocio, y el módulo Envía (en azul) constituye la capa de Datos.

2.1.1 EL MÓDULO VISTA

El módulo Vista es aquel con el que el usuario se relaciona directamente. Es el encargado de desplegar al usuario toda la información geográfica de Google Earth, tanto aquella proporcionada por Google como aquella generada por otros agentes. También es el módulo que recibirá directamente las ordenes del usuario, y que serán transmitidas al resto de módulos.

Dado que este módulo será ejecutado sobre un navegador web, las tecnologías utilizadas serán HTML para definir la estructura del documento, Javascript para la manipulación dinámica del documento y la gestión de eventos, y CSS para definir la estética del documento.

En complemento a Javascript se ha utilizado la biblioteca Jquery [9], una biblioteca para Javascript con un conjunto de funciones de propósito general de alto nivel, que facilita entre otras cosas llamadas AJAX [10] a servidores remotos, como se verá más adelante.

En el tercer capítulo de la memoria se da una visión de cómo el usuario debe interactuar con la aplicación, obviando detalles de su implementación o arquitectura.

2.1.1.1 API DE GOOGLE EARTH

Google ofrece junto al plug-in para mostrar Google Earth a través del navegador, una API basada en Javascript para manipular el mapa mostrado. Para usar esta API, deberemos cargar la biblioteca remotamente. Es importante remarcar que Google impone la norma de registrar cada servidor en el que se usará esta API. Esto conllevará tener que alojar el código del módulo de la vista en un servidor web público, es decir, no podremos ejecutarlo en modo local, o tras una intranet privada.

Para usar la API deberemos definir una instancia (o más si nuestra intención fuera mostrar varios mapas en un mismo documento) del mapa, y a partir de ahí llamar a funciones que afectarán a dicha instancia. Podemos dividir el conjunto de las funciones en dos grupos: funciones para manipular la vista, y funciones para definir elementos en el mapa.

Las funciones para manipular la vista nos permitirán cosas tales como modificar la orientación y la altura respecto al suelo de la cámara.

En cuanto a las funciones para definir elementos, son las que más se usan en la implementación de la vista. Cada vez que se quiere añadir un nuevo elemento al mapa, se crea una instancia de éste, y

posteriormente se definen sus atributos. Los atributos dependerán del tipo de elemento que estemos tratando. Por ejemplo, los atributos que definiremos en una marca serán su latitud y longitud, la imagen que usaremos como imagen para insertar en el mapa y su nombre.

También podremos modificar los atributos de un elemento posteriormente para actualizarlos. Como cada elemento lleva asociado un identificador, podremos referirnos a un elemento y aplicarle los mismos métodos que se usaron para definir los atributos de los nuevos elementos.

La API también ofrece la posibilidad de definir eventos, permitiendo la ejecución de ciertas funciones cuando ocurran cosas tales como clicar sobre el mapa, mover el cursor sobre el mapa, etc.. La funcionalidad que se ofrece en el menú Manual está basada en estos eventos: por ejemplo al dibujar manualmente una marca, se capturará la longitud y latitud del punto donde se clicó con el ratón, y se creará una nueva marca con dichas coordenadas como atributos.

2.1.1.2 ARQUITECTURA DE LA INTERFAZ

La estructura de la interfaz consta principalmente de tres regiones en las que se divide la ventana del navegador:

La más importante es la región de **visualización**, donde se despliega el mapa en tres dimensiones para que el usuario visualice la información geográfica. Esta región es manipulada íntegramente mediante la API Javascript de Google Earth: creando una instancia del objeto mapa definida sobre la región especificada. Todo lo relativo a la manipulación del mapa, como inserción y eliminación de elementos, definición de atributos de éstos y posicionamiento de la cámara se realiza utilizando métodos sobre este objeto.

Otra región es la región de **menú**, a través del cual el usuario gobierna la aplicación mediante formularios, botones, enlaces, etcétera... Con el fin de no hacerlo muy denso, se ha dividido éste en submenús, de forma que al clicar en las pestañas se accede a cierta parte de la información, ocultando el resto.

La tercera región es la de **información**, donde aparecen datos sobre los elementos cargados sobre el mapa desde los agentes. Con estos datos podremos realizar acciones con estos elementos, como renombrarlos, borrarlos, enfocarlos con la cámara, etcétera...

Existe una cuarta región cuyo fin, aparte de contener el título, es el de permitir al usuario ocultar las regiones que no son de visualización, de forma que pueda tener una visión más amplia del mapa si lo desea.

Al principio de la implementación de la interfaz se contempló la posibilidad de que se usara una única región dedicada a la visualización que abarcara toda la ventana del navegador, haciendo que los menús aparezcan directamente sobre el mapa, ya sea al clicar sobre un elemento para mostrar sus datos, o en otra parte para mostrar las diferentes opciones del usuario. Las ventajas serían disponer de una área mayor de visualización y dotar al interfaz de una mayor interactividad, ya que para manipular un elemento deberíamos cliclar directamente sobre él. En cuanto a las desventajas, aparte de que al desplegar menús sobre el mapa podría entorpecer la visualización de los elementos, supondría una mayor complejidad en la implementación ya que para mostrar estos menús en el mapa supondría trabajar directamente con elementos de la instancia de Google Earth, comportando más horas de trabajo, en lugar de trabajar separadamente con la programación clásica de un documento HTML y Javascript. Dado que se disponían de 450 horas para la realización del Proyecto de Fin de Carrera, se desestimó esta posibilidad.

2.1.1.3 COMUNICACIÓN CON EL RESTO DE MÓDULOS

Una de las principales razones de ser de esta IU es que pueda ser capaz de recibir periódicamente solicitudes para dibujar nuevos elementos, o para modificar elementos previamente recibidos. Estas solicitudes serán servidas por un módulo externo basado en PHP, llamado "Carto", cuando activemos la escucha de estas solicitudes. El cómo este módulo Carto genera las peticiones será explicado posteriormente en este capítulo.

Para que la vista sepa si hay nueva información disponible, cuando la escucha del menú Cartógrafo esté activada consultará periódicamente a Carto (comunicación por encuesta, o *polling*), y este le responderá tanto en caso afirmativo como negativo. El intervalo temporal entre consulta y consulta será de un segundo.

La consulta por parte de la vista se hace mediante AJAX (Asynchronous Javascript And XML). AJAX no es una tecnología en sí: consiste en realizar llamadas desde el script Javascript a un servidor web mediante el protocolo HTTP **durante** la ejecución del script, es decir, sin detener su ejecución y sin recargar la página. La llamada hará que el servidor responda con cierta información que será tratada por el script.

En nuestro caso el script de la vista hará una llamada AJAX enviando además el instante de la última llamada (timestamp), y el servidor consultará si hay nuevas peticiones, posteriores al timestamp. A continuación responderá con una estructura de datos conteniendo, en caso de que las haya, los elementos a añadir o actualizar y sus atributos. Como ya dijimos cada elemento lleva asociado un identificador. El script comprobará si en el documento ya se encuentra una instancia del elemento con ese identificador. En caso afirmativo se tratará de una modificación, y en caso negativo, habrá que crear una nueva instancia. En cualquier caso, aplicaremos a este elemento los atributos que hemos recibido.

En cuanto a la estructura de datos recibida, se trata de un objeto JSON (JavaScript Object Notation) [11], similar a XML, pero preparado para ser usado fácilmente por Javascript. Esta estructura es enviada siempre, tanto si hay novedades como si no, y además de los elementos contiene el timestamp de la consulta por parte de Carto, y un flag de control indicando el éxito o fracaso de la consulta.

La comunicación implementada en la pestaña Simulador es mucho más sencilla. La llamada AJAX solo se llevará a cabo una vez (no periódicamente) al pulsar el botón deseado, y en dicha llamada se indica que tipo de elemento se quiere cargar. Otro módulo PHP llamado "Simulador" contestará siempre con una estructura JSON conteniendo el elemento. Su propósito es únicamente comprobar que la comunicación es correcta, y ejemplificar el tipo de elementos que podremos cargar.

En cuanto al envío de ordenes por parte del usuario al resto del sistema, se hace de modo parecido al empleado con Simulador: se realiza una llamada AJAX con la orden deseada a un módulo PHP llamado "Robots", que según la orden realizará ciertas acciones involucrando a otros módulos, como se verá.

2.1.2 EL MÓDULO ENVÍA

El módulo Envía es el encargado de interactuar con el resto de agentes del sistema. Este módulo recibe las peticiones ejecutadas por el usuario y las entrega a los agentes, y viceversa.

