Escuela Técnica Superior de Ingeniería Geodésica, Cartográfica y Topográfica

Universidad Politécnica de Valencia

Visor 3D basado en Google Earth para visita virtual

Caso de estudio: La Puebla de Farnals

PROYECTO FINAL DE CARRERA

Autor: Alejandro Fernández Oliver

Director: Jesús Manuel Palomar Vázquez

RESUMEN

Este Proyecto Final de Carrera consiste en la creación de una aplicación Web de exploración virtual tridimensional con el propósito de dar a conocer mejor la zona, y permitir la consulta de los recursos turísticos y comerciales localizados en el ámbito del casco urbano de la Playa de la Puebla de Farnals (Valencia).

Para ello se incorpora en la página Web una ventana que muestra un modelo tridimensional de la Tierra mediante la API de *Google Earth* y sobre el cual se han situado geográficamente modelos 3D de diversos edificios pertenecientes a la zona de estudio. Dichos modelos han sido creados a través de la aplicación *Google SketchUp*.

Por otro lado, la página Web permite hacer una serie de consultas acerca de los comercios situados en la zona de estudio. Permite ver la situación de los mismos, los modelos de los edificios en los que están situados y una ventana de información que aporta datos adicionales sobre el comercio en cuestión. Por otro lado se han programado una serie de paseos virtuales automatizados que pueden ser elegidos mediante un menú y con los que es posible visitar diversas zonas del modelo.

Para todo esto se dará una introducción a la realidad virtual y a los mundos digitales 3D. Más adelante se presentaran las diferentes formas existentes para el modelado 3D y se entrara al detalle con la que se eligió para este proyecto, que fue *Google SketchUp*.

A continuación se presenta la aplicación bajo la que van a mostrarse los modelos generados y su API correspondiente, que permite hacer esto mismo dentro de una página Web. Después se da una descripción y comentario del código generado para la misma.

Por último se aportan una serie de ideas y líneas de trabajo que puedan abrir las puertas a nuevos proyectos así como mejorar y ampliar el conjunto de la aplicación aquí presentada.

A mis padres.

Reconocimientos:

Quiero dar mi más sincero agradecimiento a D. Jesús Palomar Vázquez, por haberme dirigido este Proyecto Final de Carrera, por el tiempo empleado conmigo y por su apoyo, gracias a lo cual ha sido posible la realización de este proyecto.

Al personal de las empresas que conforman la Asociación de Comerciantes Virgen del Carmen de la Playa de la Puebla de Farnals, por el trato recibido y por facilitar la información de sus comercios.

A Matías Lafuente, por ceder un espacio en su servidor de TECES.net en el que albergar la aplicación Web presentada en este proyecto.

A Marcos Sanz, por sus sabios consejos y recomendaciones de estilo.

Y finalmente, agradecer la comprensión de mis familiares y amigos, que han sido los que han tenido que aguantar los malos momentos y demás circunstancias a lo largo de la elaboración de este proyecto.

A todos muchas gracias.

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Presentación del problema

Hoy en día Internet supone la mayor de las ventanas al mundo que nos rodea. Constituye el principal medio a través del que cualquier persona o cosa puede ser dada a conocer y su aplicación en el ámbito de la cartografía era sólo cuestión de tiempo. Pese a que en un principio muchos servicios en Internet eran de pago, estos han ido abriéndose al público general de forma gratuita, permitiendo que fueran los propios usuarios los que aportaran nuevos contenidos, como es el caso principal de *Google Earth* [78]. Gracias a esta aplicación es posible seguir en la actualidad la evolución de la mancha de fuel derramada en las costas del Golfo de México [82], y ha servido para ayudar a localizar víctimas en los terremotos de Chile y Haití [83].

Dada la característica de *Google Earth* de poder visualizar modelos de edificios 3D, ha sido posible que numerosos usuarios creen modelos de edificios emblemáticos existentes en la actualidad y los compartan con la comunidad de usuarios. Ejemplo de ello es Anxo Miján, un estudiante gallego de arquitectura técnica que ha modelado numerosos edificios de Galicia, como el Cuartel de Instrucción en Ferrol, la Plaza de Galicia en la misma ciudad o la Torre de Hércules en La Coruña, entre otros [45]. Otra aplicación muy extendida es mostrar edificios que serán construidos en un futuro para así tener una idea de qué aspecto tendrán. Un ejemplo destacado es el modelo 3D del Museo y Monumento Nacional del 11 de Septiembre en Nueva York, de próxima construcción [144].

Por último, dado que es posible programar *tours* o vuelos en esta aplicación, se ha hecho recientemente muy popular hacer recorridos por zonas extensas de esta manera. Un caso muy interesante lo tenemos en la red iberoamericana de bosques modelo (RIABM), que tiene previsto elaborar un *tour* virtual que recorrerá todos los espacios que integran su red [64]. Esto, junto con un inventario y una plataforma informativa sobre los recursos presentes en la red, permitirá el fomento del turismo en esta zona.

1.2. Objetivos del proyecto

Académicos

El objetivo del Proyecto Final de Carrera es comprobar la capacidad que el alumno ha ido desarrollando a lo largo de los años de estudio para analizar, proyectar y resolver cualquier asunto relativo al área profesional de la titulación.

Es por eso que, una vez aprobadas todas las asignaturas que conforman los planes de estudio, me dispongo a entregar, exponer y defender este Proyecto Final de Carrera, con el objetivo de obtener la titulación de Ingeniero Técnico en Topografía.

Específicos

La Playa de la Puebla de Farnals es una de las playas más concurridas al norte de Valencia. Esto es debido principalmente a su cercanía a la capital, a sus comunicaciones y sus servicios. No obstante, mucha gente que acude a pasar el día a la playa desconoce en gran parte la zona. Normalmente se concentra en la zona más comercial, que es también la más cercana a la playa. Allí se pueden encontrar comercios muy conocidos, como las pizzerías Napoli y Toni, o el restaurante Orly, que llevan ya muchos años abiertos. Por otro lado, otros comercios, algo más apartados de la playa quedan descartados por los visitantes al no estar tan cerca de la zona comercial. Para poder promocionarse, algunos comercios publican anuncios en revistas locales, o se dan a conocer mediante anuncios en radio o Páginas Amarillas. Muchos otros se publicitan con el boca-a-boca.

A esto hay que añadir que hasta hace unos años, muchas de las calles de la Playa de la Puebla de Farnals, pese a tener nombre, no tenían placas que las designaran. Mucha gente desconocía el nombre de las calles y sabía dónde estaban los comercios y servicios a base de conocer la zona. Tampoco había planos en las calles para poder ubicarse. Las referencias que se tomaban y se toman aún hoy en día son los nombres de los complejos deportivos. Tanto es así que muchos comercios prescindían de poner la dirección y en su lugar indicaban su posición por el nombre del complejo deportivo y el del edificio en el que se encontraban. Hoy en día las calles ya están señalizadas y hay varios planos zonales en diversos puntos de la zona. Aún así, mucha gente residente sigue desconociendo el nombre de algunas calles. Por ello cuando algún visitante pregunta por este o aquel comercio o complejo, no es raro ver dar indicaciones de aquella manera.

Este Proyecto pretende aliviar la situación con la siguiente propuesta:

En caso de que una persona estuviera interesada en visitar la zona para pasar el día o una temporada, podría consultar, por ejemplo, *Google Earth*. Allí podría encontrar la foto aérea de la zona y un callejero. Dada la colaboración entre Google y Páginas Amarillas dispondría también de información adicional sobre algunos comercios, aunque ésta es más bien escueta. Esto último es debido a que el servicio de Páginas

Amarillas es de pago y muchos comercios no aparecen reflejados. Además la presentación de dicha información podría tener un mejor aspecto.

Es por eso que el objetivo de este Proyecto Final de Carrera es la realización de una aplicación Web de exploración virtual tridimensional, con el propósito de dar a conocer mejor la zona, fomentar el turismo y permitir la consulta de los servicios localizados en el ámbito del casco urbano de la Playa de la Puebla de Farnals (Valencia).

Para ello se considera que, dadas las tecnologías actuales en materia de cartografía digital en Internet y el interés creciente en esta área por parte de los usuarios, una buena forma de promocionar la zona de estudio es hacer un modelo 3D de la misma, a modo de maqueta digital, que pueda ser mostrada a todo el mundo. Para ello se aprovecha la aplicación *Google SketchUp* para el modelado 3D.

Los edificios de la zona deben mostrarse de la forma más realista que sea posible con los medios disponibles, para que el usuario tenga la sensación de estar moviéndose por dicho entorno. A su vez, para completar la información sobre los complejos deportivos, deben incluirse las instalaciones que cada uno tenga, esto es: pistas de tenis, frontón, padel, piscinas, parques, paseos, jardines, zonas de recreo, etc.

Deben identificarse los diferentes complejos, mediante su nombre. De esta manera el visitante será capaz de interpretar las indicaciones habladas de orientación.

Todos los comercios de la zona de estudio deben aparecer señalados en la aplicación de manera que sea fácil localizarlos. A su vez debe buscarse la forma de aportar información complementaria, tanto alfanumérica como visual, siendo ésta accesible desde el propio modelo 3D mostrado en *Google Earth*.

Aprovechando la salida al mercado de la API de *Google Earth* se creará una página Web que contenga todo lo hecho hasta este punto. El interfaz de usuario ofrecerá la posibilidad de realizar consultas sobre los comercios y un menú con una serie de rutas o vuelos que muestren diversas partes de la zona de estudio de forma automática.

De esta forma, cualquier persona que quiera informarse sobre lo que puede encontrar en la Playa de la Puebla de Farnals, podrá dar un paseo virtual por toda la zona, conocer qué servicios y comercios va a encontrar, su localización e información detallada sobre los mismos, incluso enlaces a sus páginas Web. Todo ello en un modelo digital 3D realista, donde podrá contemplar los diversos complejos con sus pistas deportivas, piscinas, zonas de recreo, etc.

1.3. Solución propuesta

La principal intención del proyecto, como se ha visto en el apartado anterior, es poder publicarlo en una página Web, para que sea accesible a todo el mundo. Para ello, en un principio se pensó en manejar aplicaciones de modelado de mundos virtuales mediante VRML, lenguaje orientado precisamente a este cometido. Pero el aumento de la popularidad de *Google Earth* en el momento de empezar el proyecto llevó a decidir que esta podría ser una mejor forma de mostrar el trabajo. Además la inmediata aparición de la API de *Google Earth* [86], que permite mostrar una ventana de esta aplicación dentro de una página Web, hizo decantarse definitivamente por este camino. De esta manera había dos formas de mostrar el trabajo, bien mediante la aplicación independiente o bien a través de una página Web.

Una vez decidido esto, y dado que proviene del mismo vendedor, la herramienta de modelado que mejor se integra con *Google Earth* es *SketchUp*. Como se verá más adelante, la decisión de modelar toda la maqueta digital mediante esta aplicación ha sido correcta. Es un programa muy intuitivo, fácil de usar y de rápido aprendizaje. Además permite el uso de fotografías como texturas, con lo que el acabado de los modelos es bastante realista.

Para señalar la localización de los comercios se utilizará *Google Earth*, por su facilidad de uso. Además dada su sencillez en la edición se maquetarán ventanas de información que amplíen los datos existentes de todos los comercios.

La creación de una página Web que integre *Google Earth* y muestre toda la información generada es ideal para la promoción de la zona, dado que, como se expone más arriba, estará al acceso de todo el mundo. Ésta mostrará unos modelos realistas y aportará información importante que el usuario pueda necesitar. Además, la inclusión de unas sencillas herramientas de consulta para poder localizar comercios concretos resultará muy útil. Por otra parte, disponer de una sección para poder elegir diversos vuelos por la zona resultará práctico para poder conocer cómodamente las partes representativas del modelo y ver detalles curiosos de varios de los complejos. Por último, la colocación de una sección de gestión de capas permitirá que el usuario muestre u oculte en la vista determinados tipos de comercios a su antojo.

1.4. Estructura de la memoria del proyecto

La memoria que describe este Proyecto Final de Carrera sigue la siguiente estructura:

Capítulo 2. Se muestra el estado del arte en realidad virtual y las técnicas asociadas a ella. Se presenta después VRML, un lenguaje para la creación de mundos virtuales, y su evolución actual, X3D. Por último se explica qué son los modelos 3D y qué métodos de representación, técnicas de modelado y procesos de renderizado existen hoy en día.

Capítulo 3. Se describe someramente el flujo de trabajo que se ha seguido en la elaboración de este proyecto. Después se describe la ubicación de la zona de estudio. Y por último se detallan las fuentes de información que han permitido el desarrollo de cada una de las fases.

Capítulo 4. Contiene una descripción detallada de la implementación de la solución propuesta, siguiendo el flujo de trabajo paso a paso. Así pues se explican los procesos de modelado 3D, captura y tratamiento de imágenes, búsqueda y tratamiento de la información y por último la elaboración de la aplicación Web.

Capítulo 5. Se muestra el presupuesto correspondiente a los trabajos realizados en la totalidad del proyecto. Aparece desglosado en capítulos y apartados según las diferentes fases de trabajo.

Capítulo 6. Se presentan las conclusiones sobre el trabajo realizado así como líneas para trabajos futuros.

Capítulo 7. Contiene la bibliografía y los enlaces de Internet que han servido de documentación de apoyo y consulta para la elaboración de esta memoria.

Capítulo 8. Se muestran imágenes de los modelos 3D de todas las urbanizaciones que componen la zona de estudio y que han sido modelados durante la elaboración de este Proyecto Final de Carrera.

Capítulo 2

Estado del arte

2.1. Introducción

En este capítulo se trata el estado del arte de la realidad virtual y de las tecnologías asociadas a ella. Se muestra en qué consiste, sus orígenes y qué aplicaciones tiene, que van desde el entretenimiento hasta su uso terapéutico.

A continuación se presenta VRML, un lenguaje de generación de mundos virtuales, y su evolución actual, X3D, ambos orientados a su uso en la Web. Se presentan sus características y las diferentes aplicaciones existentes para la edición y visualización de mundos virtuales.

En la última parte se explica qué son los modelos 3D y cómo se modelan. Para ello se explican los métodos de representación existentes hoy en día, las técnicas de modelado y el proceso de renderizado.

2.2. Realidad virtual

2.2.1. Definición

La realidad virtual puede definirse como "*una tecnología que permite a un usuario ver y manipular los contenidos de un entorno tridimensional simulado mediante un ordenador*", Stephen Matsuba [19].

2.2.2. Historia

Los conceptos básicos relativos a la realidad virtual se desarrollaron en los años 50 del siglo XX, y los primeros sistemas empezaron a fabricarse poco después.

Al final de los años 50 Morton Heilig diseñó *Sensorama* [122], un sistema que combinaba audio, video, vibración, viento y olores, proporcionando al usuario una experiencia virtual multisensorial. Dado que los gráficos generados por ordenador quedaban aún lejos, toda la experiencia estaba pregrabada.

Figura 2.1. Sistema Sensorama

En los años 60 Heilig creó un dispositivo que supondría la base de esta industria y que siempre ha estado relacionado con los sistemas inmersivos de realidad virtual; la pantalla montada en la cabeza o casco virtual.

Varios años más tarde, Ivan Sutherland, un especialista en gráficos generados por ordenador, creó “la pantalla definitiva”. Se trataba de un sistema que registraba los movimientos de la cabeza del usuario. Estos se veían reflejados en la imagen mostrada en la pantalla, de manera que el usuario se sentía inmerso en el mundo generado por el ordenador. De esta forma se creó el primer sistema de realidad virtual mediante ordenador.

En los años 70 la NASA empezó a experimentar con la realidad virtual. Construyó un casco virtual sencillo con un sistema de rastreo, ambos conectados a un ordenador. Incluso crearon un guante virtual con el que el usuario podía tocar y manejar objetos del entorno virtual. Se trataba de una "simulación generada por ordenador de un entorno 3D, en el cual el usuario era libre de ver y manejar los contenidos de dicho entorno". En otras palabras, el primer sistema de realidad virtual de verdad.

2.2.3. Características

La realidad virtual es dinámica y está en constante cambio. Los objetos que contiene pueden moverse, cambiar y responder a la interacción humana.

El usuario no sólo ve la información, sino que se mueve por ella, puede tocarla y manejarla.

En la mayoría de los casos se simulan mundos imaginarios, dado su amplio uso en el sector del entretenimiento, principalmente en videojuegos. No obstante, en muchas ocasiones se intenta que la simulación del mundo virtual sea lo más fiel posible a la realidad. Esto se consigue en gran medida en los simuladores para entrenamiento de pilotos de combate. En estos se simulan todas las condiciones y situaciones en las que pueden verse envueltos, aprendiendo así a desenvolverse y resolverlas [198].

Figura 2.2. Sistema de entrenamiento de vuelo mediante realidad virtual

La calidad de la experiencia en la realidad virtual depende de las limitaciones técnicas, i.e., velocidad de proceso, resolución de imagen y ancho de banda. Por tanto, conforme vayan apareciendo procesadores más rápidos, tarjetas aceleradoras gráficas más potentes, y mayores anchos de banda, mejor será la experiencia.

2.2.4. Dispositivos

Hoy en día la mayor parte de las experiencias en realidad virtual son visuales, ya sea en la pantalla de un ordenador o mediante cascos de realidad virtual. Dichos cascos disponen de un par de pantallas, cada una con la información visual adaptada a cada ojo, proporcionando así una sensación de inmersión mayor. A su vez, para acentuar un poco más la sensación de tridimensionalidad, se suele añadir sonidos estereoscópicos mediante auriculares.

Por otro lado, aunque menos extendida, es posible añadir información háptica (relativa al tacto y a las fuerzas). De esta manera diferentes aplicaciones, principalmente médicas y también algunos videojuegos, registran los movimientos del usuario y le devuelven como respuesta diferentes fuerzas, vibraciones y movimientos. Un ejemplo sencillo lo encontraríamos en los videojuegos de simulación de coches [162]. En ellos unos volantes especiales transmiten al usuario la misma sensación que tendrían al conducir un coche real (dureza en la dirección, tirada en curva, baches, etc.)

La forma más básica para moverse por un entorno virtual es mediante teclado y ratón. No obstante existen dispositivos, mencionados anteriormente, que están diseñados específicamente para experimentar una mayor inmersión en dichos entornos, por ejemplo, guantes, brazos o cascos virtuales o volantes, entre otros [69].

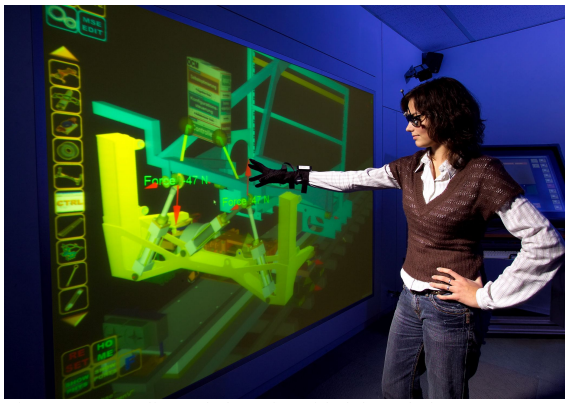


Figura 2.3. Estudiante de la Universidad de Paderborn utilizando la realidad virtual como instrumento de diseño

Figura 2.4. Cabina de piloto de coche

2.2.5. Usos y aplicaciones

El uso de la realidad virtual en el sector del entretenimiento es muy extendido. No obstante existen otros sectores en los que está presente.

Patrimonio y Arqueología

Existen lugares del patrimonio que a menudo son inaccesibles al público general o que incluso ya no existen [159]. Cuevas, entornos naturales, viejas ciudades, monumentos, esculturas, etc., pueden ser reconstruidos con precisión mediante realidad virtual [53].

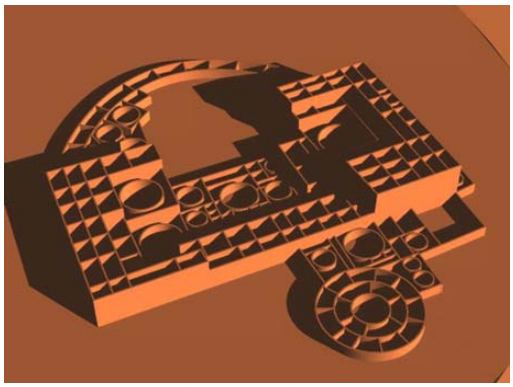


Figura 2.5. Pueblo del Arroyo, Chaco, Nuevo México. Gentileza de Dennis Holloway, arquitecto

Figura 2.6. Reproducción digital de la figura principal de la fuente de Neptuno, Piazza della Signoria, Florencia

Tiene un enorme potencial para su uso en museos como visitas virtuales.

No obstante, dada la dificultad técnica de ofrecer la experiencia en tiempo real a un grupo numeroso, se ha optado en la mayoría de los casos por mostrar las reconstrucciones en un formato pregrabado en una pantalla normal compartida, limitándose así la interacción del usuario con la realidad virtual.

El primer uso de un entorno de realidad virtual aplicado al patrimonio tuvo lugar en 1994 en el Museo del Castillo Dudley [117]. Allí los visitantes pudieron dar un paseo por una reconstrucción del edificio, tal y como era en 1550.

Figura 2.7. Foto real del Castillo de Dudley en 1994

Figura 2.8. Reproducción digital del Castillo de Dudley tal como era en 1550

Cine

La película TRON (1982) de Steven Lisberger fue la primera película en explorar la idea. La película Johnny Mnemonic (1995) de Robert Longo usa ampliamente la realidad virtual, mostrando a un ciber-mensajero que pasa información de contrabando en su cerebro. Y probablemente, la película más famosa que popularizó el tema de la realidad virtual es The Matrix (1999) de Andy y Larry Wachowski. Ésta presentaba la realidad virtual y la vida real a menudo superpuestas y en ocasiones indistinguibles.

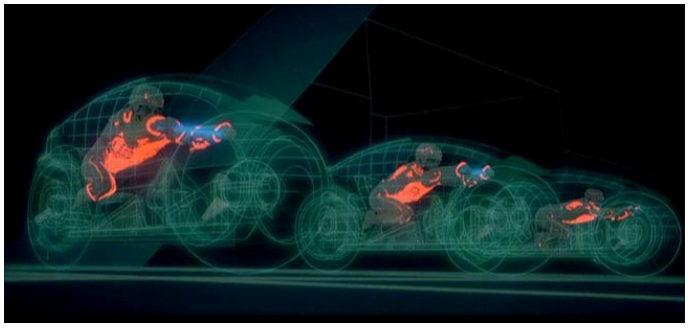


Figura 2.9. Fotograma de la película TRON (1982)

Figura 2.10. Fotograma de la película The Matrix (1999)

Por último es de obligada mención la película Avatar (2009) de James Cameron, no sólo por utilizar ampliamente la realidad virtual para recrear un completo mundo real en el que se desenvuelven los personajes, sino por ser la primera película comercialmente viable que dispone de una versión en 3D, en la que, mediante el uso de un tipo de anaglifos, es posible experimentar una inmersión nunca antes vista en la película. Esto ha provocado que, dado su éxito, numerosas productoras se animen a experimentar en este campo, viéndose así estimulado el uso de la realidad virtual.

Figura 2.11. Fotograma de la película Avatar (2009)

Videojuegos

El primer uso de realidad virtual con videojuegos tuvo lugar en 1991, cuando la compañía *Virtuality* permitió que los ordenadores *Amiga 3000* trabajaran con sus máquinas de realidad virtual. De esta manera surgió el sistema de videojuegos *1000CS*. En este el jugador permanecía de pie, con un casco de realidad virtual, unos auriculares y un joystick 3D. En las figuras 2.12~2.14 pueden verse varios de sus juegos.

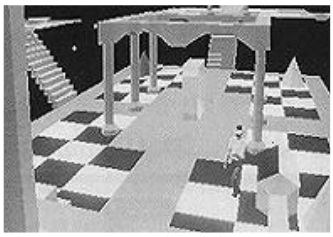


Figura 2.12. Dactyl Knightmare

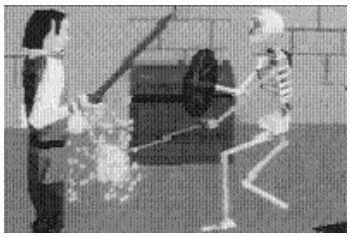


Figura 2.13. Legend Quest

Figura 2.14. Grid Busters

Conforme su uso se fue extendiendo fueron apareciendo diferentes dispositivos, desde los más sencillos hasta los más elaborados para el reconocimiento de los movimientos del jugador. Así nos encontramos con bandas para el tobillo o la muñeca, raquetas de tenis o guantes de boxeo, adaptándose cada uno al juego concreto.

Más tarde nos encontramos con el juego *Metal Gear Solid* (1998) de Konami. En éste el personaje principal puede entrenarse frente a diversas de situaciones del juego en un mundo virtual. Son las llamadas *VR missions* o misiones de realidad virtual.

Figura 2.15. *Metal Gear Solid - VR missions*

Educación sanitaria

Pese a que su uso no está muy extendido, podemos encontrar también la realidad virtual en el entrenamiento de los profesionales de la salud, desde el aprendizaje de la anatomía [72] hasta la simulación de operaciones de cirugía [197].

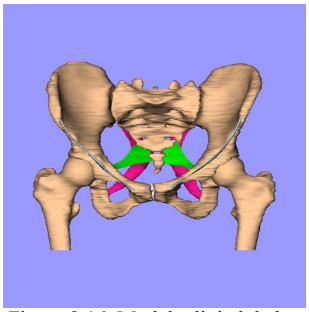


Figura 2.16. Modelo digital de la zona pélvica

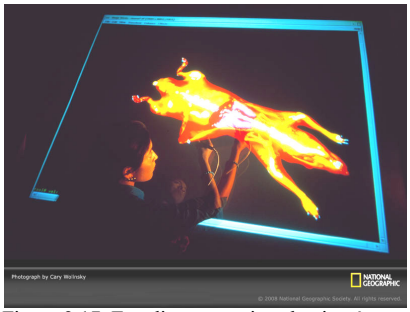


Figura 2.17. Estudiante practicando cirugía virtual con la imagen tridimensional de una rata

Usos terapéuticos

Existen varias terapias de exposición para las que se aplica la simulación de realidad virtual. Estas van desde los tratamientos de diferentes tipos de fobias hasta el tratamiento del estrés post-traumático [32].

En el caso del tratamiento de fobias, la utilización de la realidad virtual con modelos de imagen y sonido ha resultado ser de gran valor, suponiendo un paso intermedio entre la terapia de exposición básica y la exposición real [178].

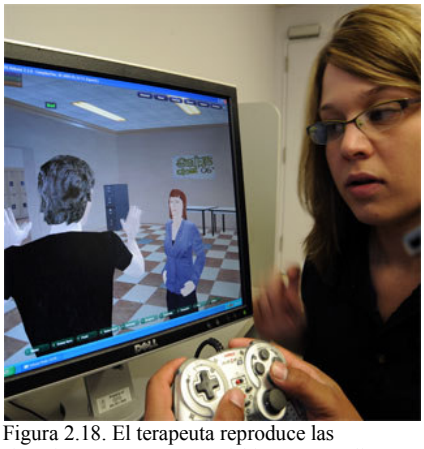


Figura 2.18. El terapeuta reproduce las situaciones que provocan fobias para analizar las reacciones de la paciente.

Figura 2.19. Empleo de la terapia virtual para tratar la fobia de la paciente a volar.

Para el tratamiento del estrés post-traumático, la marina estadounidense ha desarrollado una aplicación avanzada que sumerge a los veteranos de guerra con estrés post-traumático en escenas de combate urbano. Esta exposición parece llevar a una significante reducción de los síntomas.

Otro campo de investigación es la medicina física y la terapia de rehabilitación. La realidad virtual se está probando en la rehabilitación de las extremidades tras daños en la médula espinal y también para la parálisis cerebral y otras incapacidades. Para ello se utilizan dispositivos de respuesta táctil y robots de rehabilitación, mejorándose así la motivación del paciente durante los ejercicios [34].

Figura 2.20. Paciente durante una sesión de terapia, usando el sistema interactivo de ejercicios y rehabilitación virtual.

Procesos de Fabricación

La realidad virtual se utiliza para el diseño de nuevos productos, sirviendo a la ingeniería como herramienta auxiliar en los procesos de fabricación, prototipado y simulación [66].