Este módulo ha sido implementado en lenguaje Java. Esta elección se debe a que el resto de agentes del sistema fueron también escritos en este lenguaje, lo que facilitará a los creadores o administradores de estos agentes la comprensión de su manejo. Además, el código Java se puede ejecutar bajo cualquier sistema operativo capaz de correr la Máquina Virtual de Java, lo que facilita su distribución y portabilidad.

Otro aspecto es que esta IU hace un uso exhaustivo de los protocolos de comunicaciones HTTP y JMS. Java dispone de librerías para implementar estos protocolos.

Sin embargo, es importante mencionar que se podría haber escrito en otro lenguaje que cumpla estas premisas, como por ejemplo C o Python, lenguajes muy utilizados hoy en día.

2.1.2.1 COMUNICACIÓN CON OTROS AGENTES

Todos los agentes del sistema se comunican entre sí mediante el protocolo JMS (Java Message Service). Este protocolo se basa en el paso de mensajes de forma asíncrona, haciendo que no sea imprescindible que emisor y receptor estén presentes simultáneamente en el paso del mensaje: cuando el emisor envía un mensaje, este mensaje se almacena en una cola del servidor JMS, que en un instante inmediato o posterior será recogido por el emisor.

El módulo Envía es el encargado de enviar y recibir mensajes con los agentes del sistema. Al recibir una petición del usuario, mandará un mensaje con la petición al agente deseado. En cuanto a la recepción, el módulo Envía consulta periódicamente (cada segundo) la cola de recepción para ver si ha recibido algún mensaje, y en caso afirmativo lo tratará.

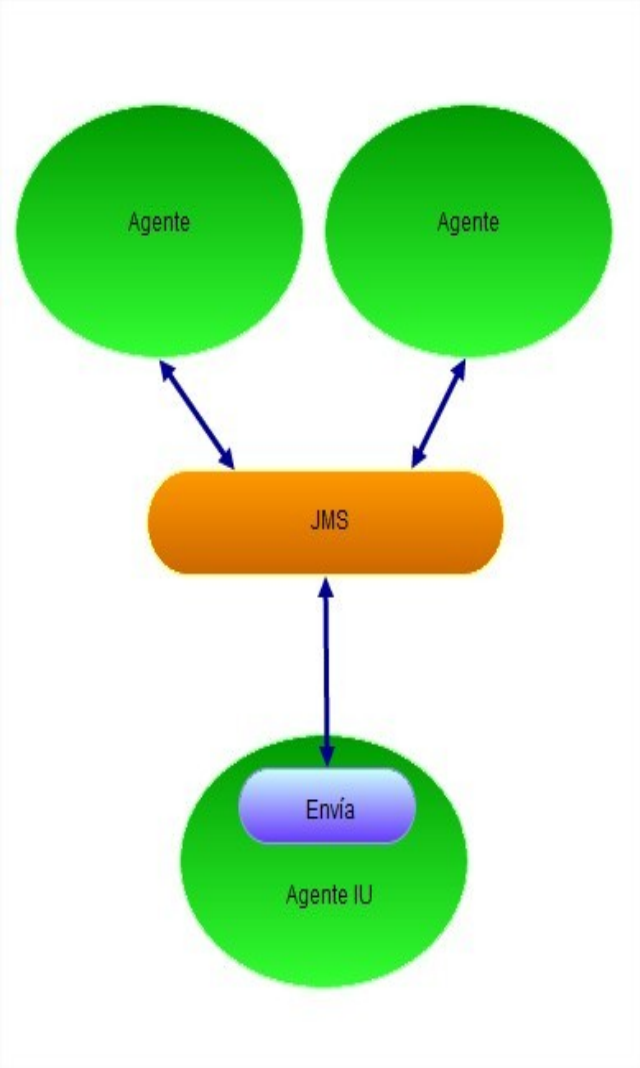

Figura 2.3: Los agentes del sistema, incluído el IU, se comunican entre sí mediante el protocolo JMS

Dado que la IU deberá responder a diferentes acciones, es importante establecer un mecanismo de formato de estos mensajes. Cuando un agente le envíe un mensaje a nuestra IU, este deberá estar formado de la siguiente manera:

OPERACION@ARGUMENTO-1@ARGUMENTO-2@...@ARGUMENTO-N

Como vemos, este mensaje se basa en una cadena de texto compuesto por una operación y un número variable de argumentos, separados por el carácter "@". Al recibir esta cadena de texto, Envía descompondrá este mensaje en campos (partiéndolo en cada ocurrencia del carácter "@"), y leerá el

primero de estos campos, de forma que sabrá de que operación se trata. A partir de aquí leerá los argumentos del resto de campos para llevar a cabo la operación en cuestión.

Una importante limitación de JMS es que la longitud máxima de un mensaje es de 32700 bytes. Dado que habitualmente la IU deberá recibir ficheros de imagen, esta longitud resulta insuficiente. Para resolver este problema, el agente que quiera enviar una imagen deberá partir el contenido del fichero en varios trozos con una longitud máxima de 32700 bytes. Una vez hecho esto, enviará tantos mensajes como trozos han salido, precedidos por un mensaje adicional que le indicará a la IU que se enviará un fichero, y de cuantos trozos está compuesto. Cuando la IU recibe un mensaje de este tipo, interrumpe la consulta periódica de mensajes para recibir seguidamente tantos mensajes como se le indicó. Una vez conseguidos todos, ya puede unir todos los trozos, consiguiendo el fichero completo, y reanudando la consulta habitual.

A la hora de enviar ficheros binarios que no estén basados en texto (como las imágenes), el contenido de estos deberá codificarse en base 64. La razón de usar esta técnica es que no ocasiona problemas de transmisión de caracteres. Esto es importante ya que los ficheros serán transmitidos mediante dos protocolos diferentes (HTTP y JMS).

2.1.2.2 COMUNICACIÓN CON EL MÓDULO VISTA

La forma en la que el módulo Envía envía y recibe información de Vista es prácticamente idéntica a como lo hacía este último, que ya vimos en el punto 2.1.1. El método también consiste en hacer llamadas a módulos PHP mediante el protocolo HTTP.

De la misma forma, para comprobar si hay peticiones del usuario, se consulta periódicamente a un módulo PHP llamado Leer_de_usuario. Esta consulta se hace en la misma iteración en la que se consulta la cola de mensajes JMS, por lo que el periodo sigue siendo de un segundo.

Igualmente, para enviar información al módulo Vista, se hace una llamada a un módulo PHP llamado Master.

2.2 MODULOS DE COMUNICACIÓN

Hemos visto que los módulos principales Vista y Envía necesitan comunicarse entre sí, y que para hacerlo acceden a módulos PHP. En este punto se verá el funcionamiento de estos módulos.

2.2.1 PASO DE MENSAJES

Como se vió en el punto 2.1.2, la comunicación entre los diferentes agentes del sistema se realizaba mediante JMS, logrando una comunicación asíncrona mediante paso de mensajes en una cola. Para implementar esto, necesitaremos configurar un servidor dedicado con un interfaz para el protocolo JMS. En nuestro caso hemos escogido como servidor JMS a OpenJMS [12], que es gratuito y de código abierto.

A priori, podría parecer obvio que nuestros dos módulos principales usaran este mismo protocolo para comunicarse, pero nos encontramos con un problema: los navegadores web no disponen de un interfaz JMS. El protocolo más usado por los navegadores, con diferencia, es HTTP. Así pues, el navegador deberá comunicarse con un servidor web, que sí soporta HTTP. Uno de los servidores web más utilizados es el servidor Apache, que además sirve de plataforma para ejecutar scripts PHP.

Con esa premisa podríamos pensar que una solución sería que el módulo PHP con el que el navegador se comunicará implemente el protocolo JMS para comunicarse con otros módulos. Esto es posible, pero además de la complejidad que supone configurar el servidor Apache para que abra un canal de comunicación con un servidor JMS, presenta un problema: una vez el módulo PHP consuma un mensaje de la cola JMS, no podremos entregarlo al navegador, ya no es el servidor Apache quien se dirige al navegador, sino viceversa, con lo que perderíamos el concepto de asincronía. Este concepto es muy importante en nuestro sistema, ya que el usuario debe ser capaz de visualizar información que se haya producido anteriormente a la ejecución de la IU, sin tener la necesidad de estar conectados en el mismo momento.