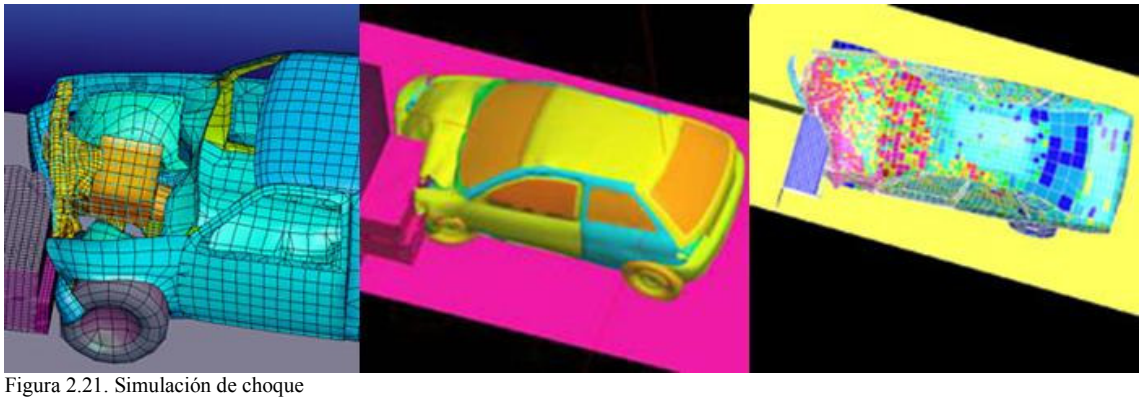


Figura 2.21. Simulación de choque

Su uso más extendido es como herramienta para el diseño asistido por ordenador, destacando también en la automatización del diseño electrónico, el análisis de elementos finitos y la fabricación asistida por ordenador.

Además, en combinación con la estereolitografía o la impresión 3D es posible crear partes físicas de objetos reales usados en ingeniería.

2.3. VRML (Lenguaje de Modelado de la Realidad Virtual)

2.3.1. Introducción

Los entornos virtuales pueden diseñarse de múltiples maneras. Mucho antes de aparecer el estándar VRML ya existían en el mercado paquetes de software para el diseño de entornos de realidad virtual y objetos 3D. Entre estos se encontraban *VREAM, Superscape, Sense8, WorldToolKit, Autodesk CDK y Division's dVS*. El problema que presentaban era que cada uno tenía su propia forma de trabajar y su propio lenguaje propietario y cerrado. De esta manera un programa no podía trabajar con los archivos de otro. Por tanto, si se quería que el uso de la realidad virtual en la Web se extendiera, era necesario desarrollar un estándar.

A continuación, y dado que se trata del lenguaje estándar para el modelado de la realidad virtual, se explican someramente sus características y funcionamiento.

2.3.2. Definición

Es un lenguaje de modelado de realidad virtual diseñado para su utilización en la Web. De esta manera se pueden incorporar en una página Web objetos 3D con los que se puede interactuar. En el momento de su estandarización se propuso también VRML como formato universal para el intercambio de gráficos 3D y multimedia, siendo numerosas sus aplicaciones, como se ha explicado más arriba.

En un documento VRML se especifican básicamente los vértices y aristas de un polígono 3D junto con sus características (color, textura, brillo, etc.) [56]. Por otro lado pueden añadirse enlaces a otros elementos como textos, sonidos, películas e imágenes. Por último pueden programarse animaciones, sonidos, efectos de luz y otros aspectos del mundo virtual para que interactúen con el usuario o sean activadas por eventos externos, temporizadores, etc.

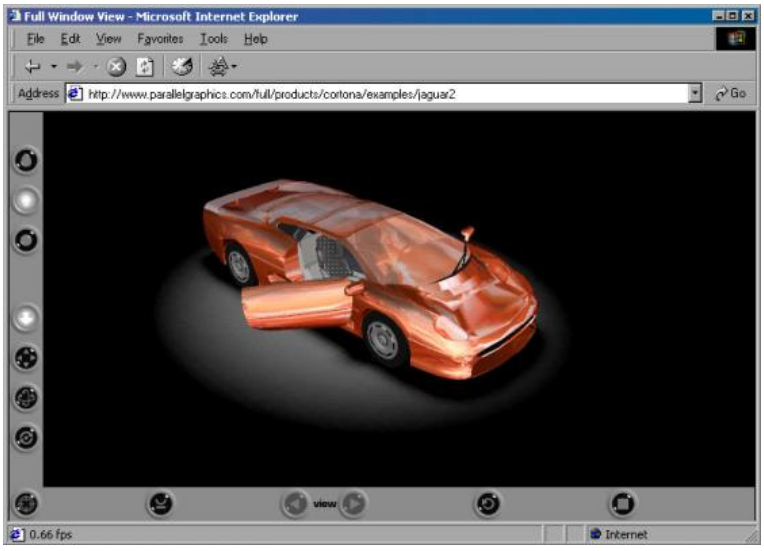


Figura 2.22. Navegador Web representando un modelo en VRML

Existe una amplia variedad de navegadores, así como herramientas para la creación de archivos VRML en diferentes plataformas. VRML soporta un modelo de ampliabilidad que permite crear nuevos objetos dinámicos 3D, desarrollando así, a partir del estándar, extensiones que interactúen entre sí.

2.3.3. Estandarización

En 1994, Mark Pesce y Tony Parisi crearon un programa llamado *Labyrinth* [131]. Se trataba de una prueba de concepto de lo que luego sería VRML. Éste podía cargar objetos a través de la Web, utilizando los mismos protocolos que utilizan las páginas Web. Se desarrolló con la ayuda de la librería de rutinas "*RealityLab*" de la compañía "*RenderMorphics*". Ésta generaba bastante rápido gráficos 3D vía software. *Labyrinth* fue presentado en la primera conferencia de la *World Wide Web* (WWW) en Mayo de 1994. Al cabo de unos meses la especificación ya estaba diseñada.

Cabe destacar que tomó como punto de partida un formato ya existente, el *OpenInventor* de Silicon Graphics [171], con el que guardaba mucho parecido, aunque desde entonces VRML evolucionó, alejándose de él.

El borrador de la especificación se presentó en la Segunda Conferencia Internacional de la WWW en Octubre de 1994. Un par de meses más tarde se liberó al dominio público un analizador para VRML. Éste se llamó *QvLib* y pese a los *bugs*, supuso un importante punto de partida para los desarrolladores de herramientas VRML.

Más adelante se creó el Consorcio VRML [192] para el desarrollo del formato, siendo finalmente aceptado como estándar internacional por la Organización Internacional para la Estandarización (ISO). La versión actual, completa y funcional es la VRML97 (ISO/IEC 14772-1:1997) [193].

2.3.4. Historia

El término VRML fue acuñado por Dave Raggett en un documento presentado en la Primera Conferencia Internacional de la WWW en 1994. En 1997 se desarrolló una nueva versión del formato, VRML97 (también conocida como VRML2 o VRML2.0), y se convirtió en un estándar ISO que fue usado en Internet en algunas páginas personales y sitios como *Cybertown*, un chat 3D que usaba el software *Blaxxun,* basado en VRML [35].

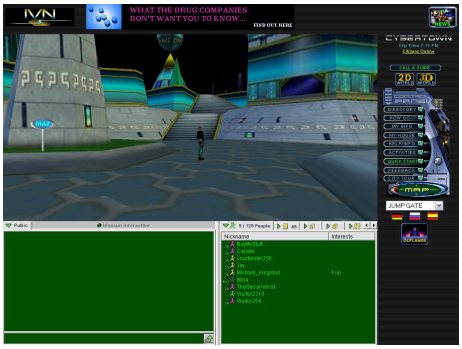


Figura 2.23. Videojuego *Cybertown*

El formato fue usado por el programa *Cosmo Software* de Silicon Graphics Inc [36]. Cuando en 1998 SGI reestructuró la división, fue vendido a Platinum Technologies. Ésta fue absorbida por Computer Associates, la cual no desarrolló ni distribuyó el software.

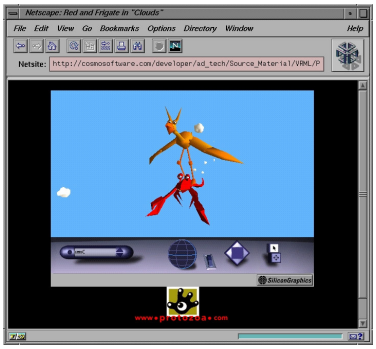


Figura 2.24. Cosmo Software

Para llenar el vacío generado, surgió durante los siguientes años una variedad de formatos propietarios para la Web3D, incluyendo Microsoft *Chrome* y Adobe *Atmosphere*, a los cuales no se da soporte hoy en día [168].

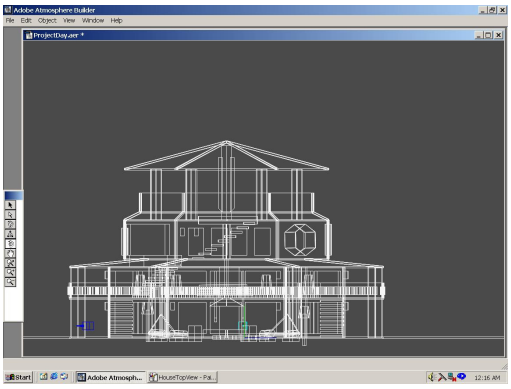


Figura 2.25. Adobe *Atmosphere*

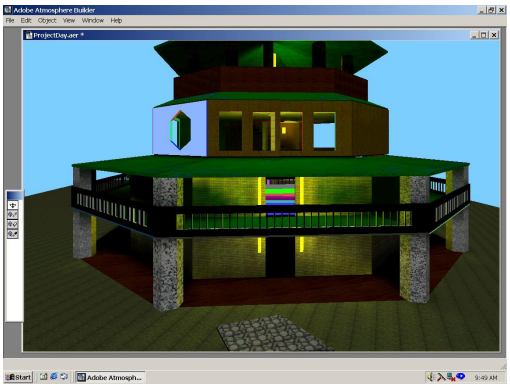


Figura 2.26. Adobe *Atmosphere*

Por otro lado, un *software* llamado *OpenVRML* [136] dispone de una herramienta de implementación del VRML para desarrollos multiplataforma. Sus librerías pueden usarse para añadir soporte de VRML y X3D a las aplicaciones, y existe también un *plugin* de GTK+ para representar mundos VRML/X3D en navegadores web.

Las capacidades del VRML se mantuvieron igual durante un tiempo, mientras que los gráficos 3D en tiempo real fueron evolucionando.

El VRML provocó mucho interés inicial, pero nunca experimentó un uso amplio y serio. Una razón pudo haber sido la falta de disponibilidad de banda ancha. En su época de mayor popularidad la mayoría de usuarios, particulares y profesionales, disponían tan sólo de acceso a internet vía red telefónica, teniendo que esperar durante largos periodos de tiempo para las descargas de datos. La experimentación con VRML se realizó principalmente en las áreas de educación e investigación.

2.3.5. Visores VRML

Al tratarse de un lenguaje destinado a su uso en la Web, los visores son por lo general los propios navegadores de Internet, utilizándose para ello *plugins* al efecto.

No obstante existen programas, editores y *applets*, escritos en *C++*, *Java*, *Java3D*, *OpenGL* o *Direct3D* que, sin ser navegadores Web, permiten visualizar los archivos VRML bajo Windows, destacando entre otros:

- *Instant Player* [71]
- *Octaga Player* [147]
- *BS Contact* [48]
- *SwirlX3D editor* [155]

2.3.6. Plugins VRML

Un *plugin* es una pequeña aplicación que funciona dentro de un programa anfitrión, proporcionándole a éste una función que hasta el momento no poseía. Esto permite ampliar el programa principal, dotándole de nuevas características conforme van siendo desarrolladas y siendo así cada vez más versátil.

Así pues, para poder mostrar y reproducir documentos VRML es necesario instalar un *plugin* VRML en el navegador Web.

Los principales navegadores Web existentes bajo Windows (*Internet Explorer, Mozilla Firefox, Opera, Safari y Google Chrome*) disponen entre otros de los siguientes *plugins* VRML:

- *Cosmo Player* [143]
- *Octaga Player* [147]
- *Cortona3D Viewer* [55]
- *SwirlX3D* [155]

2.4. X3D

2.4.1. Definición

X3D (Extensible 3D) [194] es un lenguaje estándar para la representación de objetos y escenas 3D, así como contenido multimedia diseñado para su uso a través de la Web. También se propuso como formato universal de intercambio y difusión de gráficos 3D y multimedia.

2.4.2. Características

X3D es el sucesor del VRML, mejorándolo con nuevas características. Puede codificar una escena mediante sintaxis XML [26] y *Open Inventor* VRML97 y presenta una nueva API, más avanzada. Su arquitectura, basada en componentes, permite su ampliabilidad y la especialización de la aplicación que se esté desarrollando. Existe para ello una gran variedad de componentes para X3D, cada uno con características especiales, que pueden combinarse según el objetivo final, pudiendo ir desde la simulación en ingeniería hasta la educación y el entretenimiento.

Sus características principales son:

- Soporta geometría poligonal, geometría paramétrica, transformaciones jerárquicas, iluminación, materiales, mapeado de texturas, sombreado de *pixels* y vértices, aceleración por hardware.
- Puede situar texto en el espacio, trabajar con gráficos vectoriales 2D y hacer composiciones con 2D y 3D.
- Es capaz de representar un modelo creado en un sistema CAD.
- Utiliza temporizadores e interpoladores para activar animaciones. Puede representar humanoides (*H-Anim*) y hacer *morphing*.
- Puede situar contenidos multimedia (audio y vídeo) en espacio virtual de la escena generada.
- Es posible interactuar en la escena mediante el uso del ratón y el teclado.
- Permite establecer la posición de las cámaras, el movimiento del usuario dentro de la escena y determinar los parámetros de colisión, proximidad y detección de visibilidad.
- La funcionalidad del navegador puede ampliarse con la incorporación de objetos creados por el usuario.
- La programación de *scripts* permite cambiar dinámicamente la escena.

- Es posible crear una nueva escena X3D a partir de otras ya existentes en la red, así como incorporar objetos de otras escenas.
- Puede trabajar con sistemas de partículas, representando así fuego, humo, niebla y otros efectos del estilo.
- Su punto fuerte es que puede decirle al navegador Web cómo debe representar los objetos 3D y cómo deben interactuar con el usuario.

2.4.3. Estandarización

El desarrollo del formato X3D (igual que el VRML) fue llevado a cabo por el Consorcio Web3D [192]. Su especificación (ISO/IEC 19775) fue aprobada por primera vez por la ISO en 2004. Las codificaciones XML y ClassicVRML para X3D (ISO/IEC 19776) fueron aprobadas por primera vez en 2005 [166].

2.4.4. Historia

X3D empieza su historia a partir de VRML, dado que es una evolución de éste.

El Consorcio VRML cambió su nombre a Consorcio Web3D y empezó a trabajar en el sucesor de aquél, que fue rediseñado como X3D, siendo ampliamente retro-compatible.

Basado en VRML, y posteriormente en X3D, surgió *H-Anim*; un estándar para la animación de humanoides [52].

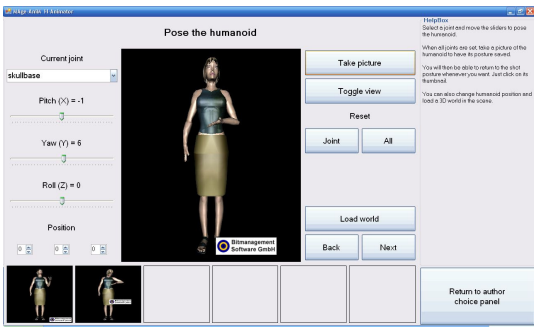


Figura 2.27. H-animator, del HCI Lab. de la Universidad de Udine

Las librerías *LibVRML97*, desarrolladas por Chris Morley, evolucionan a *OpenVRML*; una herramienta para la implementación de VRML en desarrollos multi-plataforma. Sus librerías permiten que las aplicaciones soporten VRML y X3D.

Surge un *plugin* basado en GTK+, un conjunto de herramientas para la creación de interfaces gráficas de usuario, que permite representar mundos VRML/X3D en navegadores Web.

2.4.5. Visores X3D

Los visores X3D son aplicaciones que analizan un documento X3D y después representan las escenas contenidas en él, no sólo mostrando objetos 3D desde diferentes puntos de vista sino también permitiendo su animación y la interacción del usuario con éstos. Éstos suelen aparecer como complementos de un navegador Web (*Firefox, Internet Explorer, Opera, Safari*, etc.), destacando entre otros:

- *Octaga Player* [147]
- *FreeWRL* [129]
- *OpenVRML* [136]
- *SwirlX3D* [155]

Fuera de navegadores Web también pueden ser analizados y representados documentos X3D mediante las siguientes aplicaciones, entre otras:

- *Blender 3D* [49]
- *Project Wonderland* [150]
- *InstantPlayer* [71]
- *view3dscene* [118]

2.5. Modelado 3D

2.5.1. Introducción

Para la generación de entornos virtuales es necesario un diseño previo de la escena que se va a componer y los objetos que va a incluir. Este diseño se realiza mediante el proceso de modelado 3D, que permite, no sólo diseñar la estructura de los elementos que van a aparecer sino determinar su aspecto y acabado final.

A continuación se explica qué son los modelos 3D, qué métodos y técnicas existen hoy en día para su representación. Por último se expone en qué consiste el proceso de renderizado y qué técnicas pueden aplicarse.

2.5.2. Modelo 3D

Es la representación matemática en formato digital de la estructura de un objeto 3D, real o abstracto [119].

Dado que dicha representación está almacenada internamente en el ordenador, para poder verlo en la pantalla es necesario aplicarle un proceso denominado "*renderizado 3D*", que no sólo muestra el objeto, sino que lo representa con su acabado final, incluyendo luces, sombras, texturas, reflejos, etc. Por otra parte, si se quisiera materializar físicamente un modelo 3D, existen en el mercado dispositivos de impresión 3D a tal efecto.

Hoy en día los modelos 3D están presentes en muchas áreas. En medicina se utilizan para la generación de modelos detallados de órganos y su estudio. En el cine se utilizan para crear personajes, objetos, escenarios, etc. En la química se utilizan para mostrar la estructura de compuestos químicos con un alto detalle [105]. En arquitectura permiten mostrar propuestas de edificios y paisajes. En ingeniería permiten el diseño de nuevos dispositivos, vehículos, estructuras, etc. En geografía se utilizan para la construcción de modelos geológicos 3D [201]. Y por último, en los videojuegos, que es donde más presencia tienen.

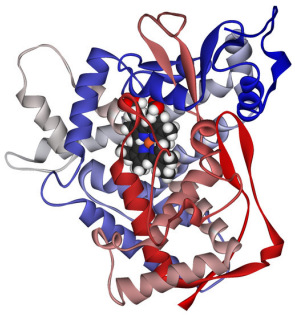


Figura 2.28. Modelo del citocromo P450 2C9

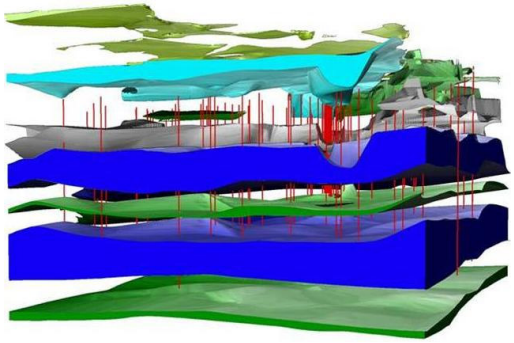


Figura 2.29. Modelo 3D de hidroestratigrafía

2.5.3. Modelado 3D

El modelado 3D es el proceso a través del cual se crea una representación matemática de la estructura de un objeto mediante el uso del ordenador.

Éste puede hacerse de forma automática o manual, siendo esta última forma muy similar a las artes plásticas, como la escultura.

2.5.3.1. Métodos de representación de objetos 3D

La forma básica de representación de un objeto 3D es una nube de puntos en el espacio, conectados entre sí por varias formas geométricas como triángulos, líneas, superficies curvas, etc [146].

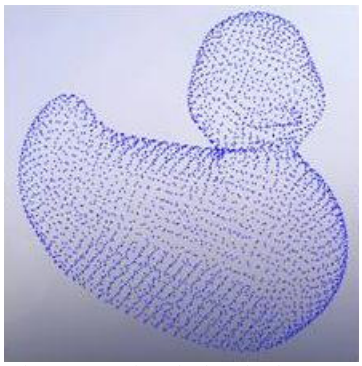


Figura 2.30. Ejemplo de nube de puntos

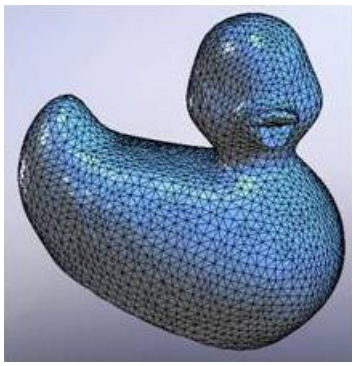


Figura 2.31. Nube de puntos conectados entre sí mediante triángulos

Existen cuatro formas principales de representar un objeto:

- **Modelado poligonal**
  Los puntos en el espacio, llamados vértices, se conectan entre sí por líneas, formando una estructura de malla poligonal [41]. Hoy en día la gran mayoría de los modelos 3D se construyen mediante modelado poligonal texturizado, ya que son flexibles y los ordenadores pueden renderizarlos rápidamente. Tienen un aspecto en el que se aprecian los polígonos y sólo pueden aproximarse a las formas curvas mediante el uso de muchos de ellos.

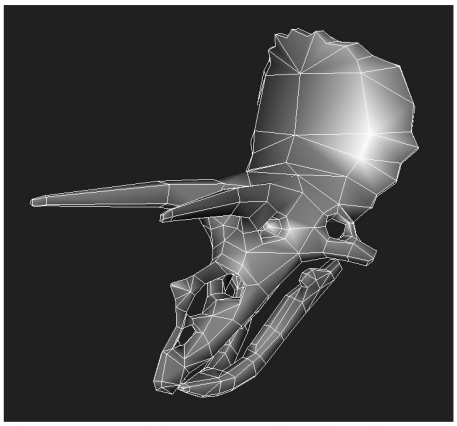


Figura 2.32. Modelo con su malla poligonal

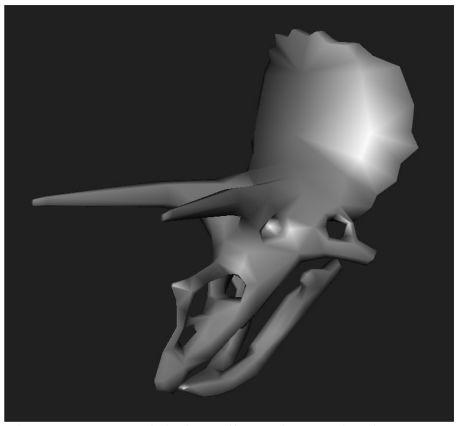


Figura 2.33. Modelado poligonal texturizado

El proceso de transformar la representación de un objeto, como una esfera, en una malla de polígonos se denomina teselado. El teselado se usa en el renderizado poligonal, donde los objetos parten de representaciones abstractas (primitivas) como esferas, conos, etc., que son redes de triángulos interconectados [106]. Pero no todas las técnicas de renderizado emplean la representación mediante polígonos.

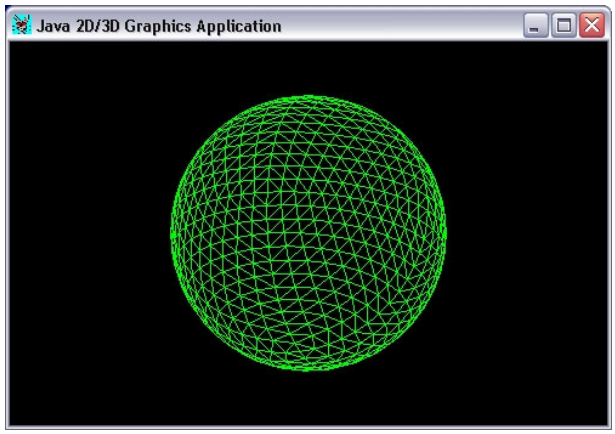


Figura 2.34. Teselado de una esfera mediante triángulos

- **Modelado mediante NURBS (B-Splines Racionales No Uniformes [134])** Las superficies NURBS se definen mediante curvas *spline*. Estas se ven influenciadas por unos puntos de control ponderados. La curva sigue a estos puntos, no interpolándolos necesariamente. Aumentar el peso de uno de los puntos hace que la curva se aproxime a éste. Las NURBS son superficies muy suaves, ideales para el modelado de formas orgánicas [195].

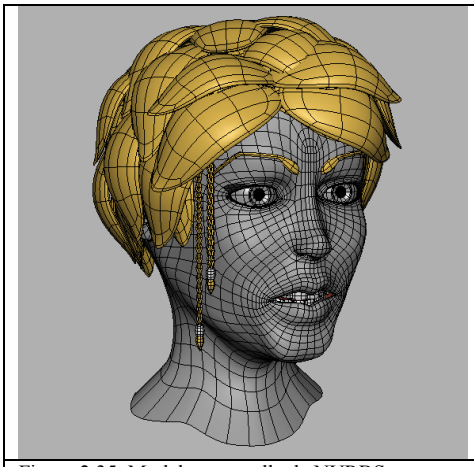


Figura 2.35. Modelo con malla de NURBS

Figura 2.36. Modelo NURBS texturizado

- **Modelado mediante *Splines* y Curvas de Bézier**
  La superficie del modelo se define mediante *splines* y curvas de Bézier [57].

  Las curvas de Bézier están a camino entre las NURBS y los polígonos, en términos de flexibilidad y facilidad de uso.

Figura 2.37. Modelado mediante Curvas de Bézier

- **Modelado mediante primitivas**
  Este procedimiento parte de formas geométricas primitivas, como esferas, cilindros, conos o cubos, a partir de los cuales se construyen modelos más complejos. Las ventajas que tiene son la facilidad y rapidez para construir modelos y que las formas están definidas matemáticamente, siendo por lo tanto absolutamente precisas.

  Este modelado está indicado para aplicaciones técnicas y no tanto para el modelado de formas orgánicas.

Figura 2.38. Modelado mediante primitivas

2.5.3.2. Técnicas de modelado

A parte del modelado clásico de un objeto, mediante la modificación de su estructura, ya sea añadiendo o quitando vértices, estirándolos o cambiando otras características, existen una serie de técnicas de modelado que facilitan mucho el trabajo del diseñador. Éstas permiten crear cierto tipo de objetos de forma sencilla, trabajar con objetos de difícil modelado, como es el caso de líquidos o estructuras complejas o refinar la estructura de un modelo de forma sencilla.

Entre las técnicas de modelado existentes, las más importantes son:

- **Geometría sólida constructiva**
  Esta técnica permite la creación de una superficie u objeto complejo combinando objetos más simples y aplicándoles operadores booleanos [104].

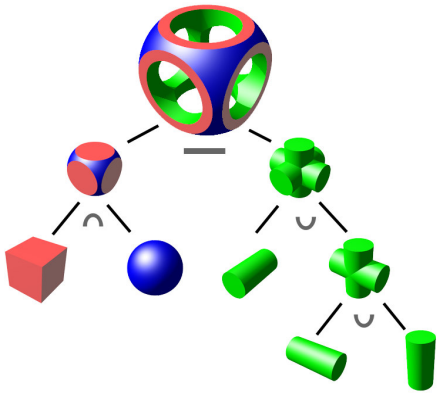


Figura 2.39. Ejemplo de combinación de objetos mediante operadores booleanos

- **Superficies implícitas**
  Se trata de una forma de representación muy útil, mediante superficies de nivel, que permite la creación de objetos complejos y la deformación de materiales que presenten muchos cambios topológicos, como es el caso de los líquidos [62].

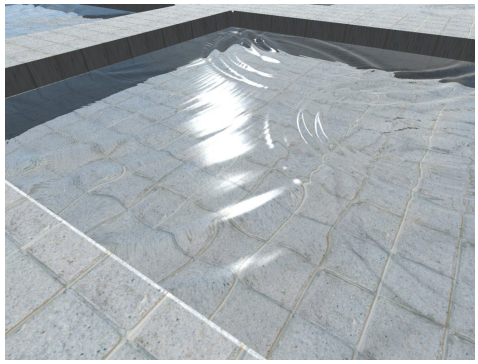


Figura 2.40. Simulación de movimiento del agua mediante superficies de nivel

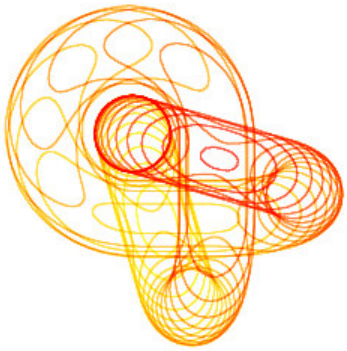


Figura 2.41. Objeto representado mediante sus superficies de nivel

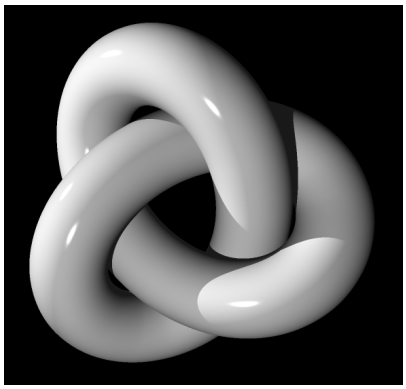


Figura 2.42. El mismo objeto finalmente renderizado

- **Superficies por subdivisión**
  Es una técnica que permite modelar una superficie de formas suaves a partir de una estructura tosca, de pocos polígonos [58]. Se aplica para ello un proceso recursivo de subdivisión de cada polígono en otros más pequeños, que permiten aproximarse cada vez más a las formas suavizadas.

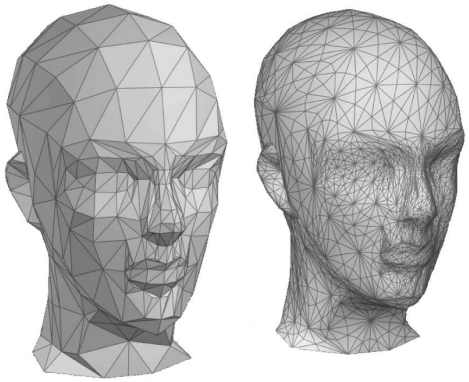


Figura 2.43. Las superficies por subdivisión dan un aspecto suave a un modelo con pocos polígonos

Por otro lado, los materiales complejos como la arena en suspensión, nubes, nieblas y sprays se modelan mediante un sistema de partículas, consistente en un conjunto de coordenadas tridimensionales que llevan asignadas puntos, polígonos, texturas o sprites [160].

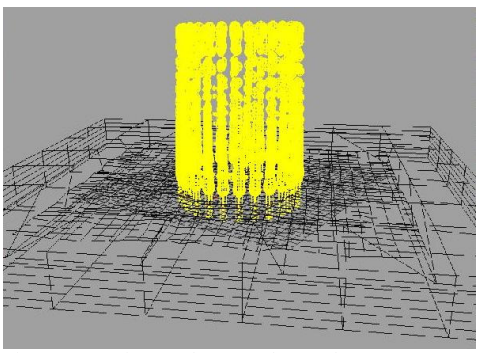


Figura 2.44. Sistema de partículas usado para crear una fuente

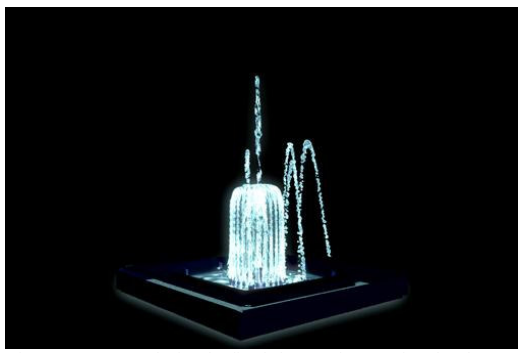


Figura 2.45. Renderizado final de una fuente simulando un sistema de partículas

Finalmente, los modelos 3D pueden clasificarse en dos categorías principales según cómo se represente su estructura:

- **Sólidos**
  Estos modelos definen íntegramente el volumen interior y exterior del objeto. Son los más realistas y los más difíciles de construir. Por eso, dada la carga de trabajo que supone su uso, se utilizan principalmente en las simulaciones no visuales, en el diseño asistido por ordenador, en aplicaciones visuales especializadas, como el "*ray tracing*" y en la geometría sólida constructiva [37].

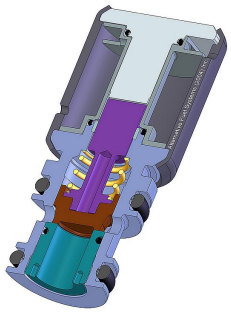


Figura 2.46. Modelo sólido de un inyector de fuel gaseoso

- **Estructuras huecas o de caparazón**
  Estos modelos representan únicamente la parte exterior o la envoltura del objeto sin tener en cuenta su volumen interior. La carga de trabajo es mucho menor que con los anteriores. Dado que la apariencia de un objeto depende sobre todo de su aspecto exterior, las estructuras huecas son las que más se utilizan en la generación de gráficos por ordenador. La mayoría de los modelos de videojuegos y películas son de este tipo.

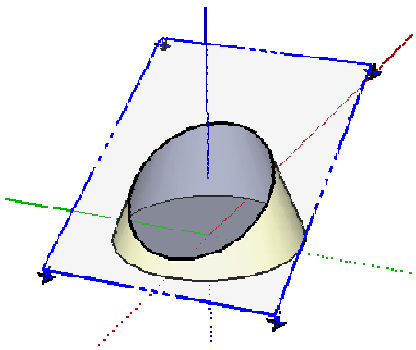


Figura 2.47. Ejemplo de modelo con estructura hueca

En el caso de este proyecto, y como era de esperar, se ha trabajado con estructuras de caparazón, dado que se ha orientado mucho más a la apariencia de los modelos más que a sus estructuras internas.

2.5.3.3. Comparación entre modelos 3D y 2D

A menudo, sin necesidad de modelar un objeto 3D, y limitándose a utilizar gráficos vectoriales 2D o imágenes *raster*, pueden conseguirse efectos fotorrealistas, similares a los que consigue el renderizado 3D. Esto es posible gracias al software de edición gráfica existente en el mercado.

No obstante, las imágenes 2D quedan en desventaja, dado que el uso de modelos 3D y su posterior renderizado permiten:

- Modificar ángulos o animar imágenes durante su renderizado con mayor rapidez.
- Calcular y aplicar automáticamente efectos fotorrealistas.
- Menor posibilidad de error humano debido a una mala colocación, repetición u olvido en la aplicación de un efecto visual.

Aún así las imágenes 3D también tienen sus inconvenientes, dado que el aprendizaje en el manejo de los programas de modelado 3D es difícil, así como también lo es conseguir determinados efectos fotorrealistas. Lo bueno es que a menudo pueden conseguirse con filtros especiales de renderizado que ya vienen incluidos en el propio software de modelado 3D.

Algunos artistas optan por coger lo mejor de cada método, usando una combinación de modelado 3D seguido de la edición de las imágenes 2D renderizadas a partir de dicho modelo 3D. El resultado es lo que se denomina 2.5D, con el que se obtienen imágenes de mayor calidad que las renderizadas en tiempo real [161].

Figura 2.48. Imagen 3D, renderizada en tiempo real

Figura 2.49. Imagen 2D obtenida a partir de un prerrenderizado 3D, de mayor calidad que el renderizado en tiempo real

2.5.3.4. Mercado de modelos 3D

La tecnología *3D Catalog* ha revolucionado el mercado de los modelos 3D [67], ofreciendo colecciones gratuitas de modelos de calidad para los usuarios de programas de diseño asistido por ordenador. *Google 3D Warehouse* [87] es un ejemplo de ello.

Por otra parte, existe también un mercado de compra-venta de modelos 3D, texturas, *scripts*, etc. Esto permite a los creadores obtener beneficios de sus modelos y a las compañías ahorrar dinero comprando modelos prefabricados en lugar de pagar el coste de su creación desde cero. Normalmente los mercados y los creadores se reparten los beneficios en un 50%. El creador conserva la propiedad del modelo y el cliente tiene únicamente derecho a usar y mostrar el modelo.

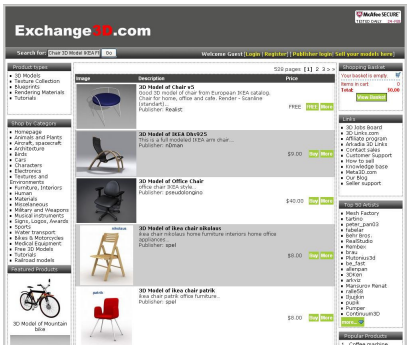


Figura 2.50. Portal Exchange3D de intercambio y compra de modelos 3D

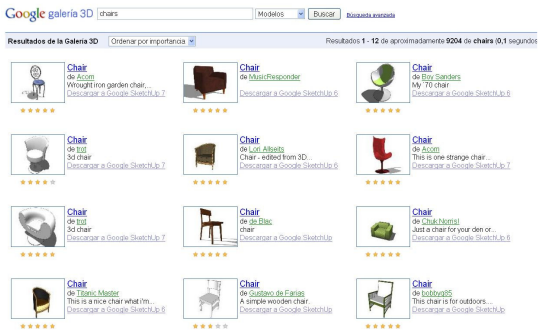


Figura 2.51. Google 3D Warehouse, para la descarga gratuita de modelos 3D en *Google SketchUp*

2.5.3.5. Software para modelado 3D en ordenador

Hay en el mercado numerosos programas especializados en modelado de objetos 3D. La mayoría de ellos tienen una arquitectura basada en los *plugins*, de manera que pueden ampliarse sus características a la carta. Otros funcionan como complementos de terceras aplicaciones, como es el caso de *Shaper* o *Lofter*, que funcionan bajo *3DS Max*.

Características

La mayoría de los programas se utilizan para modelar objetos del mundo real, desde plantas hasta automóviles. Otros, sin embargo están especializados en el modelado de ciertos objetos, como compuestos químicos o formas orgánicas, como órganos del cuerpo humano.

Permiten crear y modificar objetos 3D mediante la edición de su estructura de malla 3D, pudiendo añadir, quitar, estirar y cambiar elementos de ésta. Es posible ver el modelo desde cualquier punto de vista, incluso desde varios puntos de vista a la vez, mediante varias ventanas. El modelo puede rotarse y se puede hacer *zoom* sobre él.

La mayoría de los programas incluyen unas características comunes, como "*ray tracers*" (trazadores de rayos), que se verán más adelante, herramientas de renderizado, mapeado de texturas, animación de los modelos o generación de un archivo de video a partir de varias escenas de renderizado.

Paquetes de software más conocidos

Entre los programas de modelado 3D más conocidos se encuentran:

- *3DS Max*
  Utilizado principalmente para modelar objetos, personajes y crear escenas de video en videojuegos. También se emplea en visualizaciones arquitectónicas, ya que es altamente compatible con *AutoCAD*. Utilizado en la producción de la película *Kaena: La profecía* (2003) de Chris Delaporte.

- *form·Z*
  Es un programa de modelado de superficies y sólidos 3D. Se caracteriza por su renderizado fotorrealista y capacidad de animación. Es utilizado principalmente por arquitectos, diseñadores de interiores, ilustradores y diseñadores de productos y escenarios.

- *Houdini*
  Utilizado para la creación de efectos visuales y animación de personajes, se usó en la película de animación *Salvaje* (2006) de Steve Williams.

- *Maya*
  Se trata de un programa para el modelado y animación 3D. Caracterizado por su capacidad de simulación, creación de efectos visuales, renderizado y composición. Su principal uso es en cine, televisión y sobretodo en videojuegos, aunque también se utiliza para la visualización arquitectónica y el diseño.

- *SketchUp Pro*
  Es un pack de modelado 3D que se caracteriza por representar una aproximación al modelado mediante bocetos. Soporta la exportación de modelos 2D y 3D al *3D warehouse* de Google, donde los usuarios pueden compartir gratuitamente sus contenidos.

2.5.4. Renderizado

2.5.4.1. Definición

Es el proceso mediante el cual se genera la imagen digital de un objeto en la pantalla del ordenador a partir de su modelo matemático [70]. Su objetivo es representar dicho objeto con todos los matices visuales de su acabado final (geometría, puntos de vista, texturas, iluminación, sombreado, etc.).

2.5.4.2. Usos

Una vez se ha terminado de modelar los objetos que conforman una escena, se aplica el renderizado. Éste añade texturas, iluminación y demás efectos al conjunto. El resultado es una imagen de la escena con el acabado final que se buscaba [44].

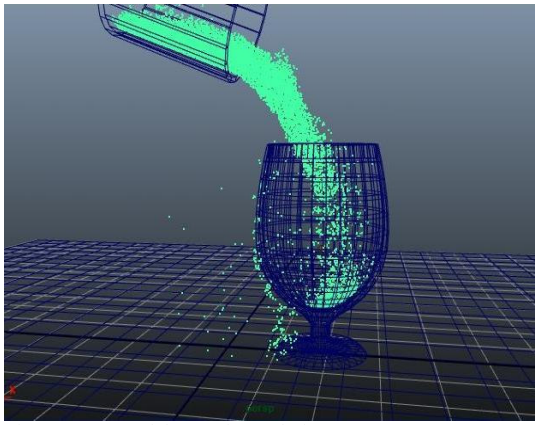


Figura 2.52. Modelo 3D sin renderizar

Figura 2.53. Modelo 3D renderizado

En el caso de los gráficos 3D, el renderizado puede ser un proceso lento, como ocurre con el prerrenderizado, o un proceso rápido, como pasa con el renderizado en tiempo real. El prerrenderizado es un proceso de cálculo largo e intensivo que suele usarse para la creación de películas, donde se exige un acabado perfecto, mientras que el renderizado en tiempo real se usa más para videojuegos, siendo necesario para ello el uso de tarjetas aceleradoras gráficas.

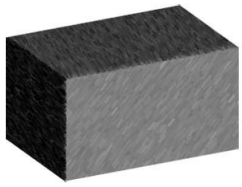
En películas de animación deben renderizarse multitud de fotogramas que luego son ordenados en un programa especializado. La mayoría de los programas de edición de imagen 3D incorpora esta función.

2.5.4.3. Características

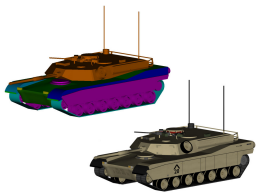
Una imagen renderizada puede entenderse como la suma de diversas características y efectos visuales aplicadas sobre un objeto 3D. Y dado que el objetivo final del diseñador es que el objeto y la escena parezcan lo más real posible, se han puesto todos los esfuerzos en conseguir simular dichas características y efectos visuales.

Hoy en día existen numerosos algoritmos y técnicas particulares, que en combinación permiten simular las siguientes características visuales en el renderizado:

Sombreado
Variación del color y el brillo de una superficie con la iluminación [152].



Mapeado de texturas
Método para aplicar detalles a las superficies [135].



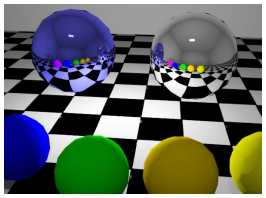
Bump mapping
Método para simular pequeñas irregularidades en las superficies [127].

Niebla
Desvanecimiento de la luz cuando pasa a través de una atmósfera densa [40].

Sombras
Efecto de obstrucción de la luz [183].

Penumbras
Variación de la sombra, provocada por fuentes de luz parcialmente oscurecidas [203].

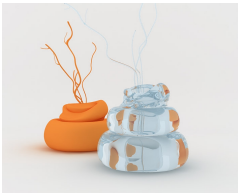
Reflexión
Efecto espejo o reflexión con alto brillo [163].



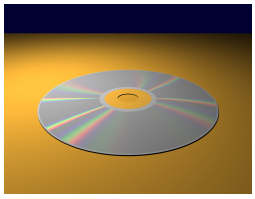
Transparencia u opacidad
Transmisión parcial de luz a través de objetos sólidos [103].

Translucidez
Transmisión de la luz altamente dispersada a través de objetos sólidos [76].

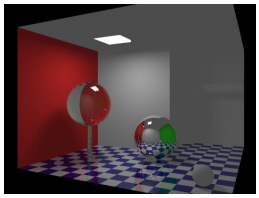
Refracción
Cambio del ángulo de incidencia de la luz asociado a la transparencia [50].



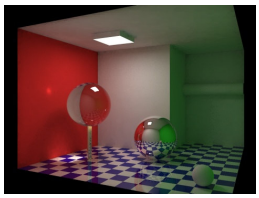
Difracción
Cambio del ángulo de incidencia así como dispersión e interferencia de la luz al pasar por un objeto o apertura [61].



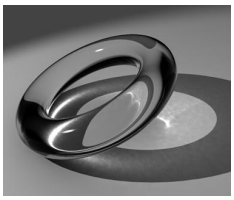
Iluminación directa
Iluminación generada en una superficie por una fuente de luz [102].



Iluminación indirecta
Iluminación generada en una superficie por la luz reflejada en otra superficie.



Cáusticas
Reflexión de la luz en un objeto brillante, o enfoque de la luz a través de un objeto transparente, para producir brillos en otro objeto o superficie [165].



Profundidad de campo
Los objetos aparecen borrosos o fuera de enfoque cuando se alejan o acercan del objeto enfocado [101].

Borrosidad en movimiento
Los objetos aparecen borrosos debido al movimiento de éstos a alta velocidad o al movimiento de la cámara [124].

Renderizado no fotorrealista
Renderizado de escenas en un estilo artístico con la intención de parecerse a una pintura o dibujo [63].

2.5.4.4. Técnicas de renderizado

Una parte importante del trabajo de renderizado es el tratamiento de la luz y la simulación del transporte de ésta por toda la escena. Para esto se suele tratar a la luz como una serie de rayos que van y vienen por el entorno virtual. El problema está en que trazar todos y cada uno de los rayos de luz existentes en una escena es inviable; llevaría una enorme cantidad de tiempo. Incluso trazar tan sólo una parte de estos rayos lo suficientemente grande como para producir una imagen aceptable puede llevar una excesiva cantidad de tiempo. Es por eso que los motores de renderizado actuales permiten una simulación eficiente de la luz en la escena mediante cuatro técnicas principales:

- *Rasterización*. Emplea el renderizado mediante líneas de escaneo. Proyecta geométricamente los objetos de la escena sobre un plano imagen, sin efectos ópticos avanzados.
- *Ray casting* (lanzamiento de rayos). Considera que se observa la escena desde un punto de vista concreto, y calcula la imagen observada basándose sólo en la geometría y las leyes ópticas básicas de intensidad de reflexión. Utiliza los Métodos de Monte Carlo para simplificar el proceso.
- *Radiosidad*. Usa la matemática de elementos finitos para simular la dispersión difusa de la luz en las superficies.
- *Ray tracing* (trazado de rayos). Similar al ray casting, emplea una simulación óptica más avanzada y los métodos de Monte Carlo para obtener resultados más realistas, aunque a una velocidad bastante más lenta.

La mayoría de los programas combinan dos o más de estas técnicas para obtener resultados excelentes y a un precio razonable.

A continuación se detallan las características de cada una de estas técnicas de renderizado.

Rasterización y Renderizado mediante líneas de escaneo

La representación de una imagen puede realizarse a partir de primitivas en lugar de *pixels*. Estas primitivas suelen ser formas geométricas sencillas, como líneas, triángulos, polígonos, etc., situadas en el espacio [140].

El renderizado de un objeto mediante *pixels* sería un proceso largo y muy lento, mientras que hacerlo mediante primitivas resulta más útil y rápido. En esta técnica un algoritmo pasa por todas las primitivas, determina a qué *pixels* de la imagen afecta, y modifica esos *pixels* como corresponda. A esto se llama rasterización y es el método de renderizado usado por todas las tarjetas gráficas actuales.

Si hay áreas de la imagen que no contienen primitivas, la rasterización las pasará por alto. Esta omisión no la haría el renderizado mediante *pixels* y perdería mucho tiempo en ello. Por otra parte, la rasterización reduce el trabajo redundante aprovechando que los *pixels* que ocupa una primitiva tienden a estar contiguos en la imagen. Todo esto la convierte en la elección más común cuando se busca un renderizado interactivo.

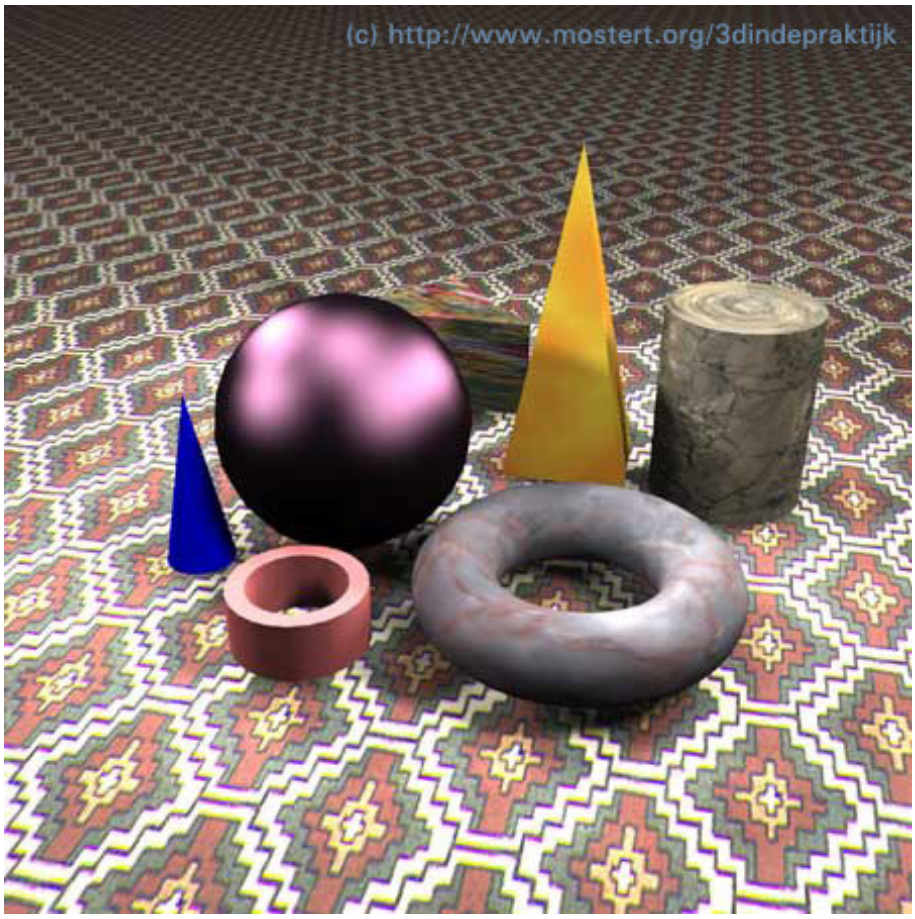


Figura 2.54. Renderizado mediante líneas de escaneo

Ray casting. Lanzamiento de rayos

Es una técnica utilizada principalmente en el renderizado en tiempo real, sacrificando un poco el detalle de las imágenes en pro de un mejor rendimiento. Es por eso que su uso más extendido es en videojuegos y dibujos animados. Las imágenes resultantes tienen un aspecto típicamente poligonal, aunque estos detalles pueden cuidarse mediante trucos adicionales.

La técnica consiste en establecer un punto de vista determinado y lanzar desde éste una serie de rayos que permiten analizar la geometría de la escena, pixel a pixel y línea a línea. En el momento en que un rayo intersecta con algún objeto, se evalúa el valor del color en el punto de intersección mediante varios métodos. Otra forma de determinar el color en dicho punto es mediante un mapa de textura. Un método más sofisticado permite variar el valor del color mediante un factor de iluminación. Para simplificar la técnica, se promedian un número de rayos en direcciones ligeramente diferentes.

También se pueden emplear simulaciones más sencillas de propiedades ópticas. Se hace un simple cálculo del rayo desde el objeto al punto de vista. Se hace otro cálculo del ángulo de incidencia de los rayos desde la fuente de luz, y con estos, así como con las intensidades específicas de las fuentes de luz, se calcula el valor del pixel. Otra simulación usa iluminación trazada con un algoritmo de radiosidad (comentado en el siguiente punto) o una combinación de ambos.

Figura 2.55. Renderizado mediante *ray casting*

Radiosidad

La radiosidad es un método que intenta simular la forma en la que superficies directamente iluminadas actúan como fuentes indirectas de luz que iluminan a su vez otras superficies. Esto produce un sombreado más realista y parece capturar mejor el ambiente de una escena interior. La base óptica de la simulación es que parte de la luz difusa desde un punto en una superficie dada se refleja en un gran espectro de direcciones e ilumina el área alrededor de ella [70].

Muchos renderizados realizan un cálculo muy sencillo de la radiosidad, simplemente iluminando una escena entera con un factor conocido como ambiente. Cuando se combina un cálculo avanzado de la radiosidad con un algoritmo de *ray tracing* de alta calidad, las imágenes muestran un realismo convincente, particularmente en escenas de interior.

Por todo esto, la radiosidad se ha convertido en el método más utilizado para el renderizado en tiempo real, y está siendo usada para crear un gran número de películas de animación 3D [39].

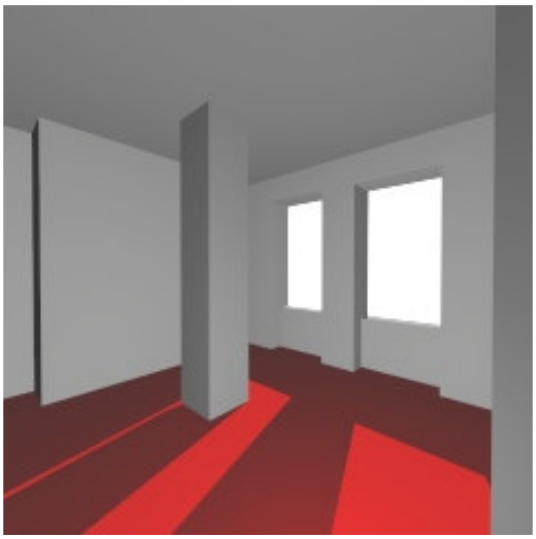


Figura 2.56. Escena creada mediante ray tracing

Figura 2.57. Escena con radiosidad

Ray tracing. Trazado de rayos

Es una técnica que simula el flujo natural de la luz, interpretada como partículas. Para aproximar la solución a la ecuación de renderizado se emplean los Métodos de Monte Carlo [205].

Esta técnica consiste en la proyección de múltiples rayos por cada pixel y su trazado, no sólo hasta el primer objeto con el que intersecten, sino un número determinado de rebotes, usando leyes de la óptica como "el ángulo de reflexión es igual al ángulo de incidencia" y otras leyes referentes a la refracción y rugosidad de la superficie. Una vez termina el proceso se evalúa la iluminación de la superficie en el punto final, y en todos los diferentes rebotes producidos para poder así calcular el valor observado desde el punto de vista.

Dada la enorme carga de trabajo que estos cálculos conllevan, esta técnica no ha podido tenerse en consideración para el renderizado en tiempo real, y hasta hace poco, ni siquiera para renderizar en tiempo real secuencias de calidad baja. No obstante sí que se ha utilizado para renderizar escenas de efectos especiales y en anuncios donde, por tratarse de secuencias breves, se ha podido renderizar en alta calidad e incluso en calidad fotorrealista.

Para aligerar de trabajo a este proceso y hacerlo más rápido, se ha optado por reducir el número de cálculos en las áreas que requieren menos detalle. Esto ha hecho que se haya extendido el uso de esta técnica. Hoy en día existen tarjetas aceleradoras gráficas que permiten utilizarla y *demos* de juegos que utilizan esta técnica, renderizando en tiempo real tanto por software como hardware.

Figura 2.58. Escena generada mediante *ray tracing*

2.5.4.5. Muestreo y filtrado

Un problema con el que tiene que enfrentarse cualquier sistema de renderizado es el del muestreo. En esencia, el proceso de renderizado intenta transformar una función continua del espacio imagen en una serie de colores, usando para ello un número finito de *pixels*. Como consecuencia del Teorema del Muestreo [115], no se puede representar información más detallada que la que la resolución de imagen permite.

Así pues, si se usa un algoritmo de renderizado sencillo, las altas frecuencias en la función de la imagen causarán un efecto de dientes de sierra llamado *aliasing* en la imagen final. Para poder eliminarlo, el algoritmo de renderizado debe filtrar esas altas frecuencias. Este proceso se denomina *anti-aliasing*, y el resultado es una imagen suavizada.