Para resolver todo esto, se ha optado por implementar la cola de mensajes en una base de datos MySQL, añadiendo persistencia de datos y por lo tanto otorgando asincronía a nuestro sistema. Escribir en la base de datos simulará guardar un mensaje en la cola, y leer la base de datos simulará extraer el mensaje.

Podríamos definir una analogía con una pizarra: podríamos pensar en una persona A (el módulo que escribe mensajes) que escribe un texto (el mensaje) en una pizarra (la base de datos). Más tarde, una persona B (el módulo que lee mensajes) va a la pizarra y consulta si hay algo escrito en ella.

PHP incorpora una eficiente librería para el manejo de base de datos MySQL, con lo que resulta sencillo leer y escribir esta información.

Resumiendo, se usará LAMP (así es conocido el entorno en que se reúnen Linux, Apache, PHP y MySQL) para implementar el sistema de paso de mensajes: tecnologías gratuitas, sencillas de configurar (las opciones por defecto nos bastarán, sin necesidad de configurar ningún parámetro adicional como tendríamos que hacer para configurar JMS) y enormemente extendidas (es sencillo montar este entorno en una máquina local, y cualquier servicio comercial de hosting minimamente decente lo ofrece). Además, en Apache podremos alojar el documento HTML en el que se basa el módulo Vista.

2.2.2 IMPLEMENTACIÓN DEL SISTEMA DE PASO DE MENSAJES

En nuestra IU tendremos dos colas de mensajes, una para cada canal unidireccional (uno de Envía a Vista, y el segundo viceversa). Cada cola se implementará en una tabla de la base de datos, y cada registro de esa tabla constituirá un mensaje. Estas tablas son Órdenes y Cartógrafo, y pueden ser consideradas módulos de comunicación del sistema, independientes de los módulos PHP. La información asociada a cada mensaje varía en función del canal.

Cuando Vista manda información a Envía, se usa la tabla Órdenes. Cada registro de Órdenes consiste únicamente de un campo con la orden en sí que ha solicitado Vista. Cada vez que Envía solicita al módulo

Leer_De_Usuario que consulte si hay nuevos mensajes, éste hace consulta a la base datos, extrae todos los mensajes que hay, y los devuelve como respuesta a Envía. Acto seguido todos los mensajes en la tabla son borrados, ya que si no lo hiciera, se repetiría la misma orden a cada consulta. Por otro lado, cuando Vista quiere enviar una orden, le pasa ésta al módulo Robot, que se encarga de crear un nuevo registro en Órdenes.

Por otro lado, cuando Envía quiere mandar mensajes a Vista, se usa la tabla Cartógrafo, que contiene tres campos por registro. Aparte del campo con el mensaje en sí, tenemos un campo que contiene un identificador, y otro con una marca de tiempo. El módulo Master se encarga de recibir el mensaje y el identificador asociado a él que Envía quiere mandar, y además obtiene el instante en que este mensaje es mandado. La razón de incluir el identificador y la marca de tiempo se comentará en el siguiente punto.

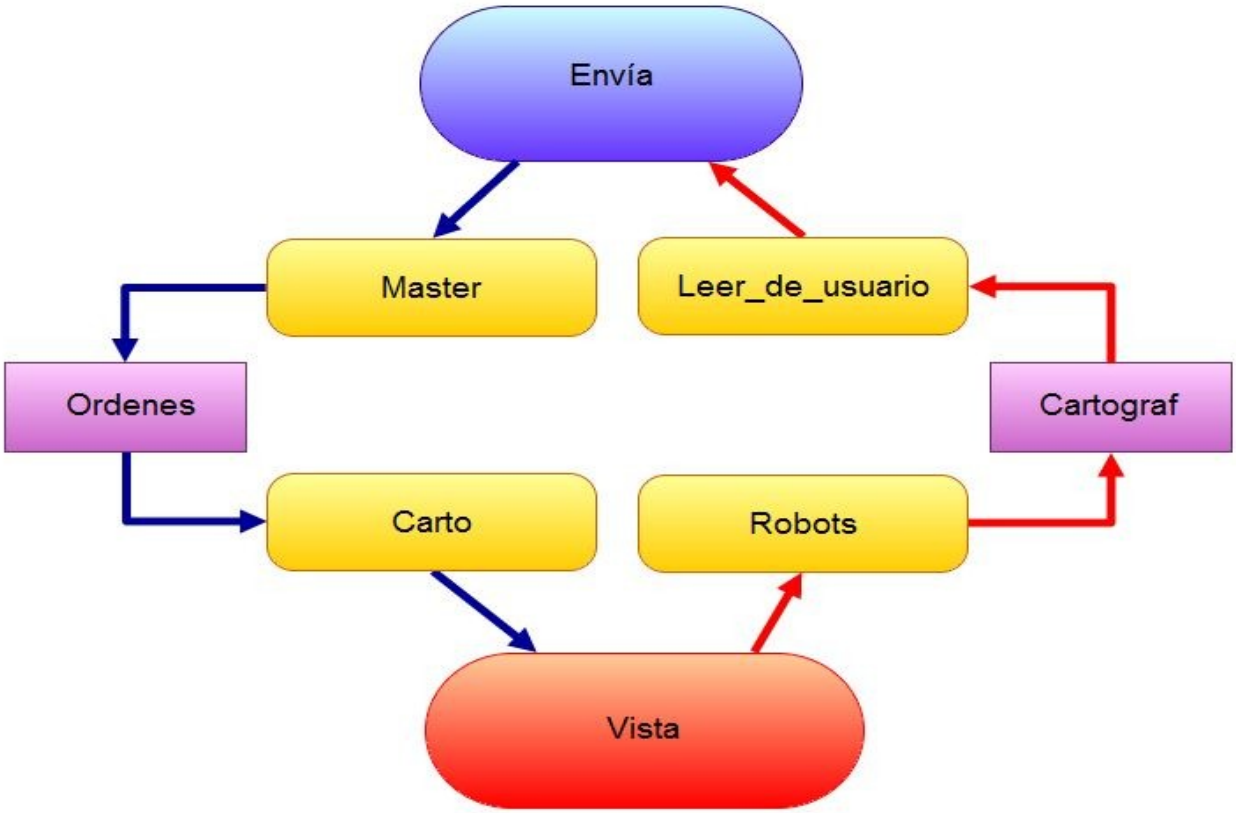

Figura 2.4: Arquitectura de la IU. Los módulos PHP aparecen en color amarillo, y las tablas de las bases de datos en color violeta

2.2.3 INTERPRETACIÓN DE MENSAJES POR PARTE DE VISTA

Los mensajes en la tabla Cartógrafo contienen una representación de los elementos que se querrán insertar o actualizar sobre el mapa del módulo Vista. Los mensajes tienen en el siguiente formato:

TIPO##NOMBRE##ATRIBUTO-1##ATRIBUTO-2##...#ATRIBUTO-N

El uso de este formato es similar al comentado en el punto 2.1.2.1. Al descomponer el mensaje en campos a partir del separador "##", obtenemos como primer campo el tipo de elemento que queremos insertar (marca, imagen, línea...), el nombre con el que el usuario se referirá a este elemento como segundo campo, y los atributos del elemento en el resto de campos.

Si el usuario usa el Simulador para generar elementos predefinidos, podrá observar estos mensajes activando la opción "Get command too".

El módulo PHP Carto, encargado de leer los mensajes de la tabla Cartógrafo cuando Vista se lo solicita, construye la estructura JSON comentada en el punto 2.1.1 con estos campos, el identificador. Este identificador es usado internamente por el módulo Vista para referenciar cada elemento. Cuando Vista reciba la estructura JSON, consultará el identificador, y examinará si ya existe cargado algún elemento con dicho identificador. En caso afirmativo actualizará los atributos de ese elemento, y en caso negativo creará el elemento y le añadirá estos atributos. Es importante no confundir el nombre del elemento con su identificador: el usuario referencia cada elemento por su nombre, y Vista lo hace por su identificador.

Cuando Vista solicita a Carto la consulta de nuevos mensajes, le indica el instante en que se consultó por última vez. Carto solo recupera aquellos mensajes que fueron insertados en la tabla posteriormente a dicho instante. De esta forma, si hay dos usuarios A y B utilizando la IU simultáneamente no se crearán inconsistencias.

Imaginemos que en el instante inicial *i* el usuario A accede a la IU, y en el instante *i+1* recibe hay una actualización en la tabla, que será tratada por su instancia de Vista. En el instante *i+2* el usuario B accede y recibe la información de la tabla, pero el usuario A no tiene necesidad de recargar el elemento, ya que no ha sufrido variaciones desde que lo cargó.