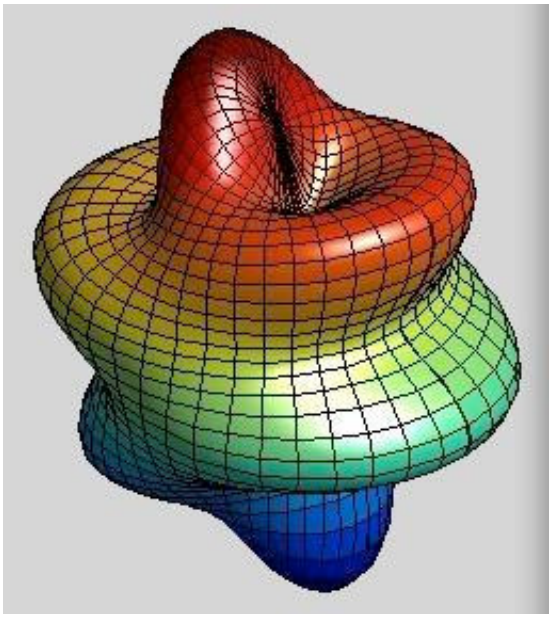


Figura 2.59. Modelo con efecto *aliasing*

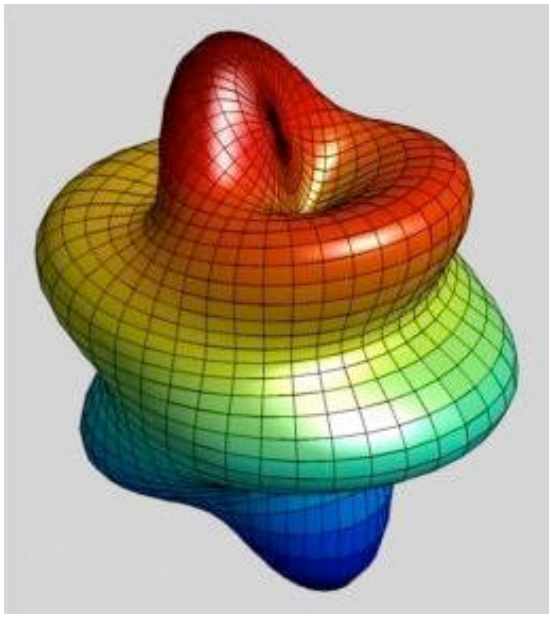


Figura 2.60. Modelo con filtro *anti-aliasing* aplicado

2.5.4.6. Óptica física

Como se habrá podido apreciar en el apartado anterior, el renderizado está relacionado casi exclusivamente con la naturaleza corpuscular de la luz, tratándola como una serie de partículas que rebotan por toda la escena (óptica geométrica). Es una forma de simplificarlo, pero es apropiada y da buenos resultados.

No obstante también es posible renderizar fenómenos de la naturaleza ondulatoria de la luz (óptica física). Ésta es insignificante en muchas escenas y difícil de simular. Entre los fenómenos de ésta se incluyen la difracción [59], como puede apreciarse en los colores de los CD y DVD, y la polarización [110], como ocurre con las pantallas LCD.

Ambas naturalezas de la luz pueden reproducirse mediante ajustes en el modelo de reflexión, el cual describe la interacción de la luz con una superficie.

Figura 2.61. Efecto de difracción en un CD

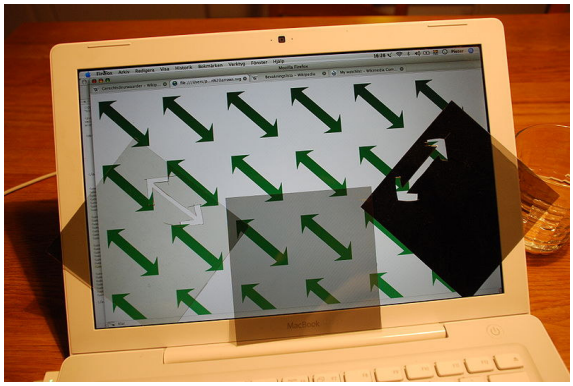


Figura 2.62. Efecto de polarización en una pantalla LCD

2.5.4.7. Software de renderizado 3D

En el mercado hay una gran variedad de renderizadores o motores de renderizado. Unos son aplicaciones independientes, incluso de código abierto, mientras que otros aparecen integrados en los paquetes de modelado comentados anteriormente. Todos se basan en la física de la luz, la percepción visual, la matemática, y demás disciplinas relacionadas. Entre los más conocidos se encuentran los siguientes:

- *3Delight*. Renderizador que cumple con el estándar *RenderMan* [156].
- *Aqsis*. Una suite de renderizado que cumple con el estándar *RenderMan.*
- *Brazil*. Motor de renderizado para *3DS Max* y *VIZ.*
- *FPrime* para *Lightwave*. Añade una vista previa rápida y puede usarse en muchos casos para un renderizado final.
- *Gelato*. Renderizador en tiempo no real y acelerado por hardware.
- *Hypershot*. Renderizador fotorrealista en tiempo real para imágenes de alta resolución.

Capítulo 3

Desarrollo del proyecto

3.1. Introducción

En este capítulo se muestra una descripción general del proyecto, donde se describe cuál ha sido su proceso y el flujo de trabajo que se ha seguido para cumplir los objetivos. A continuación se hace una descripción de la situación geográfica de la zona de estudio. Por último se describen las fuentes de información a partir de las cuales ha sido posible desarrollar las diferentes fases de que se compone el proyecto.

3.2. Descripción general del proyecto

Para llevar a cabo este proyecto ha sido necesario el modelado 3D de los edificios e instalaciones de la zona de estudio, partiendo para ello de la cartografía catastral correspondiente.

En un principio se probó con varias aplicaciones de modelado mediante X3D (SwirlX3D, Vivaty Studio y X3D-Edit) por tratarse del estándar de realidad virtual aplicado a la Web. Por otro lado también se probó el modelado mediante *Google SketchUp*, siendo éste más sencillo e intuitivo que con las anteriores aplicaciones. El resultado de todas las pruebas era correcto, pero se comprobó que *Google SketchUp*, al estar en combinación con *Google Earth*, disponía de una infraestructura más completa y se adaptaba mejor a los objetivos de este proyecto. Disponía de un entorno tridimensional de la Tierra con imágenes satélite que permitía una ubicación de los modelos en un entorno muy elaborado que además aportaba información de relieve gracias a los MDT asociados, así como información adicional mediante la activación de capas. Para obtener el mismo resultado en X3D hubiera sido necesario georreferenciar y solapar numerosas imágenes satélite y asociarlas a MDTs de una amplia zona alrededor de la zona de estudio. Por otro lado *Google Earth* cuenta con herramientas y entidades para señalar de forma sencilla lugares de interés, ampliar la información de dichos lugares y mostrarlos mediante movimientos de cámara automatizados. En X3D todo lo anterior es más complicado. Al tratarse de un lenguaje pensado para la interacción del usuario con un entorno tiene muchos aspectos y parámetros que no pueden establecerse mediante una aplicación y necesitan ser programados directamente en dicho lenguaje, requiriendo para ello un aprendizaje mucho más profundo de éste. Por último, dado que X3D está pensado para su aplicación en la Web se pensó en la posibilidad de mostrar de alguna manera los modelos generados en *SketchUp* bajo X3D en una página Web. Otra opción que se barajó fue subir los modelos al servidor de Google y compartirlos públicamente para que fueran mostrados en *Google Earth*. Pero finalmente, con la salida de la API de *Google Earth* durante la elaboración de este proyecto se resolvieron todas estas cuestiones. Se decidió hacer el modelado 3D mediante *SketchUp* y trabajar en combinación con *Google Earth* y su API para mostrar los modelos y la información adicional sobre los comercios de la zona y poder realizar consultas sobre los mismos.

Así pues se procedió al modelado 3D, después del cual fueron necesarios la captura y el tratamiento de imágenes de los edificios reales para que los modelos tuvieran un acabado final realista. Una vez preparadas las imágenes se aplicaron a los modelos, consiguiendo el acabado que se buscaba.

A continuación se mostraron todos los modelos a la vez bajo *Google Earth* para comprobar que la aplicación podía moverlos con rapidez. Además, el uso de esta aplicación permitiría que estos modelos pudieran ser vistos por todos los usuarios de la Web.

Como complemento a lo anterior se quiso ofrecer información sobre los comercios y servicios que pueden encontrarse en la zona de estudio. Para ello ha sido necesario recabar información alfanumérica y gráfica. Con esta información ha sido posible crear marcas de posición y ventanas de información bajo Google Earth.

Otro de los objetivos es que el trabajo fuera accesible desde la Web. Esto, anteriormente, sólo era posible subiendo los modelos y marcas de posición a los servidores de Google, compartiéndolos así con todos los usuarios. No obstante, gracias a la reciente aparición de la API de *Google Earth* no fue necesario hacer esto, ya que es posible crear una página Web que muestre la escena digital generada y permita moverse por ella.

Dado que otro de los objetivos es que pueda hacerse algún tipo de consulta, durante la creación de la página Web se programó la posibilidad de hacer consultas sobre comercios. De esta manera se facilitaba su búsqueda mediante la elección del tipo de comercio y la posterior elección de aquél sobre el que se quiere tener información. Por otro lado se programaron una serie de rutas que permiten dar un paseo por diferentes zonas de la escena.

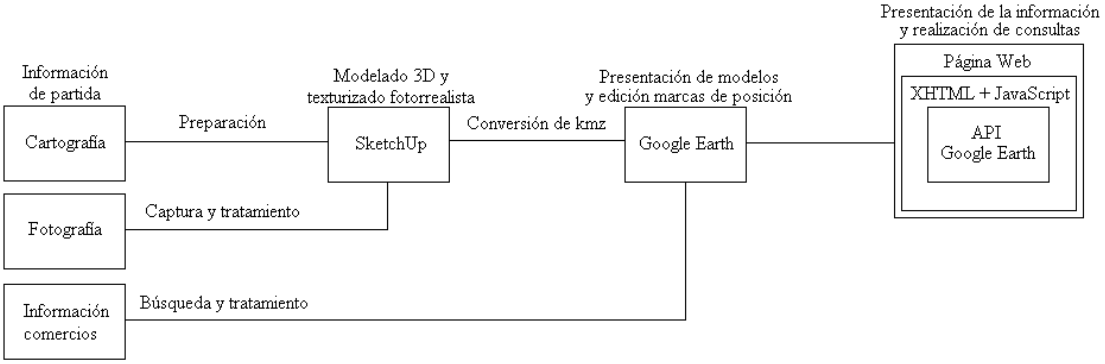


Figura 3.1. Esquema del flujo de trabajo

Para el desarrollo del proyecto ha sido necesario el aprendizaje de múltiples conocimientos, la mayoría de los cuales no fueron vistos durante la carrera:

- *SketchUp* dispone de algunas herramientas que pueden encontrarse en *AutoCAD* o *MicroStation*, pero tiene muchas otras dedicadas al trabajo en 3D con las que no se había tenido contacto. Además el trabajo en 3D ha generado en ocasiones situaciones que han debido resolverse y en las que no se tenía experiencia. En ocasiones se han producido comportamientos extraños de la geometría generada que han debido ser resueltos y han requerido mucha paciencia y tiempo.

- Otra aplicación con la que ha habido que desenvolverse ha sido Adobe *PhotoShop*. El retoque fotográfico que se ha hecho mediante este programa ha sido sencillo, sin entrar en los trabajos complejos que permite. No obstante se ha tenido que dedicar bastante tiempo para encontrar la mejor forma de dar un aspecto correcto a las fotografías tratadas.

- En la parte de retoque fotográfico también ha habido que tratar con el montaje de fotografías panorámicas. Esto ha hecho que hubiera que ponerse al día en las técnicas y programas que permiten su montaje. Ha supuesto un tiempo importante de formación y numerosas pruebas hasta tener una idea clara de la mejor manera de conseguir un resultado óptimo y lo más correcto posible.

- En la edición de las marcas de posición también ha habido que aplicarse. *Google Earth* sólo permite una edición somera de éstas. La edición al detalle ha sido necesaria hacerla fuera de *Google Earth* para poder adaptarlas a las necesidades de la API en la página Web. Esto ha llevado a un proceso iterativo en el que se ha refinado el aspecto y comportamiento de las marcas de posición.

- También ha sido necesario el aprendizaje de los lenguajes XML y KML, para lo cual ha sido de mucha ayuda la generación de este tipo de archivos bajo *Google Earth*. Esto ha permitido trabajar con ellos y comprender su estructura y funcionamiento.

- Por otra parte, al tener que desarrollar una página Web ha habido que formarse en la programación mediante XHTML, y también un somero estudio en *JavaScript*, basado únicamente en las necesidades de la aplicación.

- Por último ha habido que aprender el funcionamiento y programación de la API de *Google Earth* para su integración en la página Web, para lo que ha sido necesario estudiar la documentación que Google ofrece al respecto.

3.3. Ámbito de la zona de estudio

La zona de estudio se encuentra ubicada en el término municipal de La Puebla de Farnals [153], a mitad de camino entre Valencia y Sagunto. Pertenece a la comarca de L'Horta Nord, en la provincia de Valencia y limita con los términos municipales de Massamagrell, El Puig y Rafelbunyol.

Figura 3.2. Vista general de la zona de estudio

El término municipal cuenta con dos áreas urbanas. Una interior, que es el núcleo de población original y otra en la zona de la costa, de desarrollo posterior. Esta segunda zona es con la que se ha trabajado para la creación de modelos 3D de edificios e instalaciones de los complejos urbanísticos. Concretamente, la zona que ha sido modelada es la limitada por la avenida de la Constitución, la avenida del Mar, la Avenida de Massamagrell y la costa (Fig. 3.3.).

Toda esta zona está constituida en su práctica mayoría por complejos urbanísticos. También consta de zonas verdes, zonas de recreo, aparcamientos, paseo marítimo y una zona comercial dentro del área de puerto deportivo. El objetivo es darle a esta zona una representación en formato digital, mediante modelos 3D.

Figura 3.3. Vista al detalle de zona de estudio

3.4. Información de partida

El flujo de trabajo está claramente definido:

- Modelado 3D.
- Captura y tratamiento de imágenes, que serán aplicadas a los modelos 3D para darles más realismo.
- Búsqueda y tratamiento de información de los comercios de la zona.
- Creación de una página Web que muestre todo lo anterior y permita realizar ciertas consultas sobre los comercios.

Cada una de estas fases necesita una información de partida determinada para poder completarse.

Para la primera fase, el diseño de los modelos 3D de los edificios e instalaciones de los complejos urbanísticos, ha sido necesario partir de una cartografía catastral, que por sus características ha sido ideal para el diseño de los modelos.

En este proyecto se ha trabajado con la cartografía catastral digitalizada del término municipal de La Puebla de Farnals, proveniente del Catastro.

Ésta se obtiene por restitución numérica a partir de un vuelo fotogramétrico, por digitalización de documentos gráficos existentes en el Catastro o tiene su origen en cartografía digital de otras Administraciones Públicas sobre la que se vuelca el parcelario [60].

Figura 3.4. Plano de catastro

La cartografía catastral tiene las siguientes características:

- Proyección UTM 30N
- Sistema Geodésico: ED50
- Escala de captura 1:1000

Y se encuentra disponible en los siguientes formatos:

- FICC: Formato de Intercambio de Cartografía Catastral, ficheros en código ASCII que necesitan de un programa de transformación.
- DXF: Formato de cartografía catastral que se puede leer por los programas CAD y SIG de uso más extendido.
- SHAPEFILE: Formato que se puede leer con la mayoría de los Sistemas de Información Geográfica.
- SVG: Es un formato XML para gráficos vectoriales, que cumple para éstos un papel similar al de JPEG para imágenes.

El formato con el que se trabajó fue DXF; formato propietario de AutoDesk. Existen dos versiones (ASCII y binario). El catastro lo suministra siempre en la versión ASCII.

La cartografía catastral consta de un único fichero con todos los elementos cartográficos (parcelas, líneas, puntos, rótulos). A continuación, algunas particularidades del DXF generado:

- Hay muy pocas primitivas (sólo LINE, POLYLINE, TEXT, POINT,...) con objeto de que pueda ser leído fácilmente por cualquier sistema.
- Cada elemento pertenece a una capa que tiene por nombre su código (tema, grupo, subgrupo) en la Codificación de la Información Geográfica Catastral.
- Los elementos no tienen color, estilo, etc., y la simbología está a nivel de capa.
- Los objetos catastrales que representan recintos (parcelas, p.ej.) se descargan como un texto con el rótulo y a continuación una polilínea cerrada con el perímetro completo.

La cartografía catastral ha sido de gran utilidad para el trabajo aquí desarrollado, ya que, aparte de la información planimétrica, incluye datos sobre las alturas de los edificios, lo que ha facilitado el trabajo a la hora de realizar el modelado 3D.

El programa utilizado para el tratamiento, comprobación y preparación de la cartografía, antes de su uso en el programa de modelado, ha sido Autodesk *AutoCAD*. La codificación de elementos en la cartografía catastral estaba en su gran mayoría bien realizada. No obstante, mediante una serie de salidas a campo se comprobó que unos cuantos habían sido codificados erróneamente. Una vez en gabinete se procedió a mover dichos elementos a las capas correctas.

Por otro lado, pese a incorporar la cartografía catastral capas con información acerca de aceras, alturas de edificios, ámbito del proyecto, nombres de calles, número de parcela, número de manzana o números de policía, éstas no fueron utilizadas en la fase de modelado 3D por no considerarse relevantes para dicho proceso, reduciéndose así un factor importante, la memoria consumida.

Por último cabe destacar que, dado que se ha buscado la sencillez de las formas para que el posterior modelado ocupara menos memoria, varios edificios que presentaban exceso de detalle han sido simplificados, sin que por ello se haya visto afectada su geometría y precisión.

Una vez preparada la cartografía catastral ha sido posible ponerse a trabajar en el diseño de los modelos 3D, proceso que se explicará al detalle más adelante. En este paso los modelos 3D presentan su acabado final a falta de texturizarlos con imágenes reales de las fachadas de los edificios.

Para eso es necesaria una segunda fase, en la que se ha procedido a la captura y tratamiento de imágenes. Esto ha hecho que los modelos 3D tengan un aspecto realista.

Antes de texturizar estas imágenes sobre los modelos 3D ha sido necesario obtener la información de partida, proceso consistente en salir a campo y tomar fotografías de los edificios correspondientes.

Una vez hecho esto, las imágenes tuvieron que ser tratadas con el fin de sacar el máximo provecho de ellas, que tuvieran una buena estética y que ocuparan una cantidad de memoria aceptable. Se emplearon para ello varias técnicas de retoque fotográfico, tratamiento panorámico, teselado, normalización y redimensión.

Una vez las imágenes estuvieron preparadas se procedió a terminar el proceso de modelado mediante la aplicación de éstas como texturas sobre los modelos 3D.

Con todo lo hecho hasta el momento, fue posible mostrar los modelos en *Google Earth*. Allí se comprobó que su apariencia era correcta y que el conjunto de modelos podían moverse por la pantalla correctamente. Esto demostró que el ahorro de memoria aplicado en todo el proceso de diseño ha sido óptimo.

Una vez nos situamos en el entorno de trabajo de *Google Earth* se procedió al que constituye el tercer paso, que es asociar información extra a la escena 3D que se ha generado. Esto fue posible gracias a que la aplicación permite localizar dicha información geográficamente.

Así pues, siguiendo los objetivos del proyecto, se procedió a crear marcas de posición que señalan la localización de todos los comercios y servicios situados en la zona de estudio. Junto a las marcas de posición fue posible ampliar la información que señalan mediante ventanas de información. Por tanto, la información de partida de este tercer paso fue toda aquella relativa a los comercios y servicios. Este proceso se hizo mediante salidas a campo, visitando los comercios, consultando publicaciones locales con inclusión de anuncios de comercios de la zona, buscando información particular y pública en Internet y portales tales como páginas amarillas. Por otra parte, como complemento a la información alfanumérica fue interesante tomar fotografías del exterior del comercio y del interior, así como escanear tarjetas de visita, para completar así la maquetación que se había ideado para la ventana de información.

Finalmente, con todo lo anterior, ya está generada toda la información que debe mostrarse bajo la API de *Google Earth*, quedando así como cuarto y último paso la programación de la página Web, que incluye las consultas que pueden realizarse sobre los comercios y los vuelos que muestran diversas partes de la zona de estudio.

Capítulo 4

Implementación de la solución

4.1. Introducción

En esta parte de la memoria se describe la implementación de la solución propuesta. Para ello se detalla el flujo de trabajo, mencionado en el anterior capítulo. Se describen los algoritmos, métodos y técnicas utilizados durante su desarrollo y se muestran las decisiones que ha habido que tomar para hacer más flexible la aplicación.

Como se menciona en capítulos anteriores, el flujo de trabajo se divide en cuatro fases: modelado 3D, captura y tratamiento de imágenes, búsqueda y tratamiento de la información y programación de la aplicación Web. Para cada fase se han utilizado una o varias aplicaciones, así como diversos tipos de archivos y lenguajes de programación. En este capítulo se describen todos ellos, sus interfaces, sus características, ejemplos de su uso y de qué manera han servido a la implementación de la aplicación.

4.2. Modelado 3D

4.2.1. Introducción

A continuación se describe el proceso de modelado 3D que se ha llevado a cabo durante el transcurso del presente proyecto. Se hace una breve presentación de la aplicación utilizada y se describen sus principales características. Una vez hecho esto se procede a describir las diferentes etapas en las que se ha divido el modelado, explicando dentro de cada una las diferentes técnicas y métodos utilizados.

4.2.2. Historia

SketchUp es una aplicación desarrollada en 1999 por @last software. Ésta se diseñó de manera que el modelado fuera sencillo y más intuitivo que otras aplicaciones, utilizando para ello elementos sencillos, como líneas y formas geométricas. Su diseño innovador e implementación flexible le hizo ganar numerosos premios. Su flexibilidad permite el uso de *plugins* para ampliar sus características. Uno de ellos permite que pueda ser integrado en *Google Earth*. En 2006 Google adquirió el software y desde entonces ha continuado su desarrollo.

A principio *SketchUp* era usado por arquitectos, pero poco después se convirtió en una herramienta indispensable para diseñadores de interior, paisajistas, diseñadores de jardines, urbanistas, directores de cine y teatro, inventores, diseñadores de juegos y demás. Hoy en día se usa tanto en el trabajo como en la escuela o en casa. Ningún otro software para modelar en 3D disponible en el mercado tiene una interfaz tan sencilla de usar, es tan fácil de aprender y a la vez tan potente como *SketchUp*.

SketchUp es un modelador de estructuras huecas o de caparazón (Apdo. 2.5.3.2.). Este método es ideal para construir modelos rápidamente, ya que trabaja únicamente con el aspecto exterior de los objetos modelados.

4.2.3. Modelado

Todas las características anteriores, junto con la posibilidad de poder mostrar los modelos en *Google Earth* y de poder usar fotografías de edificios reales para dar un aspecto más realista a los modelos, han hecho que se haya escogido este programa para construir los modelos 3D que aparecen en la aplicación final.

El proceso de modelado ha tenido una evolución en dos fases debido a un problema que se presentó:

En la primera fase los complejos urbanísticos fueron modelados en alta calidad. Esto significa que, dada la versatilidad de *Google SketchUp* y el aprendizaje progresivo en su manejo, el modelado se hizo con todo lujo de detalles, dando así un aspecto muy realista al área de estudio. Los modelos eran mostrados individualmente en *Google Earth* conforme se iban acabando y su comportamiento en pantalla era correcto, moviéndose con suavidad y presentando correctamente los detalles modelados en *SketchUp*. El problema surgió al intentar mostrar los 46 modelos a la vez en *Google Earth*. Si bien éstos aparecían representados correctamente, después de mucho tiempo de carga, su movimiento por la pantalla no era fluido. La cantidad de información contenida en los modelos era excesiva, por lo que tuvieron que ser revisados.

En la segunda fase los modelos se simplificaron de manera que el tamaño de éstos se redujera lo suficiente. Además, conforme los nuevos modelos iban siendo terminados fueron mostrados en conjunto bajo *Google Earth* para comprobar que su tiempo de carga era mucho más corto y que la aplicación era capaz de moverlos rápidamente.

Dado que este proceso de simplificación de modelos y economía de la memoria empleada afecta prácticamente a todas las etapas del modelado, se explicarán en cada una de ellas el método utilizado para la primera fase, de alta calidad, y luego las modificaciones que se hicieron, para la segunda fase, de baja calidad.

4.2.3.1. Importación de la cartografía

SketchUp utiliza un sistema 3D de coordenadas planas (X, Y, Z) en el que los puntos se identifican en el espacio mediante su posición en tres ejes de coordenadas, con valores X, Y y Z positivos y negativos.

Por otro lado, *Google Earth*, el programa en el que aparecerán finalmente representados los modelos, emplea únicamente WGS84 como sistema de referencia espacial. No trabaja con coordenadas planas, de manera que, cuando se exporta la cartografía desde *SketchUp* ésta aparece con su georreferenciación y tamaño incorrectos.

Para solucionar esto *SketchUp* permite hacer una serie de ajustes. El primero de ellos es establecer las coordenadas geográficas en WGS84 de la esquina inferior izquierda de la cartografía. Para ello se toman las coordenadas de dicho punto en UTM ED-50 y se transforman a WGS84. Esto hace que, como se ha podido comprobar, cuando se exporte para ser visto en *Google Earth* la cartografía aparezca correctamente georreferenciada.

Por otro lado, para solucionar el problema del tamaño se ha procedido de la siguiente manera: se han tomado cuatro puntos de coordenadas conocidas en la cartografía (UTM ED50) y se han calculado sus correspondientes coordenadas geográficas en WGS84. Después se han señalizado dichos puntos en *Google Earth* mediante marcas de posición. Por último se ha trabajado en *SketchUp* realizando ajustes de escala en X e Y de la cartografía, exportándola sucesivas veces a *Google Earth* hasta comprobar que encajaba con dichos puntos.

Así pues, una vez preparada la cartografía en *SketchUp* ya se ha podido empezar a modelar.

Cabe destacar que, dado que la cartografía del área de estudio ocupaba una amplia extensión como para trabajar con toda ella en un único modelo, se optó por dividir ésta según las parcelas, y en ocasiones subparcelas, que ocupan los diferentes complejos urbanísticos, plazas, parkings, paseo, etc., creándose en total 46 modelos 3D.

En el siguiente ejemplo puede verse el resultado de la importación de una de las divisiones como base para modelar el complejo Parque Granell.

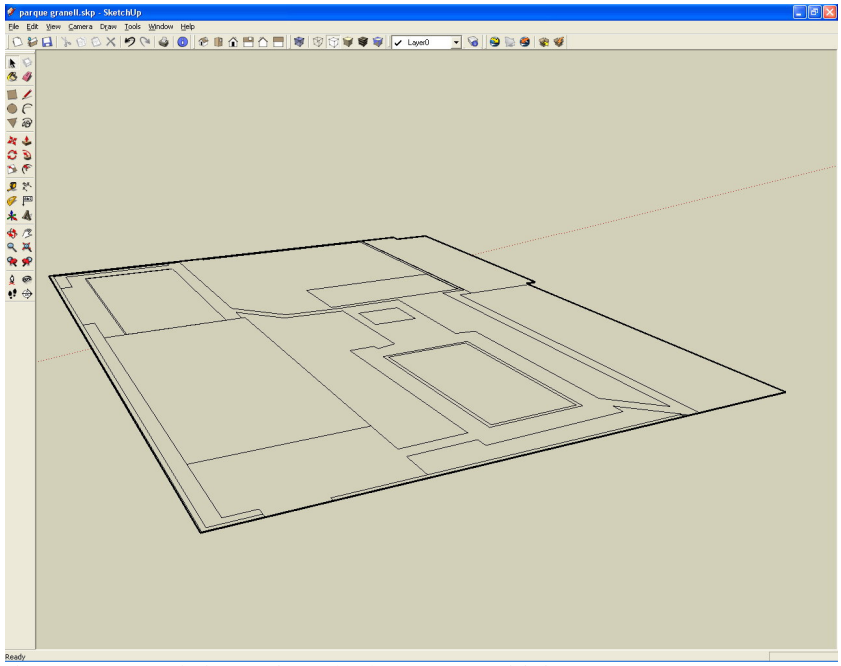


Figura 4.1. Cartografía importada en *SketchUp* para su modelado

Al importar la cartografía, *SketchUp* lo inserta como un bloque. Para poder trabajar con él es necesario descomponerlo. Esto lo convierte en un conjunto de líneas, independientes entre sí, desapareciendo de esta manera cualquier polilínea o polígono que se hubiera generado en *AutoCAD*.

Para el paso de la fase de alta a la de baja calidad, en esta primera etapa no hubo que hacer modificaciones en la cartografía, dado que se trataba de la base del trabajo.

4.2.3.2. Preparación de la cartografía

En esta etapa se preparó la cartografía de manera que se pasó de tener un conjunto de líneas independientes entre sí a tener una superficie cubierta por diversos polígonos que representan los diferentes elementos del complejo. El aspecto final de esta etapa difiere ligeramente del de la anterior (Fig. 4.2), apareciendo en color azul los nuevos polígonos generados.

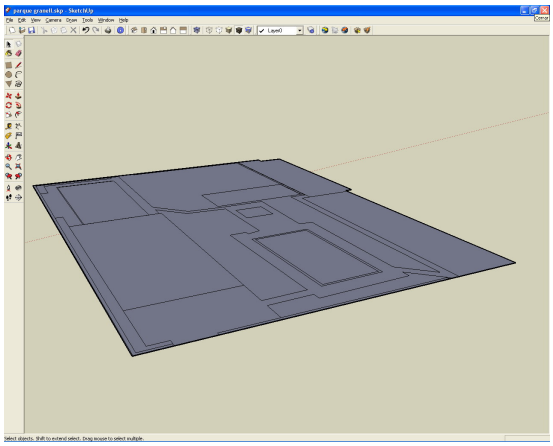


Figura 4.2. Polígonos que servirán de base para el modelado

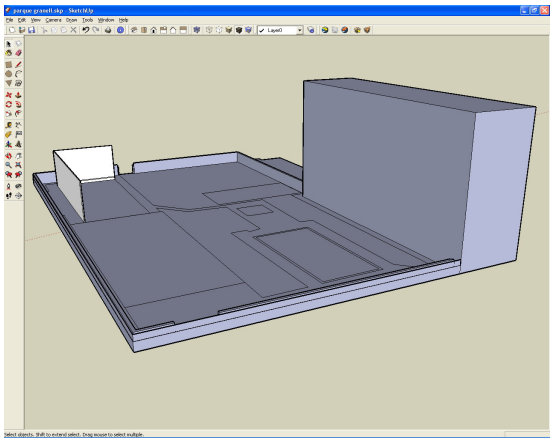


Figura 4.3. Diferentes elementos extrudidos

Para esta etapa tampoco hay diferencia entre la fase de modelado en alta calidad y la de baja calidad. Se trata simplemente de preparar la cartografía para el modelado posterior.

4.2.3.3. Extrusión

En el modelado 3D la extrusión consiste básicamente en generar un prisma a partir de una determinada base, i.e., a partir de un polígono rectangular se puede generar un prisma de base rectangular mediante su extrusión.

Así pues, en esta etapa se extrudieron aquellos elementos que debían representar volúmenes, como son edificios, paredes, vallas, muros, setos, etc. La altura de cada edificio se determinó consultando el dato "número de plantas" que proporciona la cartografía catastral, habiéndose establecido previamente una altura de 3 metros para cada planta. La altura del resto de los elementos fue determinada a estima. En la figura 4.3 puede verse el aspecto general que va a tener el complejo de referencia una vez aplicada la extrusión.

En la fase de alta calidad no se omitió ninguna característica de los elementos, abundando en el detalle. En la figura 4.4, perteneciente a dicha fase, puede comprobarse como el perímetro del complejo presenta un muro, una verja (en blanco) y un seto que sobresale por encima de la valla, cada uno con un espesor diferente; todos ellos generados mediante extrusión. A efectos técnicos esto representa el uso de una gran cantidad de polígonos. Además, el detalle de la verja (se aplicará una textura transparente) supone el uso de una textura más, con el gasto en memoria que ello conlleva.

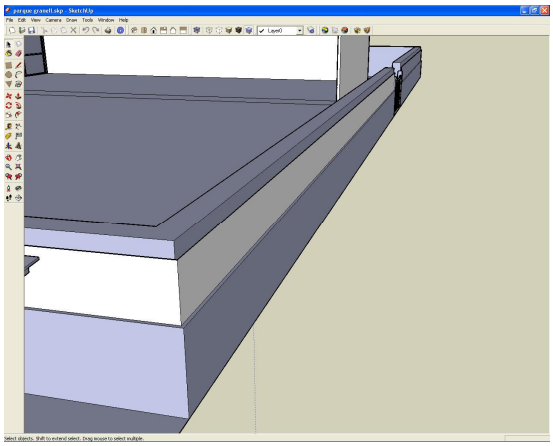


Figura 4.4. Perímetro del complejo en alta calidad

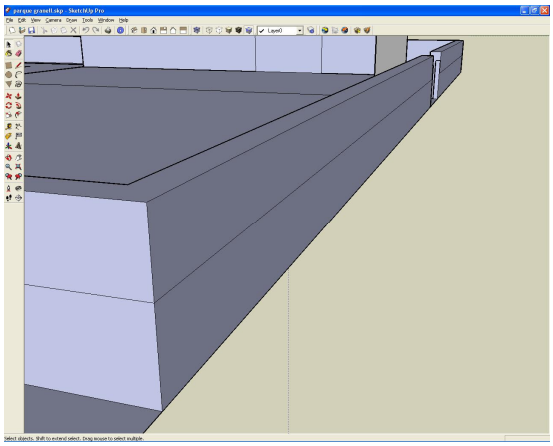


Figura 4.5. Perímetro del complejo simplificado

En la figura 4.5 se presenta la anterior escena en la fase de baja calidad. En ella se simplificaron las formas y se intentó suprimir el mayor número de polígonos posible sin que las formas y la interpretación de los elementos se vieran afectadas. Esto ahorró mucha memoria. Además la supresión de detalles como es el caso de la verja fue de gran ayuda. Cabe destacar que la representación de elementos transparentes en *Google Earth*, que es la aplicación final a la que están destinados estos modelos, es inapreciable y en ocasiones inexistente a día de hoy, por lo que la supresión de elementos de este tipo, como es el caso de esta verja, no ha tenido efectos en su acabado final.

En la figura 4.6, de la fase de alta calidad, puede verse que las paredes de la pista de frontón tienen grosor.

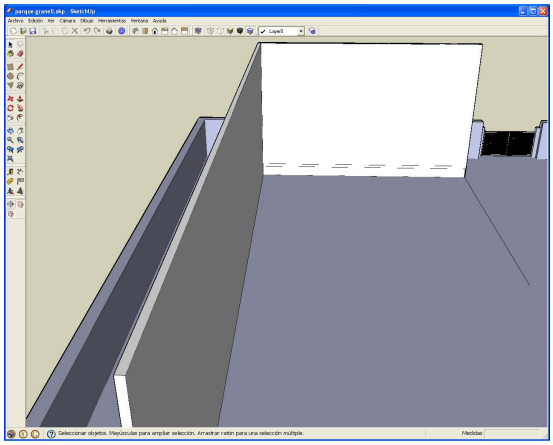


Figura 4.6. Paredes del frontón con grosor

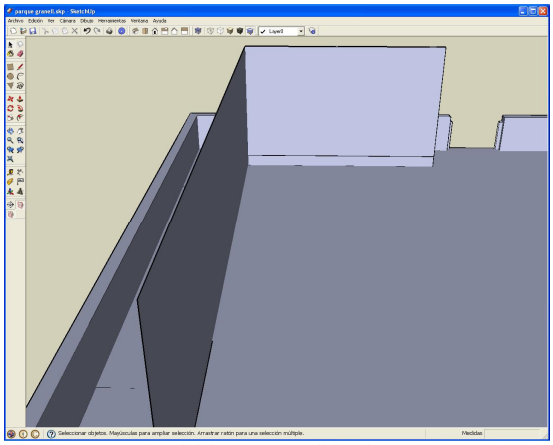


Figura 4.7. Paredes del frontón simplificadas

La figura 4.7, de la fase de baja calidad, presenta por el contrario una pista de frontón con unas paredes sin grosor. Esta simplificación reduce el número de polígonos empleado y no afecta a su aspecto final, interpretándose claramente que se trata de una pista de frontón.

4.2.3.4. Texturizado

El proceso de modelado continúa con el texturizado de todos los elementos, a excepción de la fachada del edificio, cuyo texturizado será descrito en el apartado 4.2.3.6.

SketchUp dispone de una amplia gama de texturas para cada tipo de material. En la fase de modelado en alta calidad no se escatimaron recursos en el empleo de texturas, aplicándose una gran variedad, como puede advertirse en la siguiente imagen:

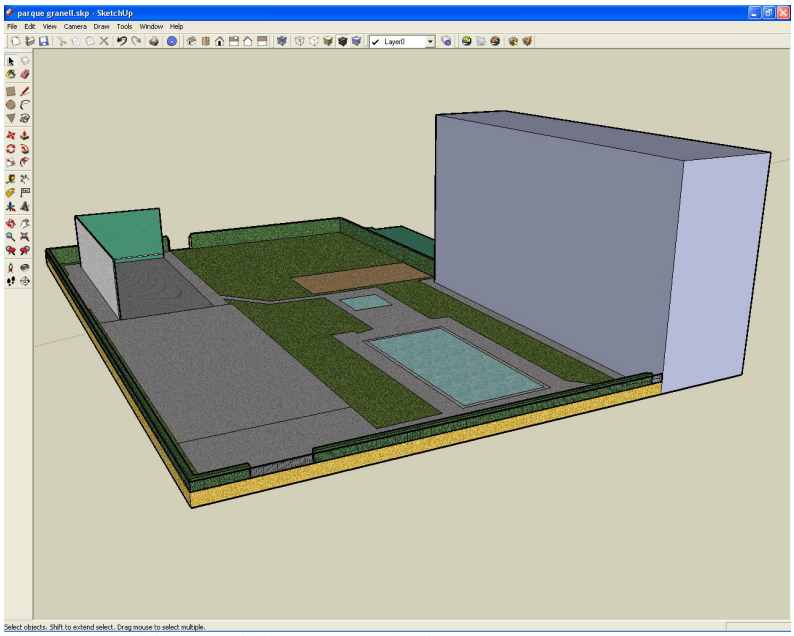


Figura 4.8. Modelo con múltiples texturas aplicadas

Para representar la vegetación se emplearon diferentes texturas para cada tipo de césped y seto. También se utilizaron diferentes texturas para las diferentes clases de suelo, como fueron asfaltos, gravillas, azulejos, cementos, ladrillos, etc. Posteriormente, en esta misma fase, algunas de ellas fueron descartadas por consumir excesiva memoria.

Para la representación del agua de las piscinas se utilizó una textura de agua que era transparente, permitiendo así que se pudieran ver las teselas que recubrían todo el interior de la piscina. En la figura 4.9 puede apreciarse este efecto.

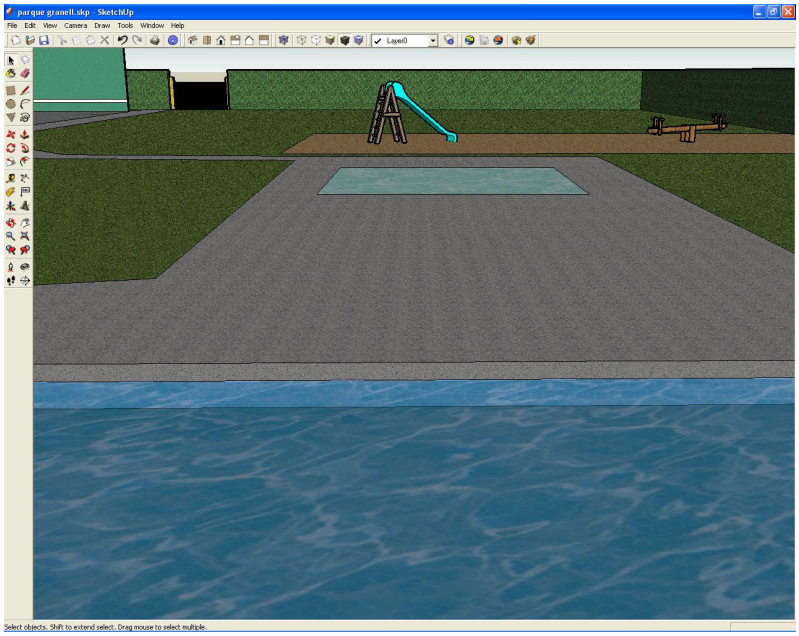


Figura 4.9. Detalle de la textura transparente para el agua en la piscina y en el fondo de ésta, la textura de teselas aplicada a sus paredes

En la fase de baja calidad se analizó la memoria que ocupaba cada una de las texturas y, aún cuando este modelo por sí solo ocupaba un tamaño razonable, se intentó reducir éste usando otras texturas que hicieran el mismo papel y no supusieran una gran modificación de su aspecto.

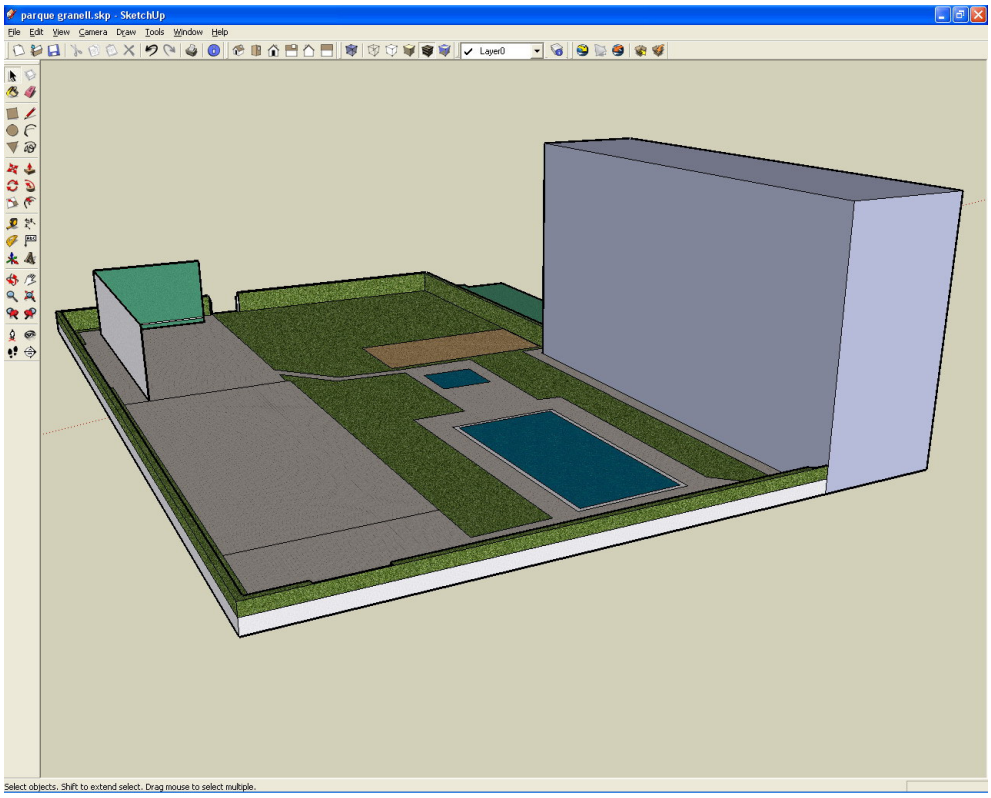


Figura 4.10. Modelo con texturas simplificadas

En cuanto a la vegetación, puede observarse cómo se aplicó una misma textura tanto para el césped como para los setos. Se utilizó la que menos memoria ocupaba y a la vez representaba mejor estos elementos. En la piscina se suprimió la profundidad de la misma, reduciéndose así el número de polígonos y prescindiendo de la textura de teselas que decoraba el interior de la misma. Para representar su superficie se eligió la textura que menos memoria consumía dentro de la gama de tipos de agua, dando un buen resultado. Para el muro que abarca el perímetro del complejo se cambió la textura de ladrillos amarillos, que ocupaba mucha memoria, por una textura de escayola blanca, que era más económica y no quedaba mal. Por último, como se ha comentado más arriba, la textura transparente que representaba la verja fue suprimida por los motivos antes mencionados, ahorrándose así más memoria.

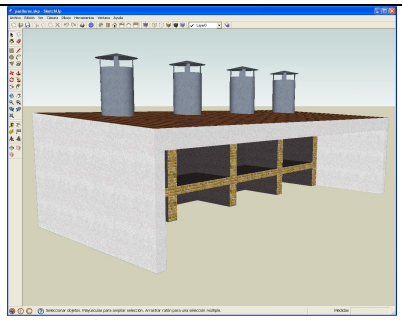
4.2.3.5. Componentes

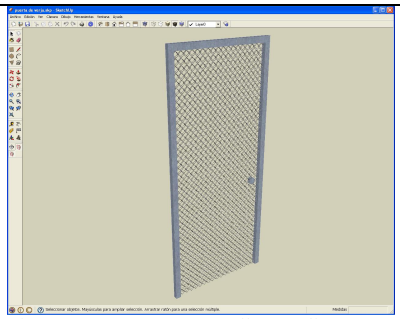
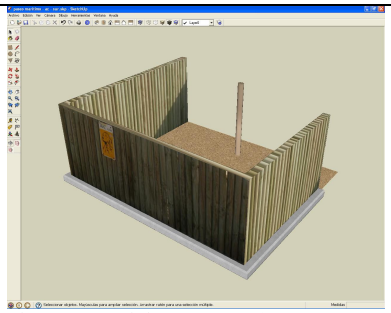
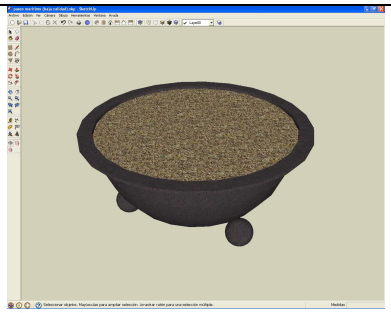
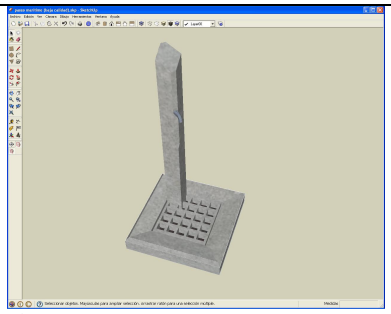
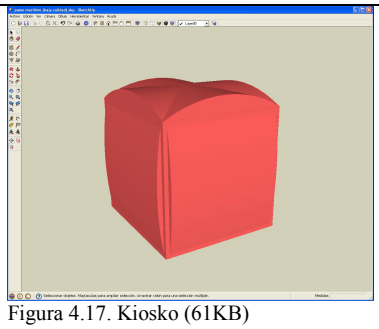
En todos los complejos existen multitud de elementos que, por ser de uso común, aparecen siempre repetidos, incluso dentro de un mismo complejo. Generalmente se trata de paelleros, pistas de tenis, de pádel, estructuras metálicas que cubren parkings privados, balancines, toboganes, puertas de acceso, verjas, cancelas, puertas automáticas de parkings, etc. *SketchUp* los denomina componentes, siendo éstos modelos independientes que pueden ser importados dentro de otro modelo. Son algo parecido a los bloques empleados en *AutoCAD*. Su uso ahorra mucho tiempo, al no tener que crearlos desde cero cada vez que aparecen en un complejo.

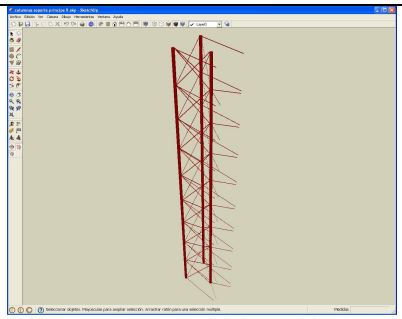
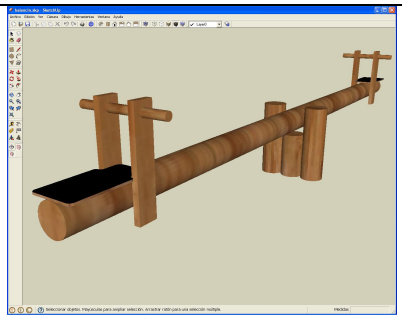
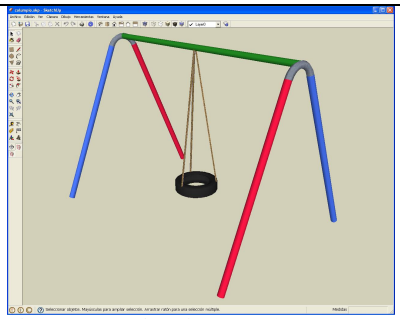
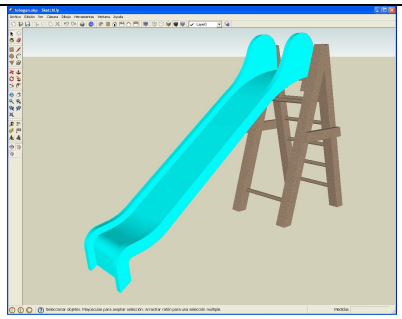
Como se comentó en el apartado 2.5.3.4, *SketchUp* dispone de un almacén en Internet llamado *3D Warehouse*, desde donde el usuario puede acceder de forma pública y gratuita a modelos 3D creados con *SketchUp* por otros usuarios. Aprovechando esta característica y para la fase de modelado en alta calidad, se utilizaron varios de estos modelos 3D, entre ellos pistas de pádel, columpios, escaleras, marquesinas de autobús, embarcaciones, pistas de tenis, porterías de fútbol, canchas de baloncesto, depósitos de agua y gas, etc.

Cabe destacar que el propio *SketchUp* tiene en sus directorios un buen número de modelos clasificados por categorías que pueden aprovecharse. Además descargando desde la Web de la aplicación una serie de archivos puede ampliarse el número de estos con nuevos modelos, así como con nuevas texturas. De entre los componentes incluidos en *SketchUp* se utilizaron puertas, ventanas, verjas, farolillos, árboles, farolas, bancos, vehículos, cabinas telefónicas, etc.

No obstante, también se tuvo que modelar componentes propios por no existir éstos ni en la *3D Warehouse* ni en *SketchUp*. He aquí algunos ejemplos:

Estructura para parking cubierto	 Figura 4.11. Parking cubierto (43KB)
Barbacoa	 Figura 4.12. Barbacoa (182KB)

Puerta con malla metálica	 Figura 4.13. Puerta con malla metálica (47KB)
Pipi-can	 Figura 4.14. Pipi-can (148KB)
Macetero	 Figura 4.15. Macetero (221KB)
Fuente	 Figura 4.16. Fuente (97KB)
Kiosko	 Figura 4.17. Kiosko (61KB)

Estructuras de soporte para el edificio Presidente V	 Figura 4.18. Soporte Edificio Presidente V (71KB)
Balancín	 Figura 4.19. Balancín (76KB)
Columpio	 Figura 4.20. Columpio (180KB)
Tobogán	 Figura 4.21. Tobogán (121KB)

A continuación se muestra el aspecto que presenta el complejo una vez colocados los diferentes componentes en la fase de alta calidad:

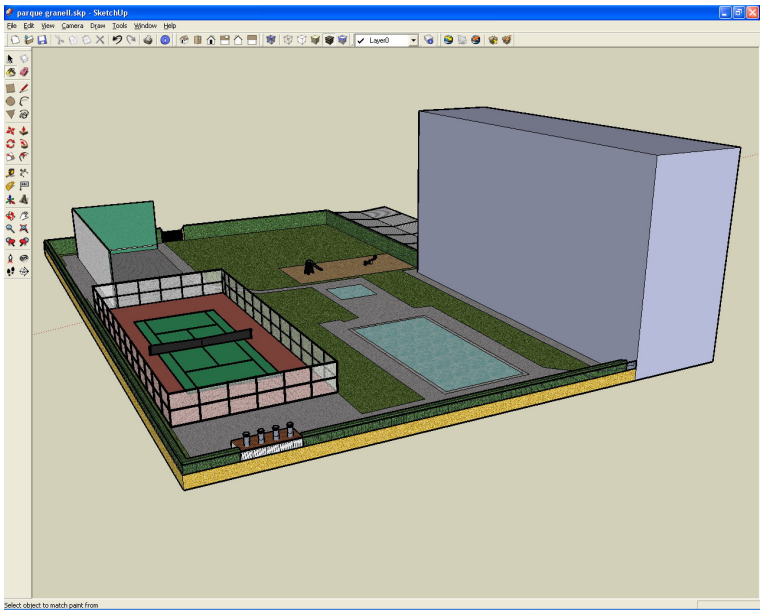


Figura 4.22. Complejo modelado con componentes en alta calidad (811KB)

La pista de tenis, el parking cubierto, el paellero, los columpios, las puertas del complejo; todos ellos son componentes y pueden verse repetidos en otros complejos.

En la fase de modelado en baja calidad algunos de estos componentes fueron modificados. Su aspecto y acabado final original era muy bueno, pero ocupaban mucha memoria. El proceso de remodelado consistió en simplificar sus formas, reduciéndose así el número de polígonos utilizado. Por otro lado también se redujo el número de texturas y se cambiaron las que ocupaban mucha memoria por otras más económicas. En los siguientes ejemplos se pueden apreciar varios de los cambios realizados:

Estructura de parking cubierto.

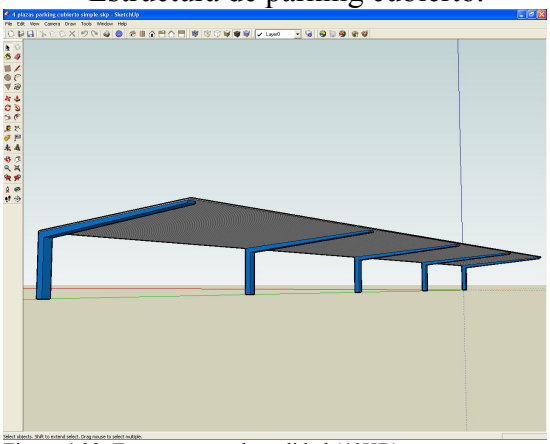


Figura 4.23. Estructura en alta calidad (40KB)

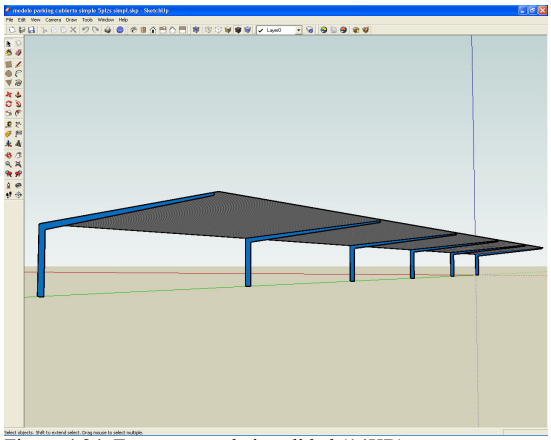


Figura 4.24. Estructura en baja calidad (14KB)

En éstos se ha eliminado el espesor tanto de los soportes como del techo de Uralita, ahorrándose así gran cantidad de polígonos y texturas asociadas a ellos. El diseño original ocupaba 39.7KB y la modificación pasa a ocupar 13.8KB, siendo éste un ahorro considerable, teniendo en cuenta que algunos complejos tienen varias zonas de parking cubierto con esta estructura.

Columnas de soporte para el edificio “Príncipe V”.