Cuando Carto recibe la solicitud de lectura de mensajes, guarda también el instante actual y lo incluye en la estructura JSON. Esto servirá para que Vista actualice su marca de tiempo con el instante de la última consulta.

2.3 DISTRIBUCION DE LOS MÓDULOS

Una de las ventajas de la arquitectura del IU es que cada módulo puede ser ejecutado en máquinas diferentes si las circunstancias lo requieren, siempre que entre ellas se puedan comunicar mediante el protocolo HTTP.

Unicamente existe una excepción: una de las restricciones de las llamadas AJAX es que las llamadas que realiza el navegador establecidas en el documento HTML deben ser a otros documentos o scripts que residan en el mismo sistema de ficheros. Por lo tanto el módulo Vista y los módulos con los que se comunica directamente mediante HTTP (Carto y Robot) deben residir en el mismo servidor.

Es importante remarcar que los módulos integrantes del agente IU se comunican entre ellos mediante el protocolo HTTP, mientras que los agentes del sistema, incluida la IU, lo hacen mediante JMS.

Capítulo 3.

INTERFAZ DE USUARIO

En este capítulo se analizará cómo el usuario se relaciona con la IU, cómo se visualiza la información solicitada, y cómo es capaz de mandar instrucciones al resto de agentes del sistema. La principal característica de la entrada/salida por parte del usuario es que todo se realiza a través de un navegador web.

3.1 REQUERIMIENTOS EN EL TERMINAL DE USUARIO

Todo el software necesario para que el usuario pueda interactuar con la IU es totalmente gratuito, y en el futuro será posible ejecutarlo en todos los sistemas operativos más comunes, aunque a fecha de escribir esta memoria las versiones estables de la mayoría estos productos solo corren bajo Microsoft Windows.

A continuación se detallan las tres aplicaciones que deberemos instalar en nuestra máquina.

3.1.1 NAVEGADOR WEB

El usuario llevará a cabo todas las acciones a través de un navegador web. La razón fundamental de esta elección es que hoy en día se está apostando muy fuerte por las tecnologías basadas en la comunicación web, desarrollándose aplicaciones cada vez más complejas y capaces de rivalizar con las aplicaciones de escritorio tradicionales. Ésto a su vez ha ocasionado que los navegadores web se hayan convertido en un software cada vez más avanzado y de más calidad, convirtiéndolos en potentes plataformas para la ejecución de estas aplicaciones, como es este Proyecto de Fin de Carrera.

Los navegadores web usados para testear la aplicación han sido Mozilla Firefox y Google Chrome en sus respectivas versiones estables. Es importante comentar que la aplicación **no es capaz** de correr bajo ninguna versión de Microsoft Internet Explorer, ya que el incumplimiento de los estandares Javascript y CSS por parte de este navegador ocasiona problemas de usabilidad con la interfaz.

3.1.2 GOOGLE EARTH

La aplicación Google Earth, ya comentada en anteriores capítulos, es la encargada de manejar toda la información geográfica. A través de ella cargaremos esta información guardada en los servidores de Google, y la podremos renderizar con su motor 3D.

Obviamente, para que Google Earth recoja la información de los servidores de Google, nuestra maquina deberá disponer de acceso a Internet.

3.1.3 PLUG-IN PARA GOOGLE EARTH

Se trata de un plug-in que nos permite renderizar la información de Google Earth en el navegador web. No sólo podremos visualizar los datos geográficos en 3D tal y como los veríamos en Google Earth, sino que además se dispone de una API que nos permitirá interactuar dinámicamente con esta información desde el propio navegador mediante Javascript.

3.2 VISTA DE LA APLICACIÓN

En este punto se explicará las diferentes maneras que el usuario tienes para interactuar con la IU, es decir, mostraremos la interfaz.

Al iniciar la aplicación, veremos que el documento está dividido en tres regiones: **mapa**, **información** y **menú.** Si en algún momento necesitaramos aumentar el tamaño del mapa para tener una mayor perspectiva de éste, podremos ocultar las otras regiones. A continuación se hará una descripción de cada una de estas regiones.

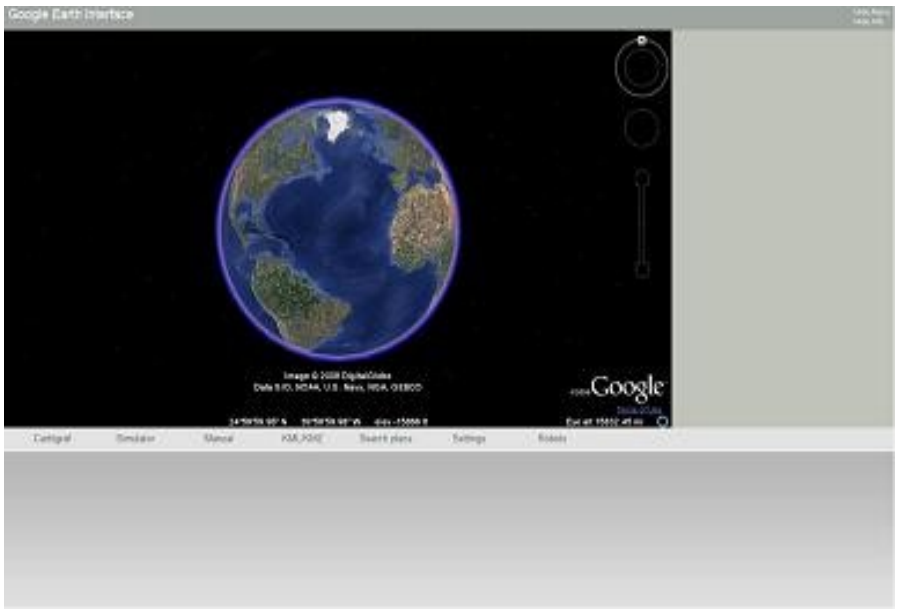

Figura 3.1: Vista inicial de la aplicación

3.2.1 MAPA

En el mapa se renderiza la salida generada por Google Earth, y como en éste, podremos desplazarnos a través de él. Podemos mover en el espacio manteniendo pulsado el botón izquierdo del ratón al ubicar el puntero sobre la región del mapa. También podemos girar la cámara manteniendo pulsado el botón derecho, y modificar el valor del zoom con el botón central.

Además de la información por defecto que ofrece Google Earth, sobre el mapa también se dibujarán otros elementos que cargará la IU. Estos elementos son **marcas** en un punto del mapa, **imágenes** que podrán superponerse al suelo del mapa, **líneas** entre dos puntos, **polígonos** rellenos y **modelos en tres dimensiones** que representarán estructuras.

3.2.2 INFORMACIÓN

Tal como se comentaba en el anterior punto, la IU cargará los elementos adicionales que suministrarán otros agentes del sistema. Dado que estos elementos serán los más consultados por el usuario, en la región lateral derecha del documento se añadirán los nombres de estos elementos.

Al clicar sobre el nombre de uno de estos elementos, nos aparecerán diversas opciones para manipular estos elementos, como renombrarlos, eliminarlos, hacer que la cámara los enfoque, o en el caso de las marcas, moverlas arrastrándolas sobre el mapa.

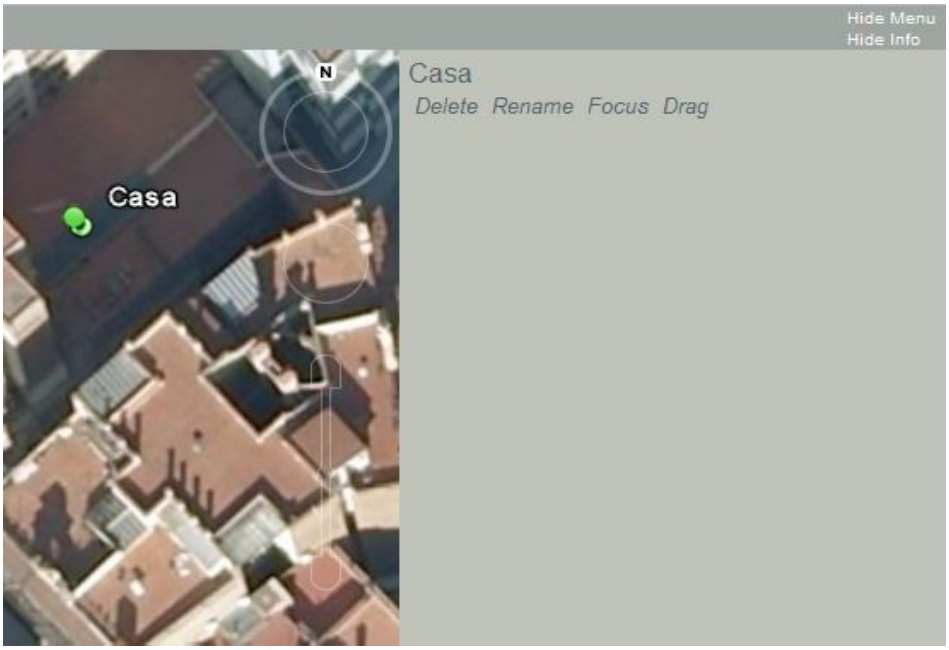

Figura 3.2: Información sobre los elementos cargados

3.2.3 MENÚ

En la región inferior del documento se encuentra el menú, que nos permite llevar a cabo diferentes acciones de interacción con la IU. Las más relevantes tienen que ver con el envío de órdenes a los otros módulos que conforman la IU, como de recepción de peticiones para añadir y actualizar elementos del mapa.

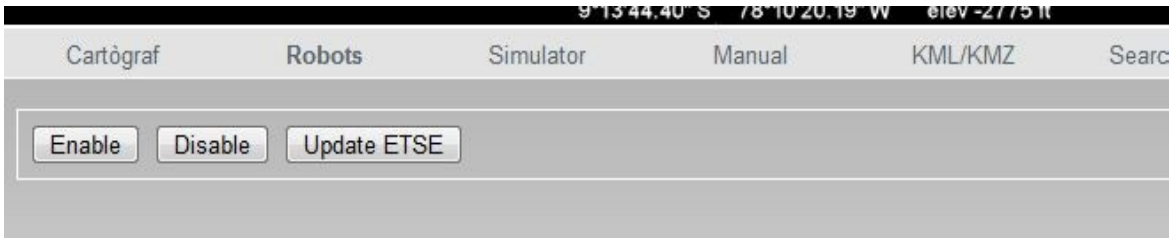

Figura 3.3: Menú de la interfaz

El menú está dispuesto en forma de pestañas, de forma que al clicar en una de ellas aparecerán diversas acciones relacionadas que el usuario podrá seleccionar.

En la pestaña **Cartógrafo** podremos activar y desactivar la la recepción de comandos para la adición o modificación de elementos del mapa. Mientras esté activada, cada segundo se comprobará la existencia de nuevos peticiones de actualización. En caso afirmativo, además de la actualización del mapa, se mostrará un mensaje detallando cual ha sido la actualización, y el instante en que se ha producido.

En la pestaña **Robots** podremos enviar órdenes a otros módulos de la IU, que a su vez enviarán peticiones a otros agentes del sistema.

En la pestaña **Simulator** tiene el fin de mostrar los diferentes tipos de elementos que la IU es capaz de mostrar. Al igual que la comunicación anterior, se envía una petición periódicamente a otro módulo indicando el tipo de elemento que se quiere cargar, pero este módulo simplemente genera un nuevo comando siempre según la petición que se hizo. Su uso es básicamente de prueba de la aplicación.

En la pestaña **Manual** el usuario podrá "dibujar" directamente sobre el mapa nuevas marcas, lineas y polígonos. Esta opción puede resultar útil para que el usuario establezca sus propias referencias.

En la pestaña **KML/KMZ** el usuario puede cargar ficheros KML/KMZ. Estos ficheros contienen información geográfica que es insertada sobre el mapa. Sin embargo, esta técnica no es la usada para cargar elementos desde los agentes de nuestra aplicación.

En la pestaña **Search Place** el usuario podrá escribir el nombre de una localización y la cámara la enfocará (si existe).

En la pestaña **Settings** podremos definir ciertos aspectos del mapa, como las referencias de escala, longitud y latitud de la cámara, controles de navegación, mapas políticos, aparición de relieve y de edificios por defecto mostrados en Google Earth.

Capítulo 4.

CASO DE PRUEBA

En este capítulo se muestra una aplicación concreta del agente de relación con el usuario en un caso de sistema de ayuda a la asistencia de las víctimas en edificios bien por incendio, fuga de gases nocivos, terremotos o cualquier otra circunstancia. En primer lugar se presenta de forma más detallada el problema que se pretende resolver con esta aplicación y luego se describe como se ha probado dicha aplicación, con notables simplificaciones que, aun así, no deja de ser ilustrativa para comprender el dominio de la aplicación.

4.1 DESCRIPCIÓN DE LA APLICACIÓN

Google Earth ofrece datos geográficos que pueden mezclarse con otros de tipo cartográfico y así ofrecer caminos de acceso los edificios. Sin embargo, dichos datos no están actualizados ni incluyen los mismos edificios (aunque ya hay algunos edificios significativos de los que sí se dispone de datos para visitas virtuales). Por ello es necesario un reconocimiento in situ mediante un conjunto de exploradores que permitan cartografiar el área de forma detallada y actualizada, cosa que permite una actuación más eficiente de los servicios de atención a la emergencia que haya ocurrido en un momento dado.

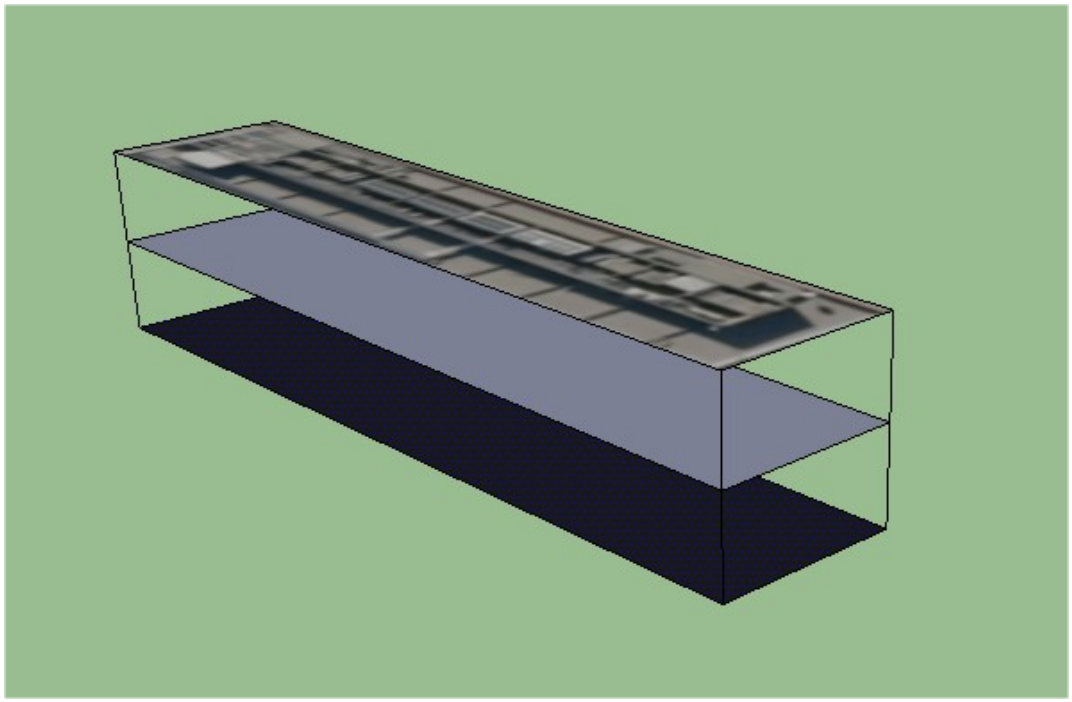

Figura 4.1: Simple modelo en 3D usado para la representación de la ETSE

En este ejemplo se simulará el reconocimiento por parte de los exploradores de uno de los pisos de un edificio, en este caso la Escola Tècnica i Superior d'Enginyeria (ETSE). Para que el usuario sea capaz de visualizar el piso explorado, se usará un simple modelo en tres dimensiones representando el edificio. Éste está formado simplemente por tres bases representando al suelo de los pisos, eliminando las paredes para poder observar su interior.

El objetivo de la aplicación es que en la base del piso intermedio (de color azul grisáceo en la figura 4.1) se dibujen los datos conseguidos por los robots exploradores. Además, también se colocarán pequeñas marcas que indicaran la posición de los robots.