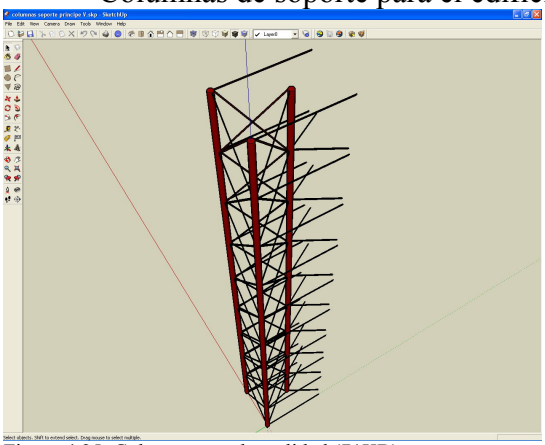


Figura 4.25. Columnas en alta calidad (71KB)

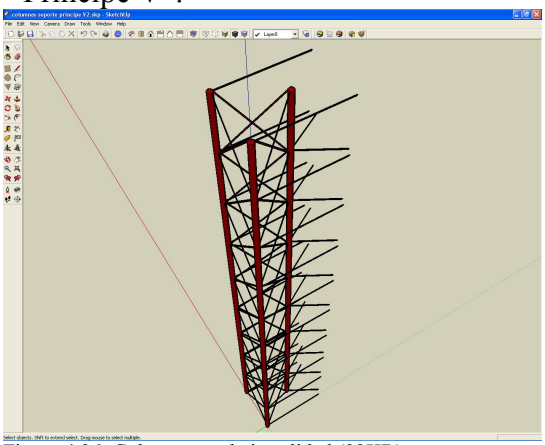


Figura 4.26. Columnas en baja calidad (32KB)

En este ejemplo la diferencia es muy sutil y apenas se aprecia debido al tamaño de las imágenes. Las columnas pasan a tener menos polígonos y para todo el conjunto se prescinde de la textura metálica roja usada en un principio y que ocupa bastante memoria, para pasar a utilizar una tonalidad rojiza, mucho más sencilla, permitiendo así ahorrar más memoria. El diseño original ocupaba 70.5KB y el modelo rediseñado pasa a ocupar 31.9KB. Teniendo en cuenta que el complejo “Príncipe V” tiene tres de estas estructuras, esto supone un ahorro de cerca de 116KB.

Paelleros.

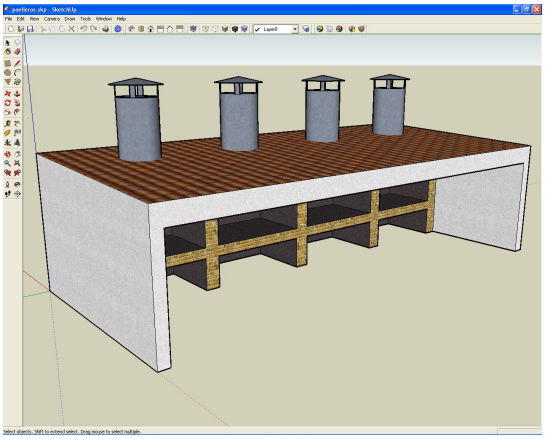


Figura 4.27. Paellero en alta calidad (182KB)

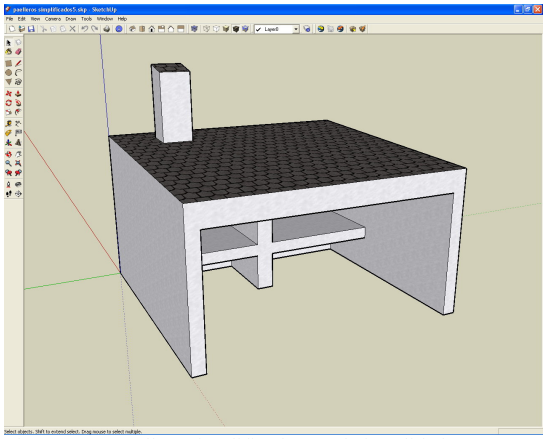


Figura 4.28. Paellero simplificado para baja calidad (24KB)

Este componente ocupa, en su versión de alta calidad, 182KB. Teniendo en cuenta que muchos complejos ocupan esa cantidad de memoria al final del proceso de baja calidad, o incluso menos, fue necesario simplificarlo. En el proceso de remodelado se pasó de usar cinco texturas de gran tamaño a dos texturas mucho más sencillas. En el componente de alta calidad se puede apreciar la suavidad en la curvatura de los cilindros que daban forma a las chimeneas. Esto es debido a que en un principio se utilizaron cilindros de 24 caras. En la fase de baja calidad pasaron a ser 4 caras y una sola chimenea. Al final del proceso el componente pasó a ocupar 24KB. Teniendo en cuenta que un gran número de complejos tiene este componente esto supone a la larga un gran ahorro de memoria.

Por último cabe mencionar que hay dos componentes para los que en un principio se usaron los modelos existentes en la *3D Warehouse*. Estos son las pistas de tenis y las de pádel, ocupan 118KB y 105KB respectivamente y están muy bien realizadas pero, como se menciona más arriba, se necesitaba que ocuparan bastante menos, dado que se trata de un componente que aparece repetido en muchos complejos.

En un principio se pensó en modificar estos modelos, pero se acabó comprobando que era más sencillo construirlos desde cero, de la manera más sencilla posible. Como curiosidad cabe mencionar que ambas pistas fueron construidas usando sus medidas oficiales, disponibles en sus respectivas federaciones nacionales.

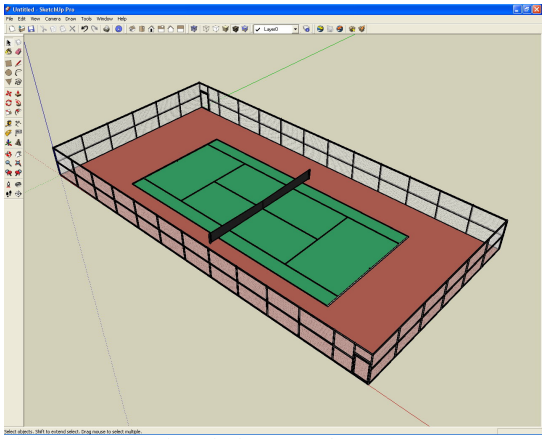


Figura 4.29. Pista de tenis de 3DWarehouse (118KB)

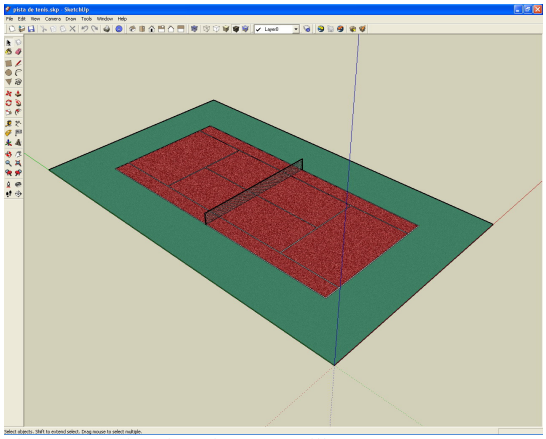


Figura 4.30. Pista de tenis más sencilla (59KB)

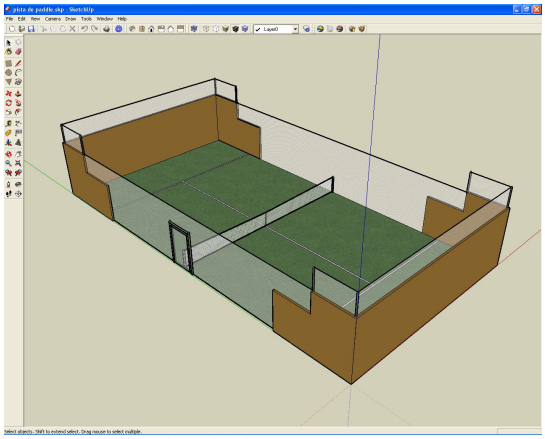


Figura 4.31. Pista de pádel de 3DWarehouse (105KB)

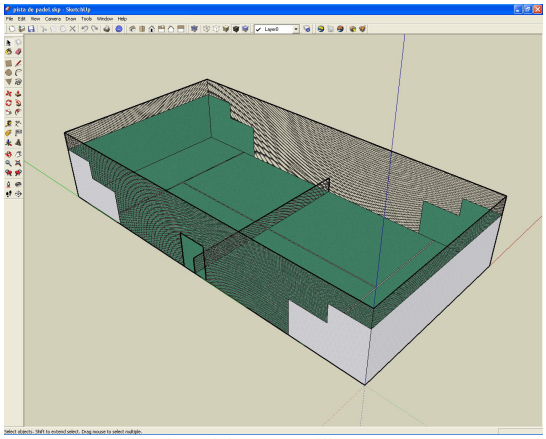


Figura 4.32. Pista de pádel más sencilla (47KB)

El resultado final de ambas es muy bueno. Se usa una menor cantidad de polígonos (las paredes no tienen espesor) y de texturas (que además son más sencillas) reduciéndose así la cantidad de memoria utilizada (Fig. 4.30 y 4.32).

No obstante, en algunos complejos, en los que se quiso ajustar más todavía la cantidad de memoria utilizada, la pista de pádel fue modificada de manera que la textura de malla metálica pasó a ser una malla de color rojo, que ocupa todavía menos memoria, o se prescindió de ella, como puede comprobarse en los siguientes ejemplos:

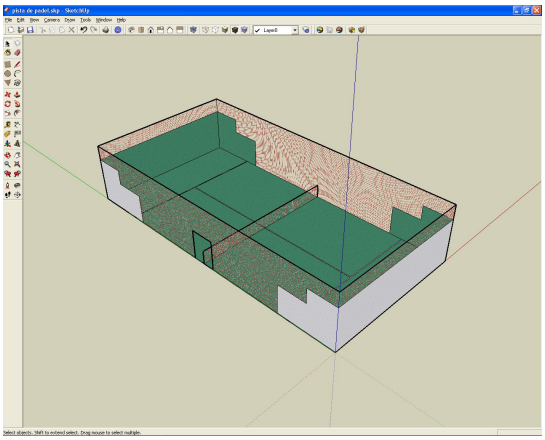


Figura 4.33. Pista de pádel con texturas más económicas (32KB)

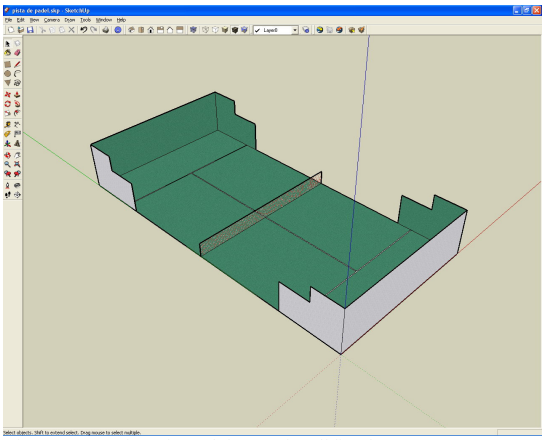


Figura 4.34. Pista de pádel más simplificada (30KB)

Por último, otros componentes, que eran más bien decorativos, fueron directamente suprimidos. Es por eso que en el complejo que se está mostrando aquí, en su versión de baja calidad, puede verse cómo algunos de los componentes han desaparecido y otros se han mantenido.

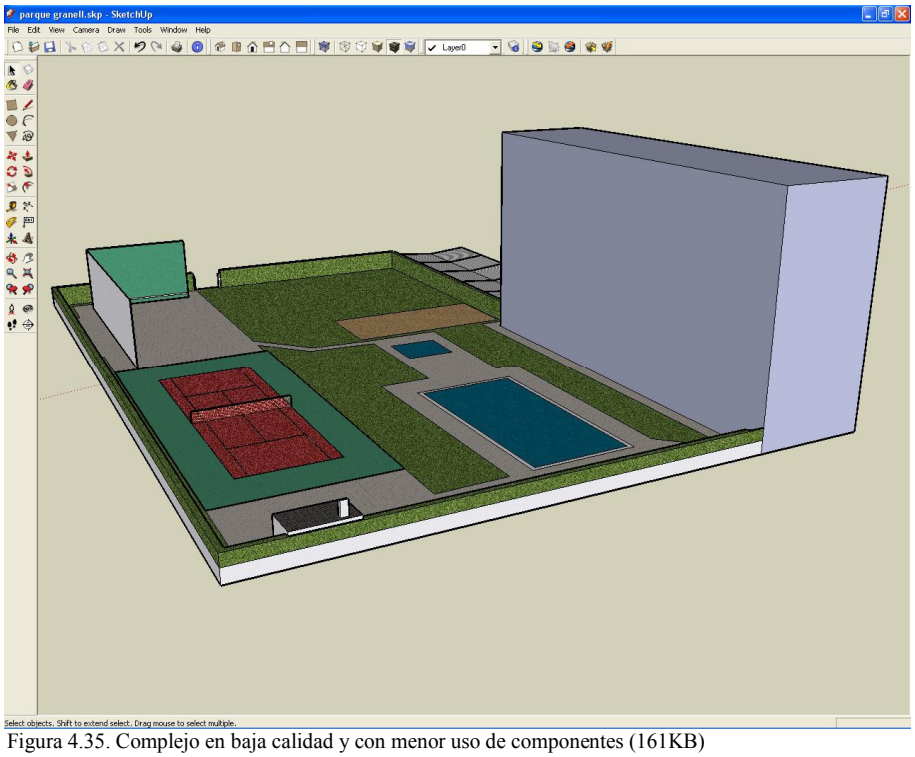


Figura 4.35. Complejo en baja calidad y con menor uso de componentes (161KB)

Las puertas exteriores del complejo, el tobogán y el balancín no aparecen. La pista de tenis, el paellero y la estructura para parking cubierto han sido substituidas por otras que han sido modeladas usando menos memoria. El acabado final sigue siendo bastante bueno.

4.2.3.6. Texturizado de fachadas

La última fase del modelado de alta calidad es el texturizado de las fachadas del edificio. Éste no funciona exactamente como se ha visto hasta ahora, aunque guarda cierto parecido.

Para aplicar las texturas a las fachadas del edificio en el modelo antes debemos disponer de fotos reales de las mismas. Para ello se salió a campo y se fotografiaron todas las fachadas de los edificios que fueron posibles. Para las azoteas se recurrió a imágenes de las mismas disponibles en *Google Earth*. Este proceso se explica en el apartado 4.3.2.

Por ejemplo, para la fachada Norte y la fachada Este del complejo con el que se está trabajando se tomó la siguiente fotografía.

Figura 4.36. Fotografía original, a partir de la cual se extraen las fotografías que servirán de fachada en el modelo.

De ésta se extrajeron las fachadas con las que se iba a trabajar. De esta manera se evita trabajar con una foto grande, ahorrándose así bastante memoria. Las fotografías obtenidas fueron las siguientes:

Figura 4.37. Fachada Norte

Figura 4.38. Fachada Este

A continuación se importaron en *SketchUp* como texturas y se indicó en qué cara del edificio en el modelo iban situadas.

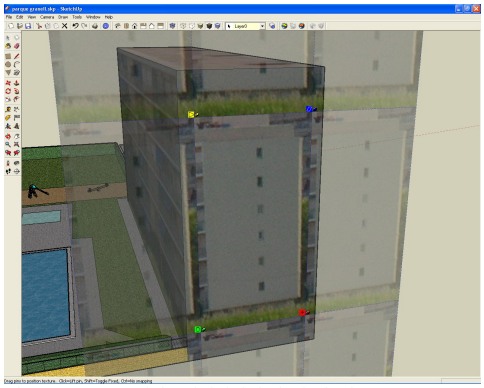


Figura 4.39. Posición inicial de la fachada Norte

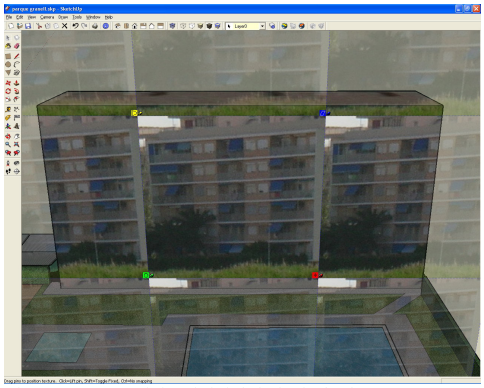


Figura 4.40. Posición inicial de la fachada Este

El proceso trata de buscar puntos homólogos entre la imagen y el modelo 3D, aplicando diferentes traslaciones, giros, escalas y deformaciones.

Primero aplicamos una traslación. Para ello se busca un punto conocido en la imagen y se traslada éste hasta su homólogo en el modelo mediante el icono  . En este caso se escogió la esquina inferior derecha de la fachada.

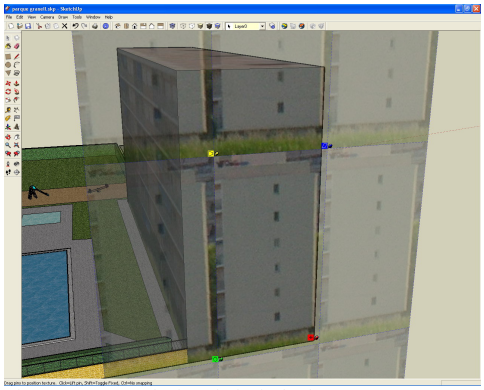


Figura 4.41. Traslación de la fachada Norte

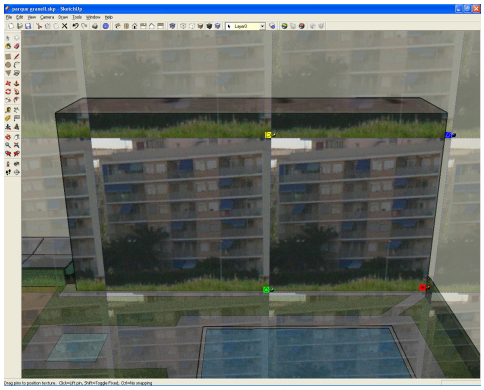


Figura 4.42. Traslación de la fachada Este

A continuación se aplica un giro (si es necesario por las condiciones de la imagen) y un escalado de la imagen a la vez. Para ello se identifica de nuevo un punto conocido de la imagen, y con el icono  se arrastra hasta su homólogo en el modelo 3D. En esta ocasión se escogió otra esquina de la fachada, opuesta a la anterior.

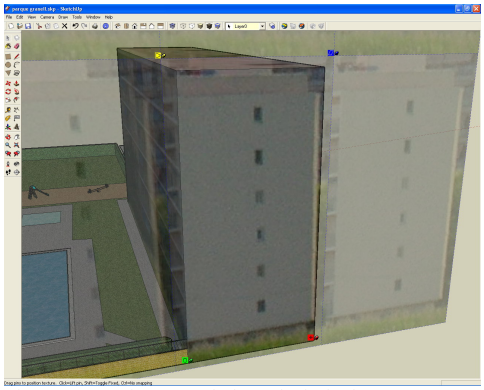


Figura 4.43. Giro y escalado en la fachada Norte

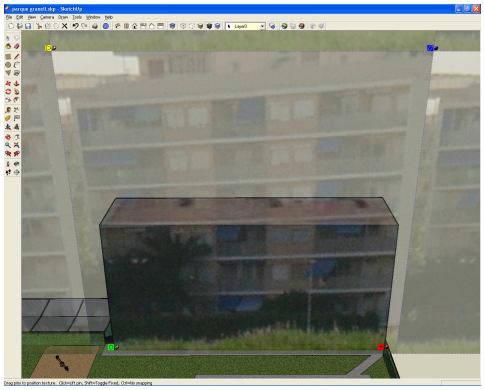


Figura 4.44. Giro y escalado en la fachada Este

En las imágenes anteriores puede observarse cómo, si bien en el primer ejemplo la fachada queda casi ajustada, en el segundo ejemplo, con el ajuste de escala anterior la fachada ha sobrepasado la altura del modelo. Esto puede ajustarse mediante la deformación en cizalla. Esta permite deformar la fotografía manteniendo siempre paralelos sus lados opuestos. Para ello elegimos con el icono  un punto conocido de la imagen (la esquina superior derecha) y lo arrastramos hasta su homólogo en el modelo.

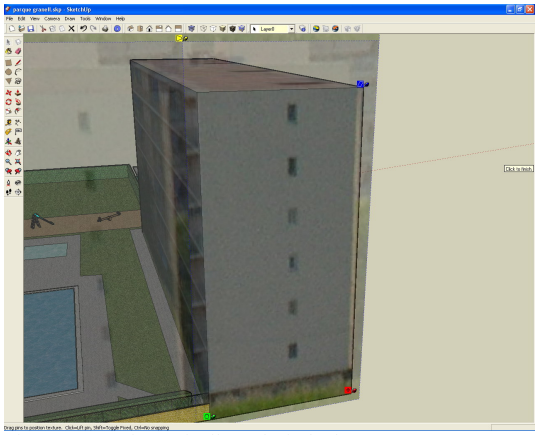


Figura 4.45. Efecto cizalla en la fachada Norte

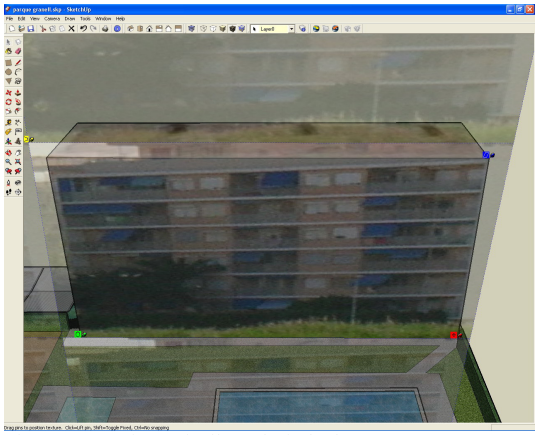


Figura 4.46. Efecto cizalla en la fachada Este

Con esta última acción la fachada suele quedar ajustada en su posición, como es el caso del primer ejemplo. Sin embargo, en el segundo aún falta por hacer un último ajuste, que se denomina distorsión. Este permite deformar la imagen manteniendo en su sitio los ajustes realizados hasta el momento. Para ello elegimos de nuevo un punto conocido de la imagen con el icono  y lo arrastramos hasta su homólogo en el modelo, quedando así ajustada finalmente la imagen al modelo.

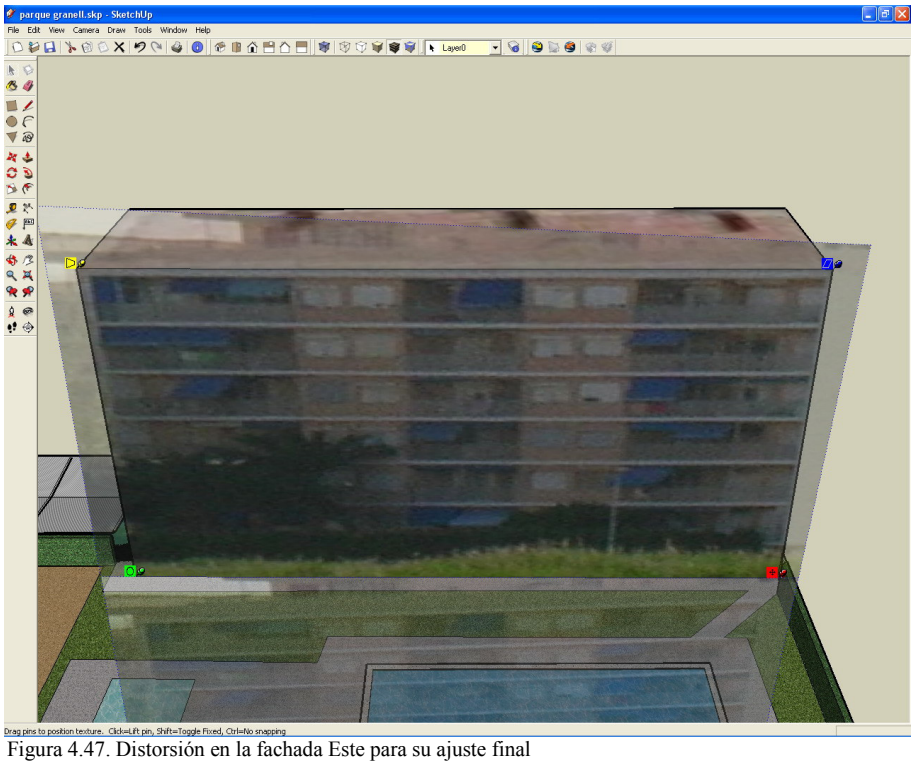


Figura 4.47. Distorsión en la fachada Este para su ajuste final

El aspecto final del complejo al final del modelado con *SketchUp* es el siguiente:

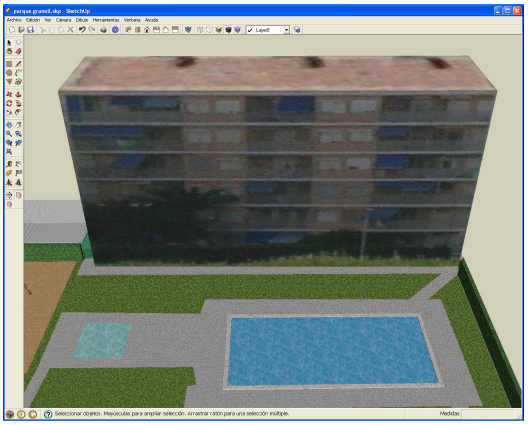


Figura 4.48. Detalle de la fachada Este en alta calidad

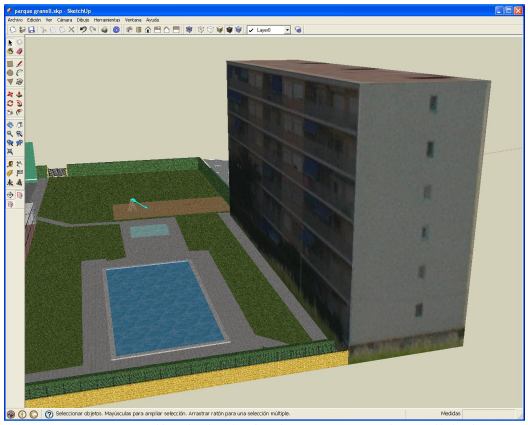


Figura 4.49. Detalle de la fachada Norte en alta calidad

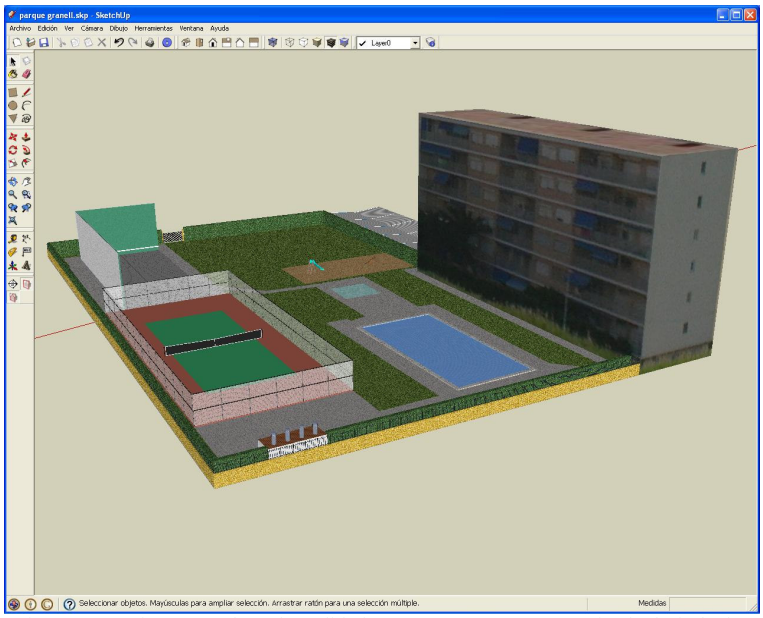


Figura 4.50. Vista general en alta calidad, con componentes y texturizado de fachadas

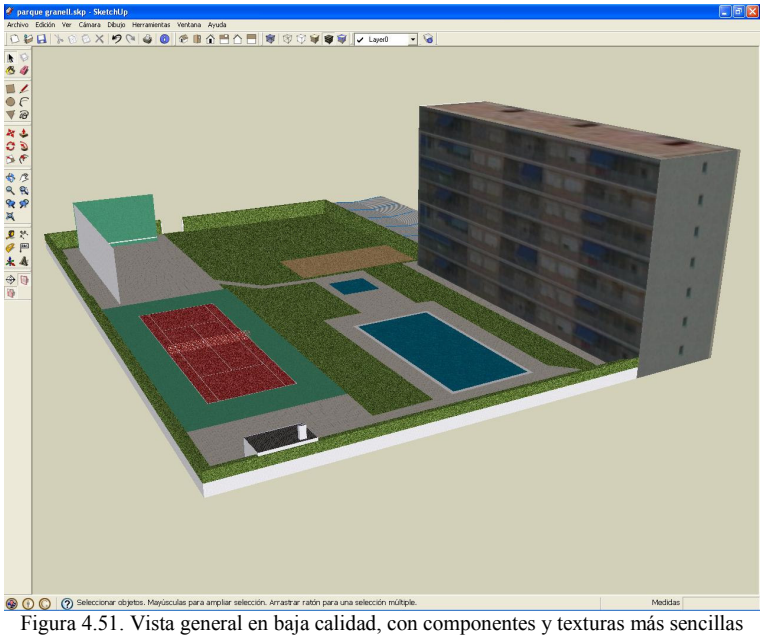


Figura 4.51. Vista general en baja calidad, con componentes y texturas más sencillas

4.3. Captura y tratamiento de imágenes

4.3.1. Introducción

El proceso de captura y tratamiento de las imágenes es muy importante ya que es el que ha permitido dar un acabado realista a los modelos 3D, completando de esta manera su modelado.