El usuario podrá navegar a través de él con el ratón para analizar las partes del piso que más le interesen. El contenido será actualizado periódicamente con los nuevos datos que consigan los robots exploradores.

A su vez este modelo en tres dimensiones será ubicado sobre el mapa de Google Earth encima de la posición donde realmente está la ETSE. De esta forma se consigue visualizar todo aquello que hay alrededor del edificio, según la documentación cartográfica previamente documentada por Google.

4.2 SIMPLIFICACIONES

Dado que el propósito de este caso de uso es mostrar a modo de demostración la viabilidad del proyecto, se simularán los datos que generan los exploradores con un agente de prueba que entregará imágenes predefinidas y posiciones de los robots aleatorias. Este agente de prueba solo generará actualizaciones cuando se le ordene de forma explícita, a través de la IU.

Se supone que los datos del edificio en cuestión están en una base de datos disponible para el servicio de atención a emergencias, ya que no se dispone de ningún sistema que genere automáticamente el modelado del edificio.

4.3 DETALLES DE LA IMPLEMENTACIÓN

A continuación se explican los detalles de la implementación de este caso de prueba. Este punto mostrará al lector todos que se llevan a cabo a la hora de mostrar sobre los mapas, lo cual servirá de apoyo para la comprensión de la arquitectura de la IU.

4.3.1 AGENTE DE PRUEBA

Para la simulación se ha implementado un agente de prueba (a partir de ahora AP) que genera una serie de imágenes predefinidas que simulan mapas generados por el sistema real, además de posiciones aleatorias para unos cuantos robots. El AP ha sido escrito en Java y es capaz de conectarse al servidor JMS de la misma forma que se conectarían los agentes reales del sistema, con lo que es capaz de comunicarse con nuestro agente IU y mandarle la imagen y las coordenadas con las posiciones de los robots.

Para que el AP sea capaz de operar el usuario deberá activarlo previamente. Como vimos en el tercer capítulo, para ello el usuario deberá pulsar sobre el botón "Enable" en la pestaña "Robots" del menú. Así nuestra petición llegará al módulo Envía vía HTTP, y a continuación este módulo pondrá el mensaje con la petición en la cola JMS, que será leído por el agente de prueba, haciendo que se su estado sea "activo".

Una vez activado, si el usuario envía la orden de actualizar la ETSE ("Update ETSE") el AP generará la imagen en formato BMP y las posiciones de los robots. La imagen se encuentra en el sistema de ficheros de la máquina donde se ejecuta el AP, con lo que el agente la abrirá y, como vimos en el punto 2.1.2.1, codificará su contenido a base 64 y lo dividirá en trozos de 32700 bytes.

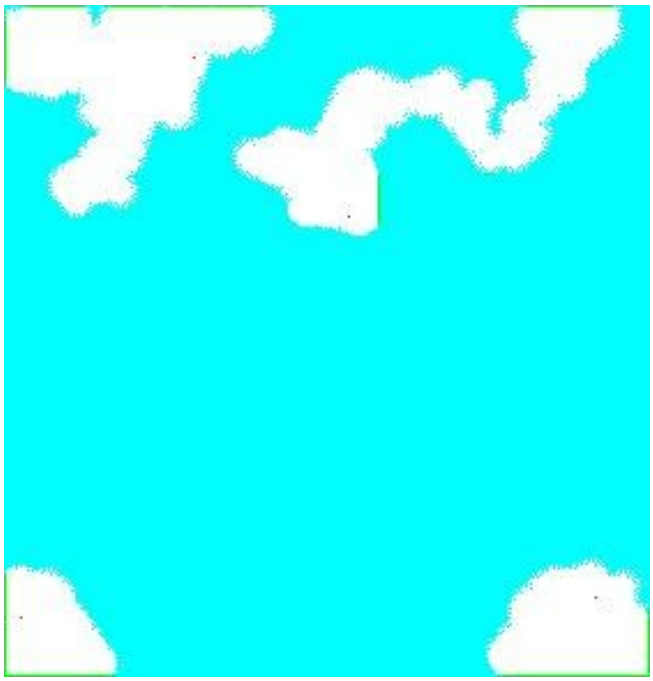

Figura 4.2: Ejemplo de imagen generada por el Agente de Prueba

A continuación se preparará el mensaje inicial que enviaremos a la IU. Este es una cadena de texto que aglutina las coordenadas de los robots, el área que abarcará la imagen sobre el piso intermedio de la ETSE, la posición de la imagen respecto a la esquina superior derecha del piso (ver el punto 4.3.2 para una explicación más detallada de los conceptos de área y posición respecto al piso), y el número de trozos en que ha sido dividido el fichero de la imagen. Una vez enviado este mensaje, enviaremos los trozos del contenido del fichero.

A medida que AP va colocando los mensajes en la cola JMS, el módulo Envía del agente IU los irá recogiendo. Del mensaje inicial guardará las posiciones de los robots y las dimensiones de la imagen, y leerá el número de trozos que componen el fichero de la imagen. Así recogerá de la cola ese número de mensajes, unirá el contenido y obtendrá el contenido completo del fichero de imagen, aún codificado en base 64.

Llegados a este punto, el módulo Envía ya tiene en su poder toda la información que le ha enviado AP.

4.3.2 CONSTRUCCIÓN DEL MODELO EN 3D

Para que Google Earth pueda cargar un modelo en 3D, deberemos indicarle que lea un fichero COLLADA [13], que contiene toda la información del modelo. Los ficheros COLLADA son ficheros con una estructura XML en que sus campos indican posiciones, caras, dimensiones, etc... El problema de cara a nuestro ejemplo es que estos ficheros son estáticos y no varían, cuestión que choca con nuestra idea, ya que este deberá actualizarse con los nuevos mapas que genere el AP.

Como se comentó al comienzo de este capítulo, lo único que modificaremos será la textura del piso intermedio. Dado que el fichero COLLADA referencia a los ficheros que usará como textura, cada vez que queramos actualizar el elemento modificaremos únicamente este fichero de imagen con los datos proporcionados por AP. De esta forma al recargar el elemento ETSE sobre el mapa, se llamará de nuevo al fichero COLLADA, y éste a su vez volverá a cargar las texturas, de las cuales la del piso intermedio habrá sido modificada.

Para construir esta imagen se ha implementado un script PHP llamado etse.php, que recibirá de Envía mediante el protocolo HTTP la información transmitida por AP. Es importante dejar claro que este script no constituye un nuevo módulo del agente, sino que es un método auxiliar para llevar a cabo este ejemplo. Su único propósito es construir la textura del modelo de la ETSE a partir de la imagen y de las coordenadas de los robots transmitidas por el módulo Envía.

Hay que recordar que el fichero de la imagen entregado por Envía estaba codificado en base 64, así que el script deberá decodificarlo antes de poder trabajar con él. La textura del modelo consiste inicialmente en una imagen de 712x157 píxeles de color rojo. A esta imagen el script le añadirá la imagen generada por AP. Dado que la simulación consiste en que los robots analicen parte del piso de la ETSE, la imagen generada solo ocupará parte de la textura, de ahí la necesidad de definir sus dimensiones y la coordenadas relativas a la imagen donde se sitúa. También añadirá unos pequeños cuadrados negros sobre la imagen para representar a los robots sobre la textura. Finalmente la imagen resultante se convertirá a formato JPEG, haciendo que ocupe menos espacio en disco, y así no saturar la red cuando el módulo Vista cargue el modelo en 3D.

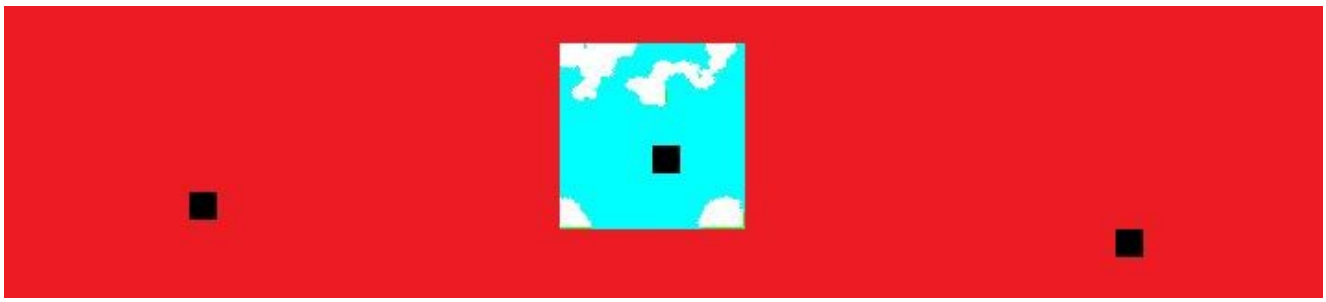

Figura 4.3: Textura resultante generada por el script etse.php

4.3.3 CARGA DEL MODELO

Cuando el script etse.php haya finalizado su labor de actualizar la textura, y con ello el modelo en 3D, responderá a Envía con un mensaje. Hecho esto ya podrá decirle a Vista que cargue el modelo en el mapa, de la forma habitual comentada en el punto 2.1.1: se le enviará un mensaje especificando que el tipo de elemento es un modelo en 3D, su nombre es "ETSE" y sus atributos son la orientación, la escala, las coordenadas y, cómo no, la ruta al fichero COLLADA.

Es importante notar que aunque la textura vaya variando, no lo hará este mensaje, ya que el fichero COLLADA no se modifica: lo que se modifica es la imagen de textura a la que el fichero COLLADA referencia.

Una vez cargado el modelo en el mapa, la cámara enfocará sobre él desde una perspectiva cenital, como se muestra en la figura 4.4.

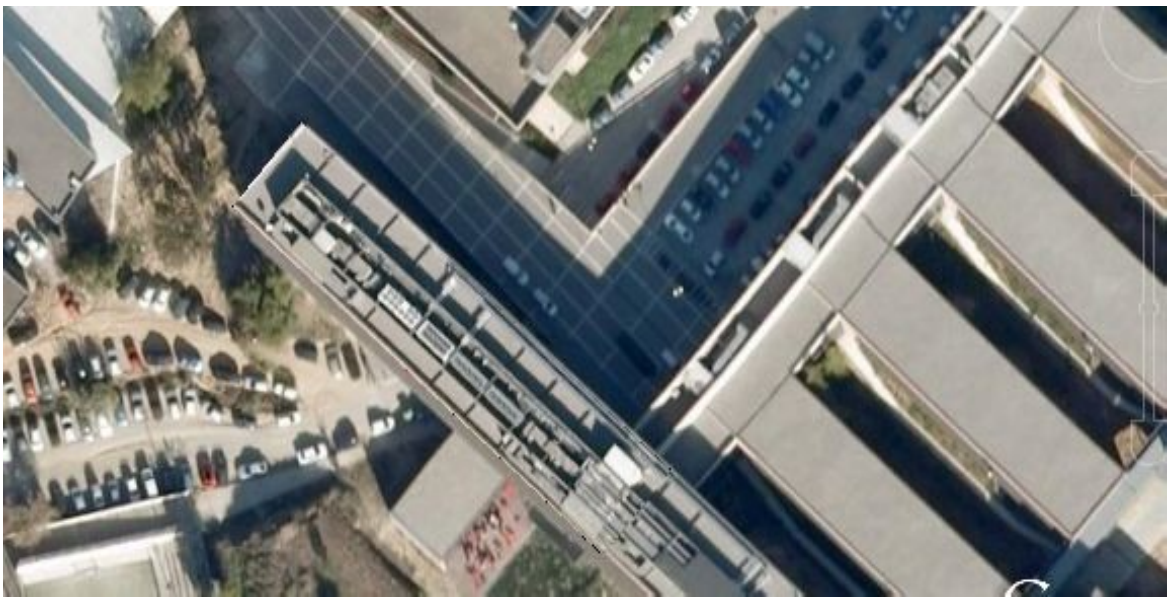

Figura 4.4: Modelo de la ETSE sobre la información de Google Earth vista desde la perspectiva inicial

Ya que como textura del tejado se ha empleado una captura de la propia ETSE vista por defecto por Google Earth, da la sensación que nada se ha cargado. Sin embargo, al rotar la cámara, vemos la estructura del edificio en 3D, y como la textura del piso intermedio se ha cargado en el modelo, como vemos en la figura 4.4.

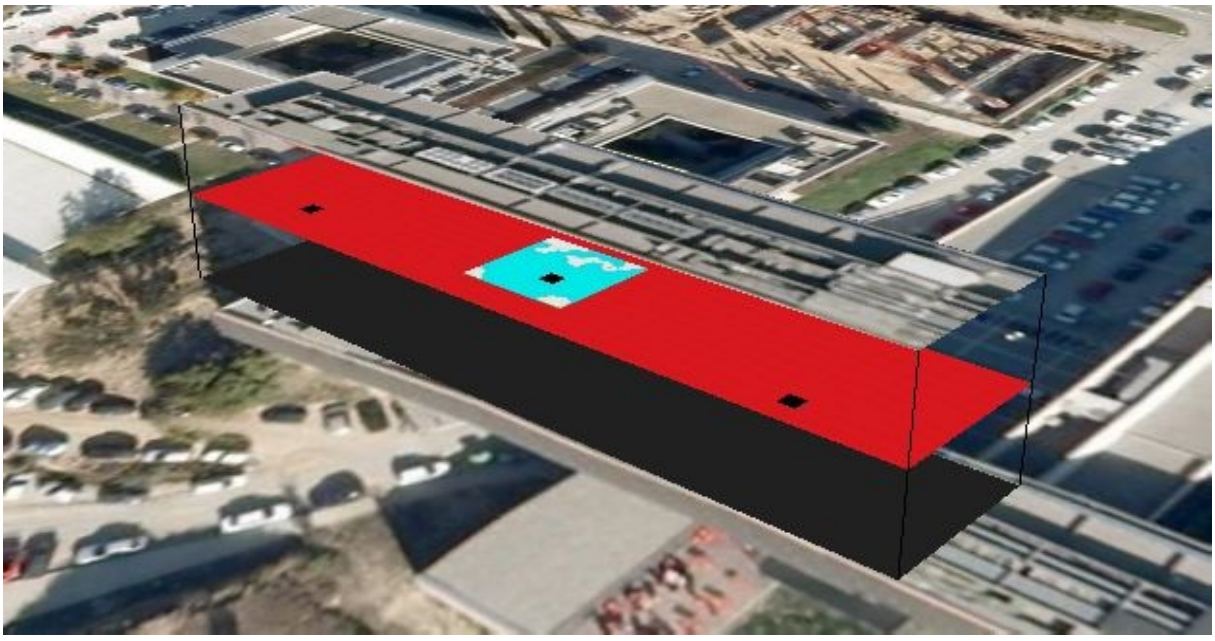

Figura 4.5: Modelo de la ETSE sobre la información de Google Earth vista desde otra perspectiva

En caso de que la información que genera el AP variase, como que la imagen fuera modificada o las coordenadas de los robots cambiasen, esta sería transmitida de nuevo al AP, repitiéndose el proceso de generación de la textura. Una vez modificada, el modelo será actualizado sobre el mapa.

Capítulo 5.

CONCLUSIONES

Finalmente en este capítulo se revisará el objetivo establecido en el primer capítulo de esta memoria, comprobando que ha sido cumplido tal y como se especificó. Además, se propondrán diferentes líneas de continuación, que de llevarse a cabo podrían llegar a incrementar las prestaciones de la interfaz, así como su usabilidad.

5.1 REVISIÓN DE LOS OBJETIVOS

En el punto 1.5 se establecieron las características que debía poseer la IU, y que con el hecho de cumplirlas se podría considerar como conseguido el objetivo principal de este Proyecto Final de Carrera. A continuación revisaremos estas características, y se discutirá si han sido cumplidas.

5.1.1 OBJETIVOS PRINCIPALES

5.1.1.1 COMUNICACIÓN CON OTROS AGENTES

Como se ha comentado en varias ocasiones anteriormente, los agentes que componen el sistema multi-robot se comunican entre ellos, utilizando para ello el protocolo JMS: los agentes se suscriben a colas de mensajes, depositando en ellos los mensajes que quieran enviar, y a su vez consultando estas colas para recoger los mensajes de los que son destinatarios.

La IU, al ser otro agente del sistema, debe ser capaz de subscribirse a estas colas y pasarse mensajes con el resto de agentes. Esta función es llevada a cabo por el módulo Envía del IU, que implementa, entre otras cosas, una interfaz para utilizar el protocolo JMS, y por tanto, una forma de establecer comunicación con estos agentes.

Además, cada vez el usuario desea mandar órdenes al resto de agentes del sistema, éstas son enviadas al módulo Envía, y este módulo retransmite las ordenes a los otros agentes también mediante el protocolo JMS.