A continuación se exponen las diferentes fuentes de las que proceden dichas imágenes. Después se describen los diferentes métodos y técnicas que se han empleado para su tratamiento fotográfico, i.e., tratamiento panorámico, retoque mediante photoshop, teselado, normalización y redimensión de imágenes. Y por último se explican diversos métodos que se han aplicado para ahorrar algo más de memoria en las imágenes.

4.3.2. Captura de imágenes

Como se explica en el apartado anterior, para poder texturizar las fachadas del modelo fue necesario salir a campo y tomar fotografías de los edificios reales. El uso de estas imágenes es el motivo principal de que los modelos presenten un acabado final tan realista.

Toma directa

La mayor parte de las fotografías fueron tomadas a pie de calle con una cámara digital compacta Nikon Coolpix 4200, de 4Mp. Con esta resolución hubo más que suficiente para usar las fotos como texturas, teniendo en cuenta que más adelante, en la fase de baja calidad, fueron reducidas de tamaño. Éste fue el principal compromiso en el proceso de tratamiento de las imágenes: que cada una fuera lo más pequeña posible pero lo suficientemente grande como para poder identificar los elementos de la misma.

Otras fotografías fueron tomadas desde las azoteas de los edificios más altos de la Puebla de Farnals y de la Playa del Puig. Esto permitió tener fotografías de fachadas completas, esto es, sin elementos que interfirieran, como suele suceder con árboles, muros, setos e incluso otras edificaciones.

Otras fuentes

A parte de la toma directa de fotografías se emplearon otras fuentes, como Microsoft *Bing Maps* [47], que además de ofrecer imágenes de fotografías aéreas cenitales, como hace *Google Earth*, también ofrece fotografías oblicuas desde los cuatro puntos cardinales. Estas imágenes se emplearon en aquellos modelos que presentan edificios con unas fachadas tan grandes que es imposible obtenerlas al completo sin elementos que interfieran, ni a pie de calle ni desde otros edificios, como ocurre con los complejos “Presidente”, “Presidente Lux”, “Torre Estudio III” y otros.

A las fotografías que ofrece *Bing Maps* también se les ha aplicado una reducción de su calidad hasta coincidir con la de las fotografías utilizadas como texturas a partir de tomas a pie de calle.

Figura 4.52. Fachada principal de "Torre Estudio II" en *Bing Maps*


Figura 4.53. Azoteas en *Google Earth*

Por último, otra fuente de fotografías que se ha utilizado es *Google Earth* (Fig. 4.53) de donde se extrajeron la mayoría de las fotografías de las azoteas de los edificios.

4.3.3. Tratamiento de imágenes

En el tratamiento de las imágenes, igual que ha ocurrido en todo el proyecto, han tenido lugar dos fases, una de alta calidad y otra de baja.

4.3.3.1. Fase de alta calidad

En la fase de alta calidad no se tuvo demasiado en cuenta el tamaño final del archivo de cada modelo. Esto provocó que, como se explicó en el apartado anterior, al intentar presentar los 46 modelos en *Google Earth*, éstos se movieran lentamente. El motivo de este problema no fue sólo el modelado en alta calidad de los modelos sino también que las fotos, pese a haber sido reducidas de tamaño, aún utilizaban suficiente memoria como para impedir el buen funcionamiento de la animación.

En esta primera fase apenas se estimó necesario tratar las fotografías. Todas y cada una de las fachadas individuales tuvieron su fotografía particular, sin apenas repetir una imagen en otra fachada para ahorrar memoria. Se pretendía que la apariencia fuera tan fiel a la realidad como fuera posible, pero fue excesivo. Otro detalle que también cuenta mucho en el derroche de memoria que hubo fue que las fotografías no fueron rectificadas antes de ser importadas como texturas. El uso de imágenes rectificadas permite ahorrar mucha memoria, ya que no se desperdicia un solo pixel con información ajena a la fachada. Muchas de las fachadas no podían ser fotografiadas justo desde enfrente de manera que quedaran lo más rectificadas posible. La mayoría de las veces, debido a la presencia de elementos que interferían en la imagen, se hizo la toma desde un ángulo de visión que permitiera sacar la fachada completa con el menor número de obstáculos posible, con la idea de rectificarlas posteriormente.

Figura 4.54. Toma oblicua de una fachada con obstáculos

En la figura 4.54 puede apreciarse la toma de una fachada bajo un ángulo muy cerrado, buscando la menor cantidad posible de árboles que la tapen y a su vez dejando ver la fachada completa. La idea posterior era rectificar esta imagen dentro de *SketchUp* con las herramientas de edición de texturas, tal y como está descrito en el apartado 4.2.3.6. El resultado final era aceptable. El problema cuando no se ha rectificado la fotografía antes de ser importada en *SketchUp* es que, aunque la recortemos lo más posible y la redimensionemos hasta que ocupe una cantidad de memoria aceptable, vamos a tener una parte de la foto, en este caso el cielo, que no va a ser utilizada y que va a ocupar memoria inútilmente. En la figura 4.54 puede observarse que la parte no aprovechable es prácticamente la mitad de la foto. Este problema no se detectó y solucionó hasta más adelante cuando se vio la necesidad de reducir el consumo de recursos.

El tamaño medio de las fotografías empleadas como fachadas en la fase de alta calidad fue de unas 24KB, siendo el Edificio Madeira, Estudio II y Presidente Lux los que usaron más memoria de todos, con 53KB, 85KB y 62 KB de media por foto respectivamente.

Fotografías panorámicas

Por otra parte, para obtener las imágenes de algunas fachadas completas fue necesaria la confección de fotografías panorámicas. Esto fue debido principalmente a que no podían hacerse desde más distancia, dada la proximidad de otros edificios o porque eran fachadas muy altas y no podían hacerse desde lejos sin evitar la presencia de otros elementos, incluyendo edificios, obstaculizando la toma.

Para obtener la fotografía panorámica se realizaron tomas solapadas de las diferentes fachadas. Estas fueron tratadas con una aplicación llamada *Hugin* [151], que crea una fotografía panorámica final a partir de dichas fotografías solapadas.

Figura 4.55. Conjunto de fotografías solapadas para generar la fotografía panorámica

Figura 4.56. Fotografía panorámica generada a partir de las fotografías solapadas

En la foto anterior puede apreciarse cómo cuando se obtiene la fotografía justo desde enfrente de la fachada ésta puede aprovecharse más y aparecer prácticamente rectificada.

El programa *Hugin* permite aplicar diferentes sistemas de proyección a la fotografía panorámica resultante; gnomónica, panorámica, cilíndrica equidistante, ojo de pez, estereográfica, Mercator, transversa Mercator, sinusoidal, cónica equivalente de Lambert, acimutal equivalente de Lambert, cónica equivalente de Albers y cilíndrica de Miller.

A todas las fotografías panorámicas montadas para el proyecto se les aplicó la proyección gnomónica, que mantiene rectos los cuatro lados de la imagen de una fachada. Esto es muy práctico para ajustar la imagen en el modelo y visualmente da muy buen resultado.

Retoque fotográfico con *Photoshop*

En la primera fase algunas fotografías fueron tratadas mediante el programa de retoque fotográfico *Photoshop* para poder ser aprovechadas como fachadas. En el ejemplo de las figuras 4.57 y 4.58 puede observarse como, al no poder obtener a fachada completa debido a la presencia de un parking cubierto en la parte inferior, se tuvo que retocar la imagen para conseguir el efecto deseado.

Figura 4.57. Fotografía original de la fachada

Figura 4.58. Fotografía retocada

A continuación puede apreciarse el resultado de su aplicación sobre el modelo.

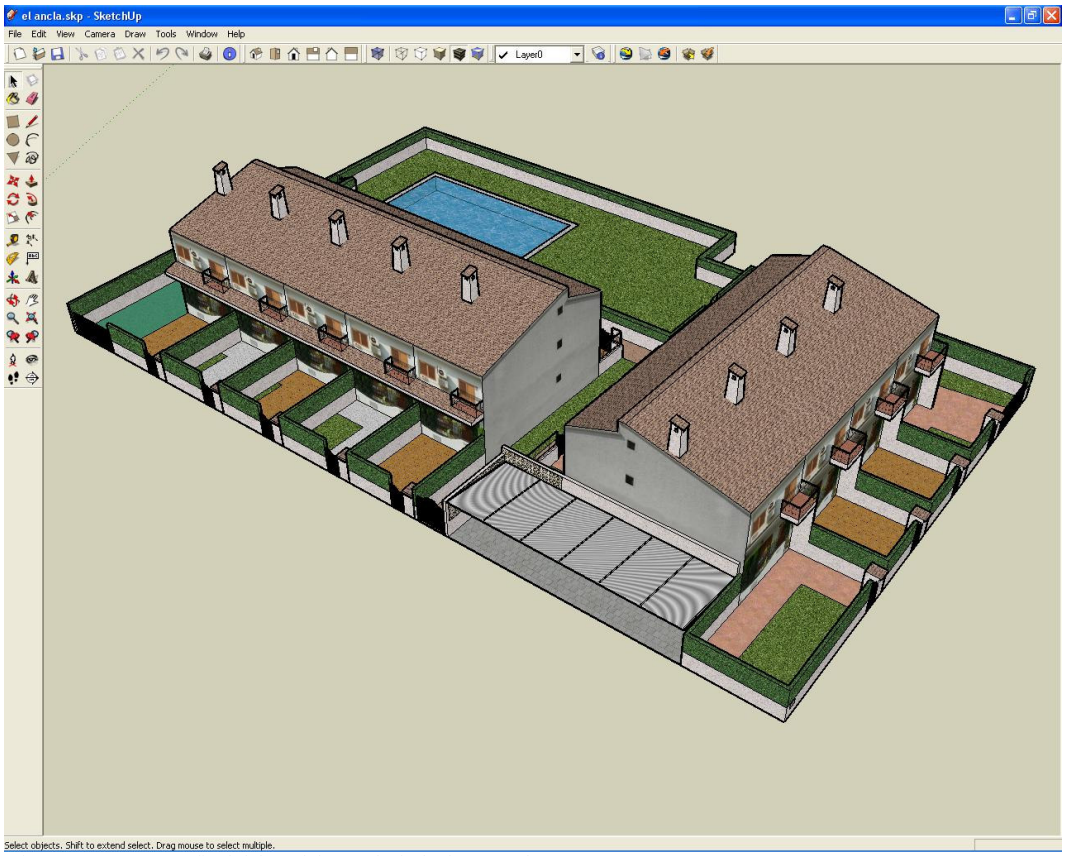


Figura 4.59. Aspecto final del modelo con la fachada retocada

Teselado

Por último, en esta fase también se utilizó una técnica para aprovechar aquellas fotografías de fachadas que estaban muy cubiertas de obstáculos o que hubo manera de fotografiarlas al completo y no fueron retocadas mediante *Photoshop*. Se trata del teselado, esto es, se cogió una parte de la fotografía de una fachada y se recortó hasta dejar, por lo general, un solo piso de la fachada. Después se importó al modelo y se hizo que se repitiera en cada uno de los pisos del edificio, formando una trama o tesela. El resultado ahorró bastante memoria y fue bastante bueno, como puede apreciarse en la siguiente imagen:

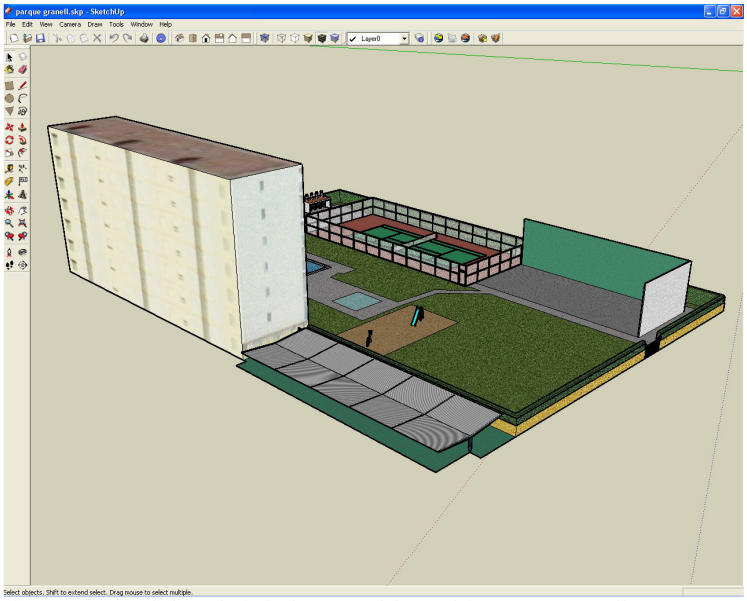


Figura 4.60. Teselado en la fachada trasera del complejo "Parque Granell"

El complejo Ramsés es el máximo exponente de esta técnica. En él se repiten en tesela balcones, fachadas, escaleras y la mayor parte de sus texturas. Como puede comprobarse, el resultado es muy satisfactorio.

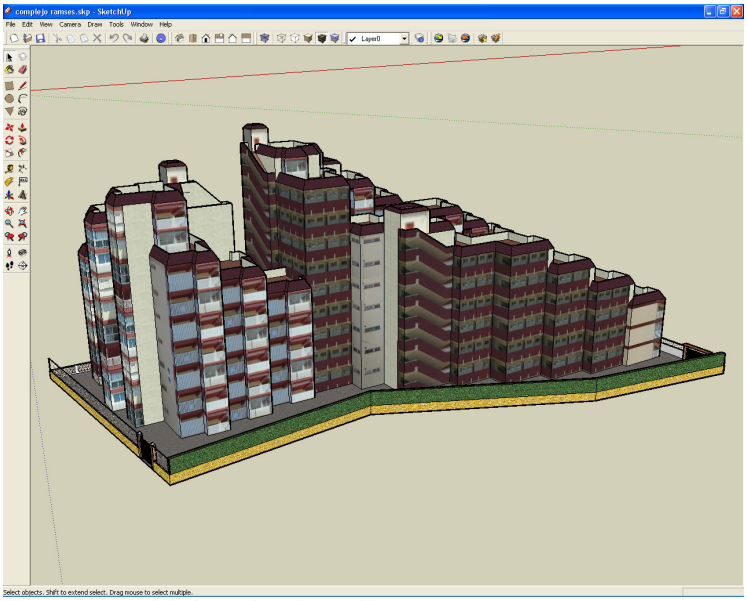


Figura 4.61. Teselado integral en el complejo "Ramsés"

Pese a que esta técnica es bastante buena, ahorra memoria y deja un buen resultado visual no se descubrió y se aplicó hasta casi el final de la fase de alta calidad, con lo que la mayoría de los modelos no hicieron uso de ella. La idea principal fue usar las fotografías originales de cada fachada y repetir lo menos posible unas fotografías en otras fachadas, dando así mayor realismo a los modelos. De esta manera se evitó el efecto de repetición de un patrón, que en ocasiones se produce al utilizar imágenes en tesela y que pueden dar un aspecto artificial al conjunto.

4.3.3.2. Fase de baja calidad

Teniendo en cuenta que el conjunto de los modelos se movía lentamente cuando era representado bajo *Google* Earth debido al excesivo consumo de memoria, se tomó la determinación de reducir el tamaño de todos ellos.

El primer modelo con el que se trabajó fue uno que tuviera diversidad de entidades y sobre el que se pudiera aplicar todo tipo de restricciones (número de polígonos, calidad y número de componentes, texturas, imágenes, etc.). El modelo elegido fue el del complejo "Orly", que presenta nueve edificios, pistas deportivas, piscinas y numerosas zonas valladas.

Una vez se remodeló el complejo y se aplicaron las nuevas texturas el tamaño del mismo era de 195KB. Es por eso que se tomó como referencia este tamaño y se intentó que todos los complejos ocuparan menos de 200KB, y así se hizo en la práctica mayoría de los modelos.

Rectificación de imágenes

La principal característica de las fotografías empleadas en esta fase de baja calidad es que todas están rectificadas, esto es, están preparadas para ser aplicadas sobre el modelo sin tener que ser deformadas, aunque haya que escalarlas en alguna dimensión para que se adapten al modelo.

Figura 4.62. Textura en alta calidad

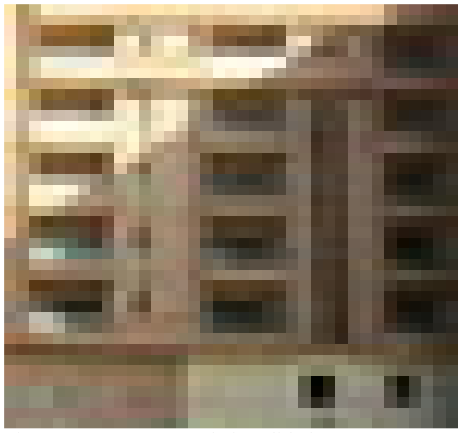


Figura 4.63. Textura en baja calidad

Como puede observarse en la figura 4.63 la fotografía se aprovecha en su totalidad, no conteniendo *pixels* inútiles.

Después se redimensionó hasta ocupar muy poca memoria, pero manteniendo reconocibles los elementos del edificio, como puede apreciarse a continuación:

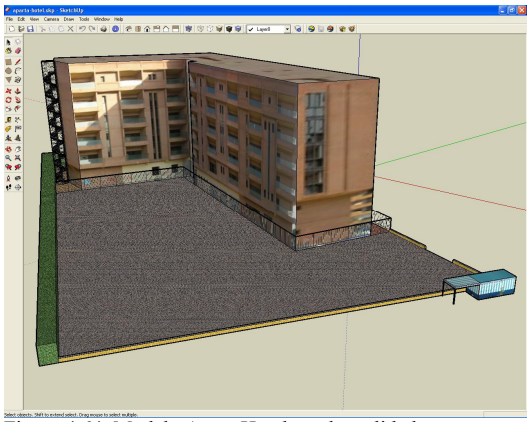


Figura 4.64. Modelo Apart-Hotel en alta calidad

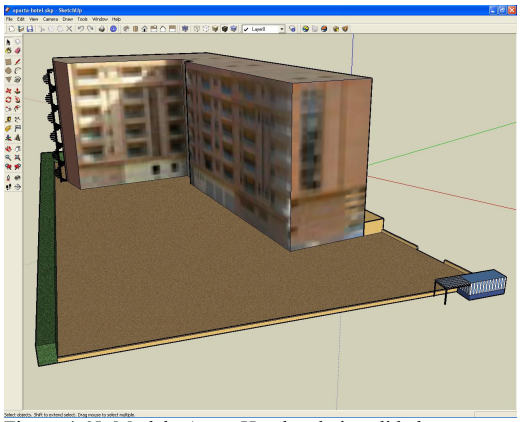


Figura 4.65. Modelo Apart-Hotel en baja calidad

Para rectificar las fotografías se emplearon dos métodos. El primero fue con *SketchUp*. En la fase de alta calidad, al aplicar las fotografías sobre los modelos, ya aparecían rectificadas. Para trabajar con ellas tan solo hubo que capturarlas, guardarlas como nuevas fotografías, reducir su tamaño y volver a importarlas en los modelos de baja calidad. Así, la fotografía de la figura 4.66 se importó y rectificó para que encajara en la fachada del modelo de alta calidad (Fig. 4.67).


Figura 4.66. Fotografía original

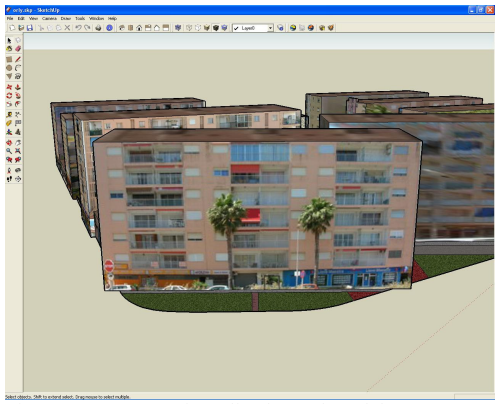


Figura 4.67. Fotografía aplicada en el modelo

Para la fase de baja calidad se copió esta fachada, se pegó en un nuevo archivo de *SketchUp* y se reorientó para obtener una vista frontal de la misma:

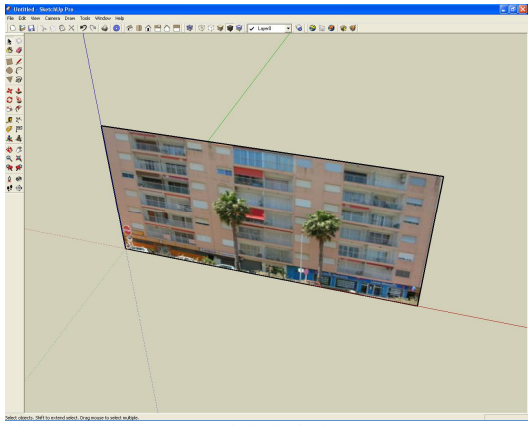


Figura 4.68. Importación de la fachada

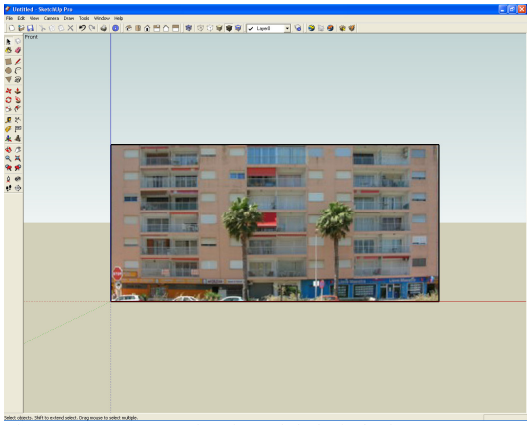


Figura 4.69. Presentación frontal de la fachada para su captura

Esto permitió capturar la imagen rectificada, que una vez recortada y redimensionada quedó así:

Figura 4.70. Imagen final usada como textura

Por otra parte, también hubo fachadas que no tuvieron tan buen resultado en la fase de alta calidad. Se trataba de las que habían sido tomadas bajo un ángulo cerrado.

Figura 4.71. Fotografía tomada bajo un ángulo muy cerrado

La imagen se recortó y redimensionó. Después se importó a *SketchUp* y se rectificó con las herramientas de edición de textura quedando como sigue:

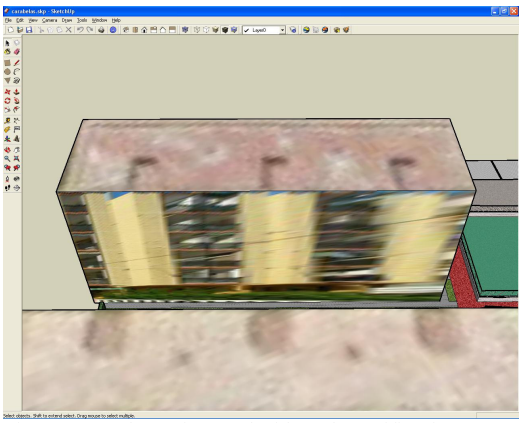


Figura 4.72. Aberración producida en la rectificación

En la figura 4.72 puede observarse que se produce una aberración de la imagen en la parte derecha. Esto es debido a que, para la zona izquierda de la fachada hay mayor número de *pixels* que para la zona derecha, que se encuentra más alejada y aparece más pequeña en la fotografía original. Este problema aparece al rectificar la foto en *SketchUp* y se acentúa al haberla redimensionado y aparecer ésta más pixelada.

La solución que se encontró para este tipo de fotografías de fachadas fue crear un nuevo archivo de *SketchUp* y dibujar un rectángulo con las proporciones de la fachada (Fig. 4.73). A continuación se le aplicó la fotografía original con toda su resolución y se procedió a ajustarla lo mejor posible hasta que quedó correctamente rectificada (Fig. 4.74).

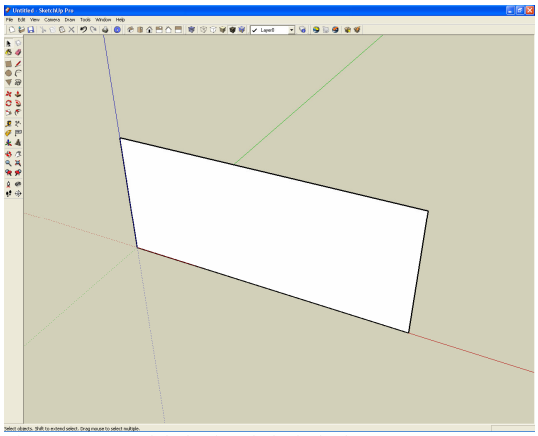


Figura 4.73. Modelo básico de la fachada

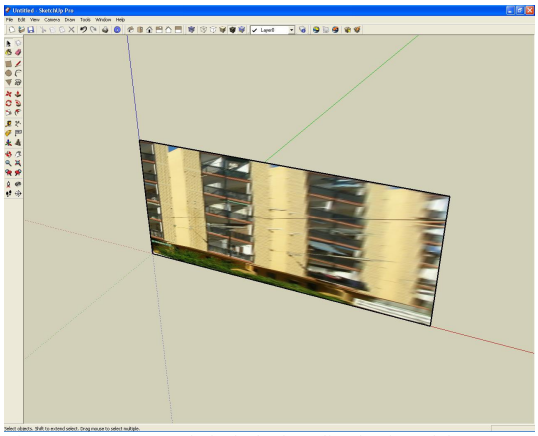


Figura 4.74. Imagen de la fachada aplicada al modelo

Después se repitió el mismo proceso explicado anteriormente para las fachadas aprovechadas de la fase de alta calidad; se muestra frontalmente, se captura, se almacena como una nueva imagen y se recorta y redimensiona.

Así pues, puede observarse en la figura 4.75 que la aberración de la parte derecha todavía era apreciable pero ya no tan pronunciada como antes. A continuación, cuando se llevó a cabo el proceso arriba indicado, se notaba todavía menos (Fig. 4.76).

Figura 4.75. Fotografía en alta calidad rectificada

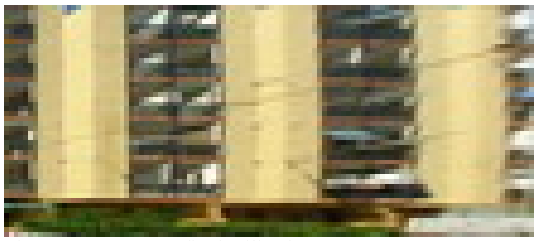


Figura 4.76. Imagen final utilizada como textura

Una vez hecho esto, ya se pudo importar la imagen a *SketchUp* y aplicar sobre los modelos de baja calidad, aprovechándose así más la imagen y ahorrándose bastante memoria.

Un segundo método para rectificar las fotografías fue utilizando la aplicación *Hugin*, un programa que permite el montaje de imágenes panóramicas y que ya se trató en la fase de alta calidad.

Las fotografías rectificadas mediante *Hugin* han sido aquellas que, siendo montajes panorámicos, presentaban los problemas de las fotografías de fachada tomadas bajo un ángulo bastante cerrado. Cabe destacar que el programa no permite coger una única fotografía y rectificarla, siendo necesario un mínimo de 2 fotografías. Dado que se comprobó que los resultados de la rectificación mediante *Hugin* eran mucho mejores que los obtenidos mediante *SketchUp*, se optó por aquel para las fotografías panorámicas.

Figura 4.77. Fotografías utilizadas para generar la fotografía panorámica

Así pues, a partir de las dos imágenes anteriores, se obtuvo mediante *Hugin* el montaje panorámico de la figura 4.78. A continuación, para poder dejar la fotografía rectificada, se hicieron modificaciones del punto de vista y se aplicó una proyección rectilínea, que dejó la fotografía panorámica como muestra la figura 4.79.