Por lo tanto, se demuestra la comunicación entre agentes, incluyendo al usuario que visualiza la información geográfica y manda peticiones a los otros agentes.

5.1.1.2 MOSTRAR DATOS SOBRE LA INFORMACIÓN GEOGRÁFICA DE GOOGLE EARTH

La IU debe ser capaz de mostrar la información recibida por otras agentes sobre la información que suministra Google Earth. Esta premisa se cumple, ya que la IU recibe la información geográfica del agente Cartógrafo y tras procesarla hace que el motor de Google Earth añada los elementos sobre los mapas, de forma que el usuario observa en el navegador ambas informaciones simultáneamente.

Además los elementos se actualizan periódicamente, y los cambios que estos sufren se ven plasmados sin necesidad de recargar la página, gracias a la técnica AJAX.

Los diferentes tipos de elementos que podemos visualizar son marcas, imágenes, polígonos sólidos, lineas y modelos en tres dimensiones.

5.1.2 OBJETIVOS SECUNDARIOS

5.1.2.1 NAVEGADOR WEB COMO PLATAFORMA PARA LA IU

Tal y como se proponía, el usuario interactúa con la IU a través de un navegador web. Al probar la aplicación ésta se ha ejecutado correctamente en los navegadores Mozilla Firefox y Google Chrome. Sin embargo, en el caso del navegador Microsoft Internet Explorer, la aplicación no se ha ejecutado correctamente, debido a que en este navegador se incumplen los estándares de programación orientada a web.

Esto supone un problema a los usuarios habituales de este navegador, que son mayoritarios en el mercado, no sólo para esta IU en concreto, sino también para cualquier otro tipo de aplicación web.

Sin embargo dedicando más horas de programación podríamos llegar a solventar este problema. Dado que el propósito de este proyecto no era que corriera bajo un navegador concreto, no se ha avanzado por ese camino.

Otra motivación para usar el navegador web era eliminar la necesidad de instalar software adicional. Esto no se cumple del todo, ya que es necesario disponer del cliente de Google Earth y un plug-in que redirija la salida del cliente hacia el navegador, aunque el usuario no necesitará tener conocimientos de su uso, con instalarlo será suficiente. Por suerte, este software está disponible para los sistemas operativos y navegadores más comunes.

Por lo tanto se podría decir que este objetivo ha sido cumplido pero con restricciones.

5.1.2.2 MÓDULOS DISTRIBUIDOS REMOTAMENTE

Como se vio en el punto 2.3 los módulos que conforman la IU pueden ser distribuidos en diferentes máquinas. En la implementación de la IU se han utilizado tres máquinas diferentes: dos PCs domésticos en uno de los cuales se ha ejecutado la vista de la IU a través del navegador web, y otra en la que se han ejecutado los módulos implementados en Java (el módulo Envía y el agente externo de prueba) y un servidor remoto de alojamiento web en los que se han ubicado los módulos PHP encargados de las comunicaciones, y las bases de datos MySQL.

Igualmente se podría haber optado por otra distribución en función de las necesidades, como por ejemplo alojar cada módulo en una máquina diferente, o aglutinar todos los módulos en una única máquina. La única restricción sería que haya comunicación mediante JMS entre agentes diferentes, y comunicación HTTP entre los módulos de la IU.

Otra restricción, como se vio en el punto 2.1.1.1, es que la máquina donde se aloje el módulo Vista debe poseer un servidor web de acceso público por Internet, dada la imposición de Google de registrar las aplicaciones que hagan uso de la API de Google Earth.

5.2 LINEAS DE CONTINUACIÓN

Una vez visto el anterior capítulo, podemos afirmar que el objetivo del Proyecto de Final de Carrera ha sido conseguido. Sin embargo, podemos imaginar unas cuantas lineas de continuación que podrían incrementar la viabilidad de este proyecto, o incluso derivar en otras ramas de investigación. A continuación se exponen un par de ideas de proyectos que respaldarían la eficiencia de la IU basada en Google Earth.

5.2.1 SISTEMA DE ESCANEO PARA LA GENERACIÓN DE FICHEROS COLLADA

A pesar del notable realismo que se puede conseguir con la representación sobre Google Earth de modelos en tres dimensiones mediante los ficheros COLLADA, éstos nos obligan a que sea un diseñador quien defina la forma de estos, dejando el nivel de realismo conseguido en manos de la habilidad del diseñador.

Por otra parte, los ficheros COLLADA son siempre estáticos y no varían su estado, de forma que es relativamente complicado redefinir sus atributos. En el ejemplo práctico expuesto en el capítulo 4 hicimos servir un pequeño "truco" para modificar el modelo que representaba el edificio de la ETSE, consistente en modificar únicamente la imagen que hacía las veces de textura para el modelo.

Dadas estas circunstancias, nos sería de gran utilidad un sistema que fuera capaz de construir ficheros COLLADA a partir del escaneo de la estructura que se desee modelar. De esta manera sería posible conseguir modelos mucho más elaborados, realistas y actualizados.

5.2.2 REDUCCIÓN DE LOS TIEMPOS DE CARGA

Como hemos visto en anteriores capítulos, el intervalo de refresco de información obtenida de un agente externo era de un segundo. El peso en bytes de la información obtenida es bajo ya que generalmente solo descargamos los atributos de los modelos, haciendo que un segundo sea un intervalo suficiente para asegurar la consecución correcta de la información.

Sin embargo, podemos encontrarnos con situaciones en que la cantidad de información sea drásticamente superior, como es el caso de imágenes o modelos en tres dimensiones con alto nivel de detalle, haciendo que el tiempo pueda ser insuficiente para descargar los ficheros, de imagen o COLLADA

respectivamente. Además los módulos que conforman la UI pueden estar distribuidos en diferentes sistemas remotos, lo que podría conllevar retrasos debidos a la congestión de la red.

También podría suceder que el problema que abordemos requiera una mayor frecuencia de refresco, como por ejemplo en sistemas de tiempo real, lo que haría que el intervalo de un segundo sea totalmente insuficiente.

Es por esto que sería muy interesante implementar técnicas que reduzcan estos tiempos de carga, como podrían ser compactación de la información, cacheo de ficheros para no hacer necesario recargar cierta información, etcétera...

REFERENCIAS

[1] **Google Earth.**

http://earth.google.es/

[2] **BlueBots: proyectos fin de carrera con robots controlados mediante bluetooth.** Lluís Ribas-Xirgo, 2009

[3] **Normativa de Projectes de Fi de Carrera d'Enginyeria Informàtica.**

Junta de Secció de l'Escola Tècnica i Superior D'Enginyeria, 2009

[4] **Bluebots: Robots como periféricos Bluetooth. Agente cartógrafo.**

Sergio Luis Benítez Rodríguez, 2008

[5] **Java Message Service (JMS).**

http://java.sun.com/products/jms/

[6] **Introducción a JMS (Java Message Service)**

http://www.programacion.com/java/articulo/jms/

[7] **¿Qué es Java?**

http://java.com/es/download/whatis_java.jsp

[8] **PHP: Hypertext Preprocessor**

http://www.php.net/

[9] **jQuery: The Write Less, Do More, Javascript Library**

http://jquery.com/

[10] **Introducción a AJAX**

http://www.librosweb.es/ajax/

[11] **Introducción a JSON**

http://www.json.org/json-es.html

[12] OpenJMS

http://openjms.sourceforge.net/

[13] COLLADA – Digital Asset and FX Exchange Schema

https://collada.org/mediawiki/index.php/COLLADA__Digital_Asset_and_FX_Exchange_Schema

Carles Pagès Rozas,

16 de septiembre de 2009

Els sistemes multi-robot de reconeixement de superfícies es poden utilitzar tant per a l'exploració de llocs remots, de difícil accés o perillosos. Habitualment, els robots no són autònoms, depenen d'operadors humans per dirigir-los. La informació que capten ha de ser processada i mostrada a l'usuari o usuària del sistema de forma intel·ligible.

Un exemple d'aplicació seria el d'un sistema multi-robot format per diversos helicòpters no tripulats que proporciona informació d'una àrea que ha patit algun desastre. El sistema informàtic recolliria la informació i la transmetria al coordinador de l'operatiu d'assistència de l'emergència.

La idea del projecte és la de combinar la informació proporcionada pel sistema multi-robot amb la de la zona disponible a Google Earth i fer d'aquesta eina l'interfície d'usuari de l'aplicació.