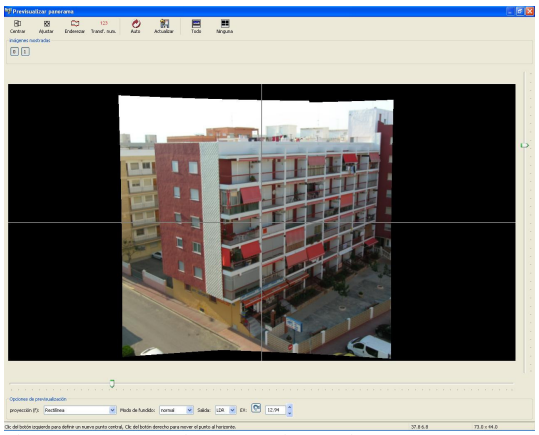


Figura 4.78. Fotografía panorámica resultante

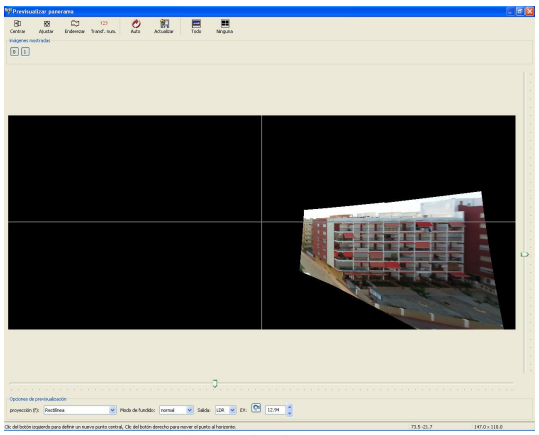


Figura 4.79. Fotografía rectificada en *Hugin*

La imagen rectificada resultante puede observarse más al detalle en la figura 4.80. Una vez rectificada y, como se ha hecho en todo el proceso de tratamiento de imágenes, ésta fue capturada, recortada y redimensionada (Fig. 4.81) hasta tener un tamaño aceptable para luego volver a importarla a *SketchUp* y usarla como una nueva textura.

Figura 4.80. Imagen resultante

Figura 4.81. Imagen usada como tesela

Cabe destacar que en este ejemplo del modelo del complejo Estudio I, además de usar la fotografía rectificada de una de sus fachadas para otras del mismo modelo, que son muy parecidas, se creo la tesela de la figura 4.81. De esta manera fue posible texturizar las dos fachadas del modelo con una única fotografía de mucho menor tamaño. Esto ahorró más memoria y el resultado fue excelente, como se puede apreciar en la figura 4.82.

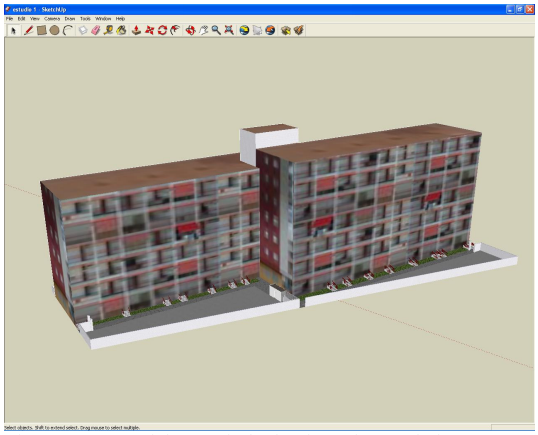


Figura 4.82. Modelo con fachadas frontales teseladas

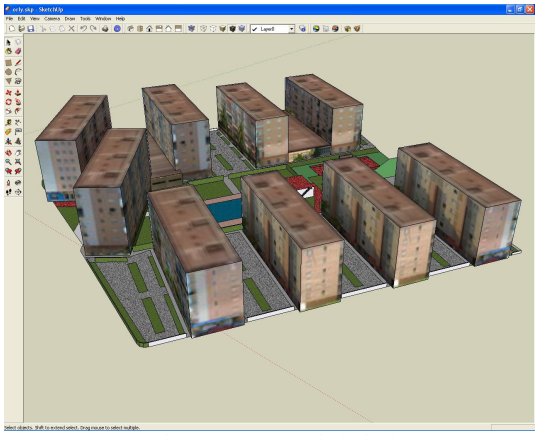


Figura 4.83. Modelo con repetición de fachadas

Por otra parte se debe mencionar que, así como en la primera fase había algunos modelos en los que se había aprovechado la fotografía de una fachada para utilizarla en otras, en esta segunda fase, en la práctica totalidad de los modelos se ha utilizado esta técnica ahorrándose así la memoria utilizada en ellos.

Así pues, puede observarse en la figura 4.83 el modelo del Complejo Orly, donde las fachadas laterales de los edificios que se encuentran en primer plano aparecen repetidas. De igual manera ocurre con las dos centrales, que son iguales entre sí, siéndolo también las que es encuentran en los extremos. A su vez las dos fachadas laterales del fondo a la derecha también son iguales.

Para el ejemplo del Complejo Orly, repetir fotografías en diferentes fachadas hizo que se pasara de usar 41 fotografías con un tamaño medio de unas 30KB por fotografía a usar 21 fotografías con un tamaño medio de 1.7KB por fotografía.

Otro aspecto en el que se ahorró memoria fue en las fotografías de las azoteas de los edificios. En la fase de alta calidad se capturaron imágenes de azoteas desde *Google Earth*, tanto del complejo entero o de edificios individuales, obteniéndose lo siguiente:

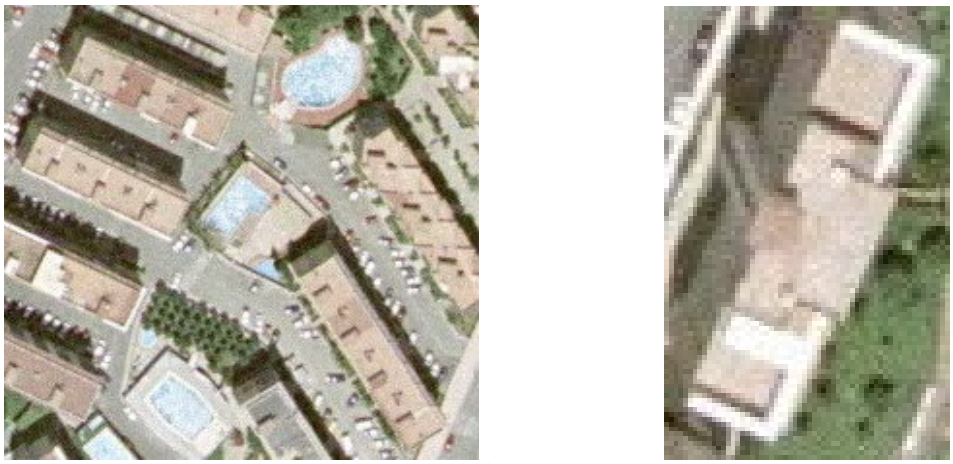


Figura 4.84. Azoteas capturadas en conjunto e individualmente desde *Google Earth*

En la fase de baja calidad, sin embargo, se reorientaron las fotografías para su máximo aprovechamiento, empleándose la misma para todos los edificios.

Figura 4.85. Azotea reorientada

En el ejemplo del Complejo Orly (Fig. 4.86) supuso un ahorro importante, dado que tiene 11 edificios.

Figura 4.86. Modelo del Complejo Orly, con todas las azoteas texturizadas a partir de la misma fotografía

Retoques fotográficos con *Photoshop*

En esta segunda fase, igual que ocurrió en la primera, aunque de manera más profusa, se hizo uso del retoque fotográfico para algunas de las fotografías de fachadas. Se realizaron retoques a cerca de una veintena de fotografías, principalmente en aquellas que iban a repetirse en diferentes partes de un mismo modelo. De esta manera apenas se notó la repetición y el resultado final fue estéticamente más limpio. Si se hubiera dejado la fotografía original, con un árbol en mitad de la misma o vegetación en su parte inferior u otros elementos, al aparecer repetida en otras fachadas podría haber dado la sensación de artificialidad, además de hacer patente el efecto aplicado. Sin embargo con estos ligeros retoques el resultado fue bastante bueno.

En el ejemplo del complejo "Madeira" se contaba con la fotografía original, que una vez rectificada y recortada quedaba así:

Figura 4.87. Fotografía original rectificada y recortada

A partir de la fotografía anterior y mediante retoque fotográfico se logró quitar la vegetación de la parte inferior y se obtuvieron las siguientes fachadas, a partir de las cuales se texturizó todo el modelo.

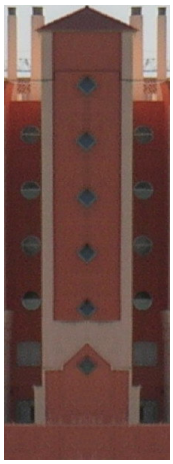


Figura 4.88. Fotografías retocadas y utilizadas finalmente como texturas

De esta manera el acabado final del modelo tras el retoque es muy satisfactorio:

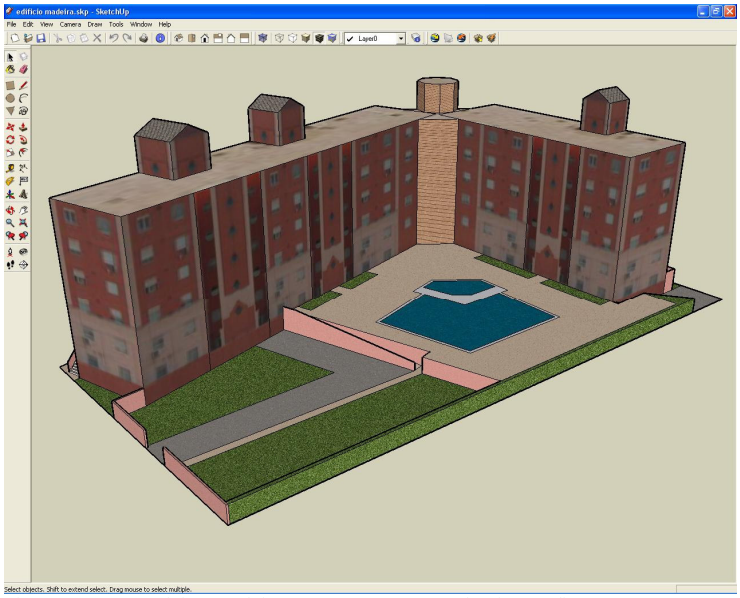


Figura 4.89. Complejo "Madeira" con texturas retocadas fotográficamente

Por otro lado, en el modelo de la feria situada en la Plaza de Las Cortes Valencianas también se aplicaron retoques fotográficos en las diferentes atracciones que forman el recinto (Fig. 4.90), repitiéndose luego en los cuatro lados de cada una.

Figura 4.90. Retoque fotográfico en las fachadas de las atracciones de la feria

Siendo el resultado final el siguiente:

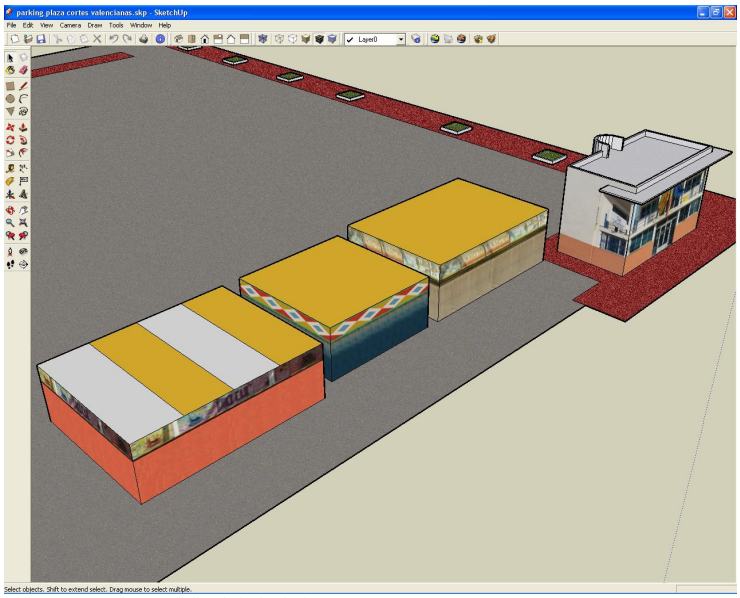


Figura 4.91. Aspecto final de la feria

Otro ejemplo de retoque fotográfico fue el realizado a las fotografías de las fachadas del complejo "Estudio I". En éstas aparecían elementos que ocultaban parte de las fachadas. Algunos de ellos estaban muy separados de éstas en la realidad, produciendo una extraña perspectiva en el acabado final. En las siguientes imágenes puede apreciarse cómo se pudo quitar la valla y el coche que aparecían en primer plano y reconstruir la planta baja a partir del diseño de otras plantas.

Figura 4.92. Retoque fotográfico de la fachada trasera

En la figura 4.93 puede verse cómo se borró una señal, situada delante de la fachada, que habría aparecido repetida en otra de este mismo modelo.

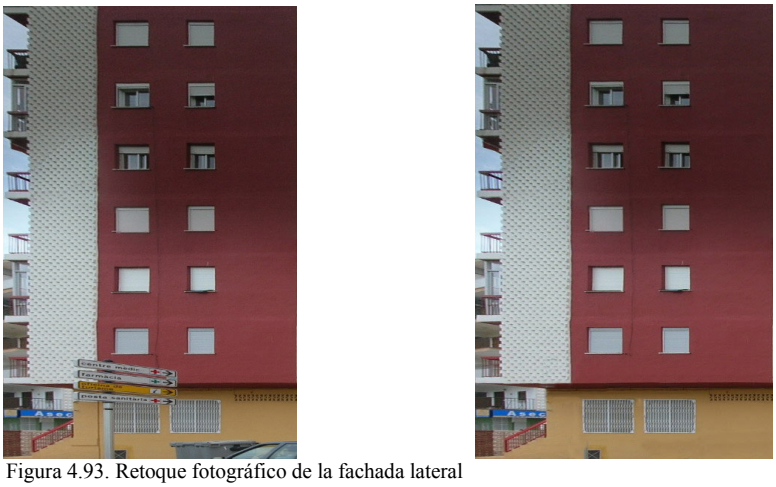


Figura 4.93. Retoque fotográfico de la fachada lateral

Por último, en la fachada trasera (Fig. 4.94) se pudo borrar un muro y una farola que había delante de la urbanización y se reconstruyó la parte inferior de la fachada a partir del patrón de otras plantas. De esta manera se pudo eliminar el muro que, debido a la rectificación de la imagen, aparecía deformado en su parte inferior derecha.

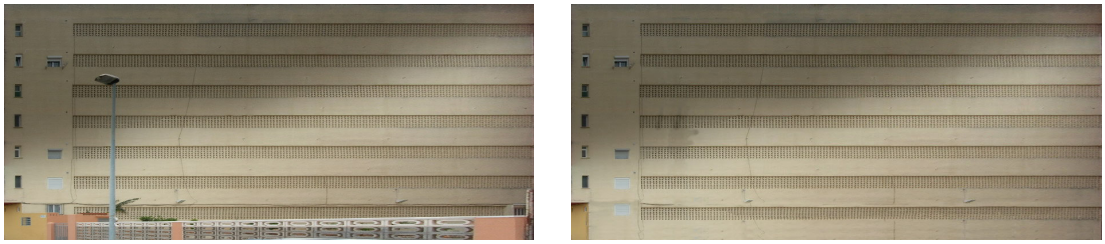


Figura 4.94. Retoque fotográfico en la fachada trasera del complejo "Estudio I"

Después del retoque fotográfico las fotografías fueron reducidas de tamaño y de nuevo aplicadas al modelo, siendo su aspecto final el siguiente:

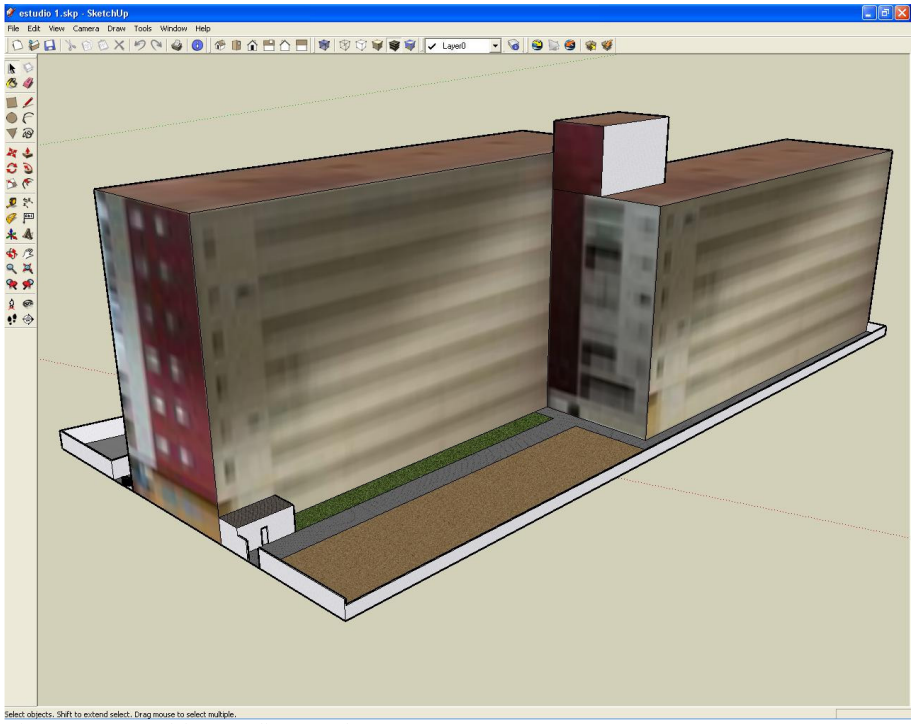


Figura 4.95. Complejo "Estudio I" con las nuevas texturas

Finalmente, como último ejemplo de retoque fotográfico con *Photoshop* cabe destacar la fachada lateral del complejo Port-Mar, que aparecía repetida en todo el modelo. Revisando su versión en alta calidad (Fig. 4.96) puede comprobarse que había una sombra de palmera en dicha fachada.

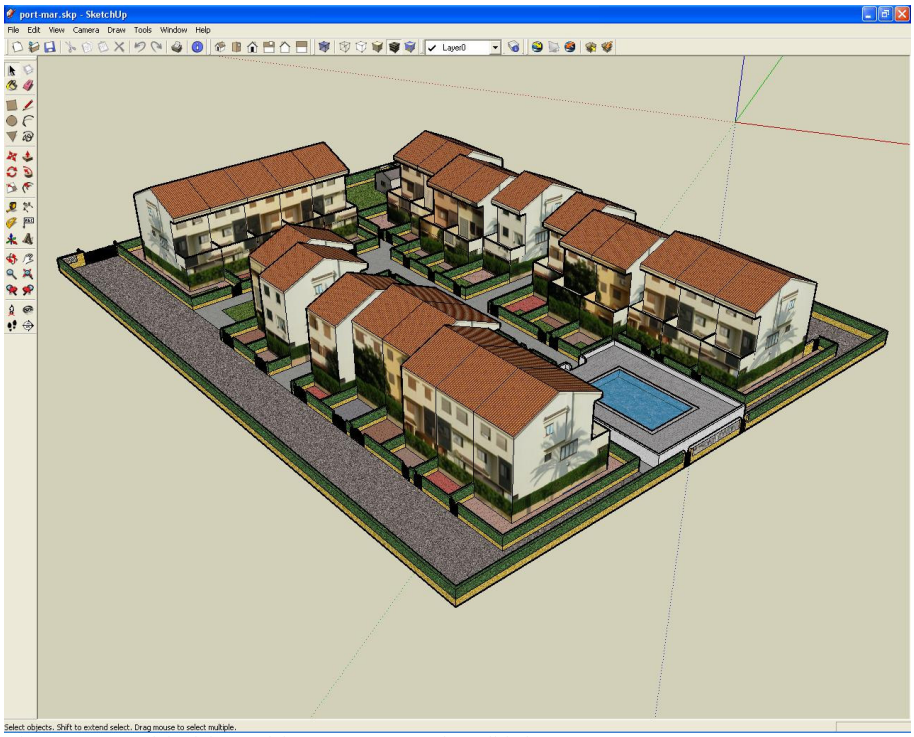


Figura 4.96. Modelo del complejo "Port-Mar" en alta calidad

Así pues la fotografía original (Fig. 4.97) fue rectificada, recortada y retocada, obteniéndose la imagen de la figura 4.98.

Figura 4.97. Fotografía original de la fachada

Figura 4.98. Imagen retocada fotográficamente

Ésta, a su vez fue redimensionada (Fig. 4.99) para texturizar el modelo.

Figura 4.99. Textura final

Así pues, su aspecto final en baja calidad es el siguiente:

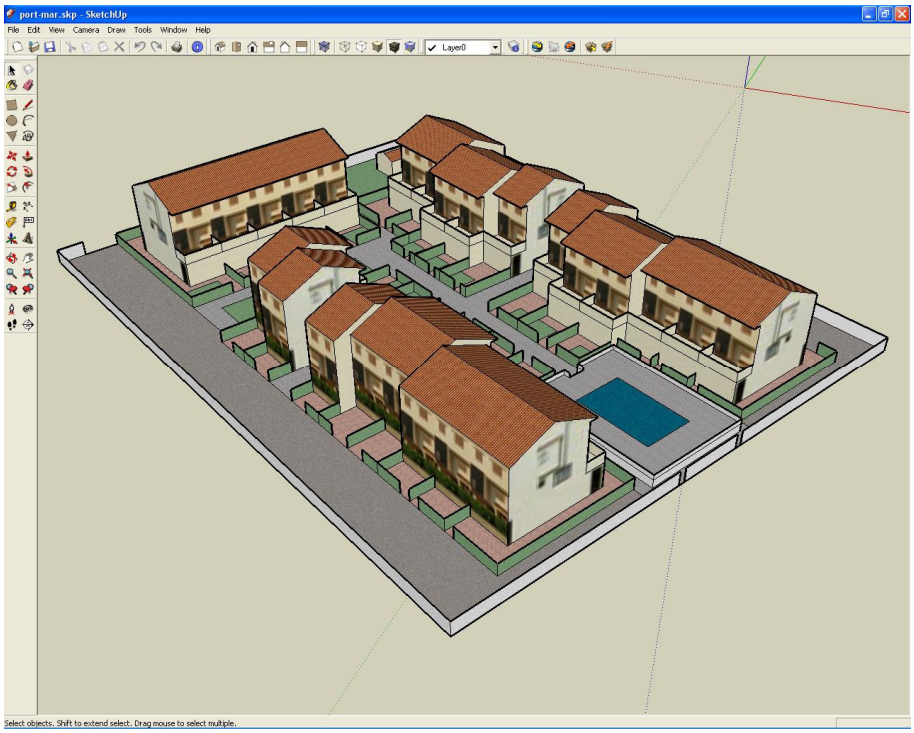


Figura 4.100. Modelo del complejo Port-Mar con las nuevas texturas en baja calidad

Teselado

En la fase de alta calidad se empleó esta técnica en sólo 6 modelos y de manera poco profusa. Esto se debió a que fue descubierta hacia el final de la misma. Sin embargo, en esta segunda fase el número de modelos en que se empleó el teselado fue algo superior; 15 en total. En la primera fase ya se explicó en qué consistía la técnica y en ésta tan solo puede decirse que se consideraron sus bondades. Se empleó principalmente en aquellos complejos que, pese a la reducción de la calidad de las fotografías y pese a que estas estuvieran rectificadas y el modelo en sí hubiera sido también simplificado, seguían ocupando mucha memoria. También debió darse la circunstancia de que la fachada lo permitiera y que las fotografías con las que se contaba lo permitieran de una manera óptima, sin que se llegara a notar el efecto a simple vista. Es por eso que no se aplicó en la totalidad de los modelos y sólo en una tercera parte de los mismos. Así, en el modelo de Urbanización El Ancla, se partió de la siguiente fotografía rectificada:

Figura 4.101. Imagen usada como tesela

Ésta pudo texturizarse sobre el modelo, representando la fachada principal de cada uno de los adosados y utilizando para ello únicamente un polígono que las abarcaba a todas. El resultado fue muy satisfactorio, a la par que económico.

Figura 4.102. Modelo del complejo “El Ancla” después del teselado

Para el modelo del complejo Parque Granell se prepararon varias fotografías que fueron aplicadas en tesela. Para la fachada principal se escogió una tesela formada por tres pisos (Fig. 4.103), para que no se notara mucho el efecto del teselado.

Figura 4.103. Tesela para la fachada frontal

En los laterales, así como la parte trasera se preparó una tesela de un solo piso:

Figura 4.104. Teselas usadas para las fachadas laterales y la trasera

Cabe destacar que, al observar el edificio en la realidad se aprecia una repetición en las fachadas laterales y trasera, muy parecida a la mostrada en el modelo digital.

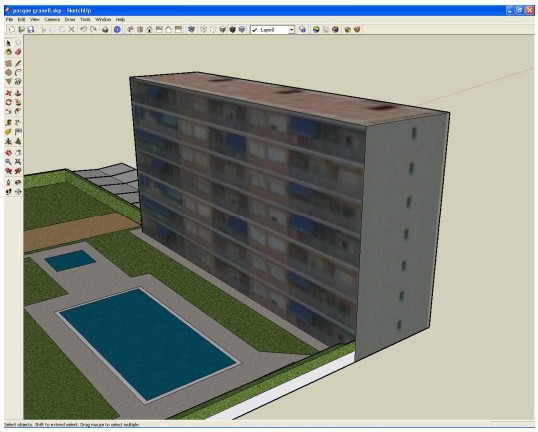


Figura 4.105. Teselado en la fachada frontal

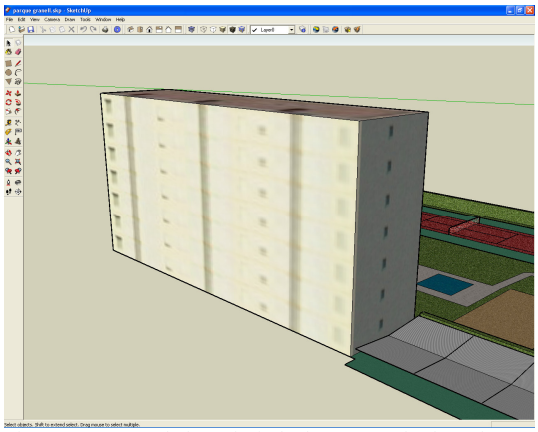


Figura 4.106. Teselado en la fachada trasera y las laterales

Otro ejemplo de teselado tiene lugar en el Complejo Lord. En éste todas sus fachadas se texturizaron mediante teselas. Las siguientes son fotografías de pisos individuales empleadas como teselas:

Figura 4.107. Teselas empleadas en el texturizado del Complejo Lord

Obteniéndose, como puede observarse a continuación, muy buenos resultados:

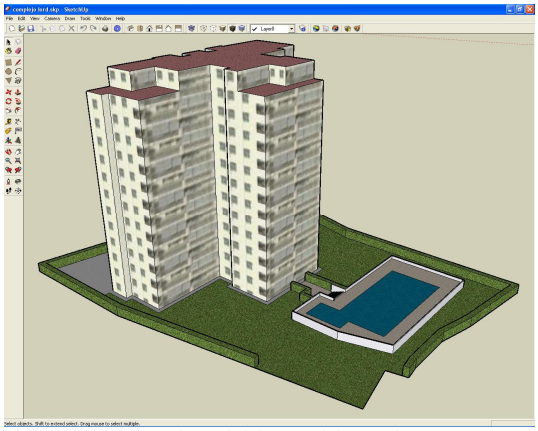


Figura 4.108. Vista frontal del Complejo Lord

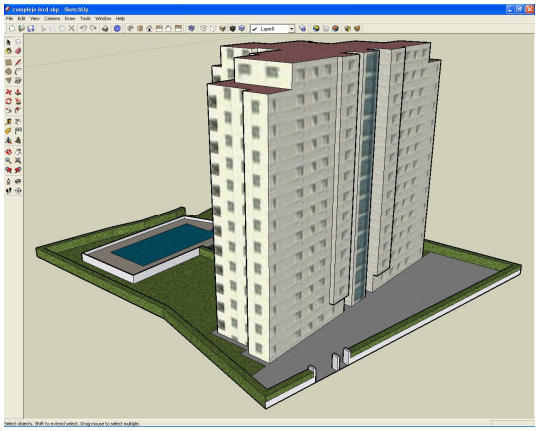


Figura 4.109. Vista posterior del Complejo Lord

Por último hay que resaltar el modelo que hace mayor uso del teselado y que ya se comentó en la fase de alta calidad; el Complejo Ramsés. La repetición de las teselas en este complejo ha sido mayoritariamente en vertical, aunque también horizontalmente en alguna ocasión. El resultado es espectacular.

Figura 4.110. Vista posterior del complejo Ramsés

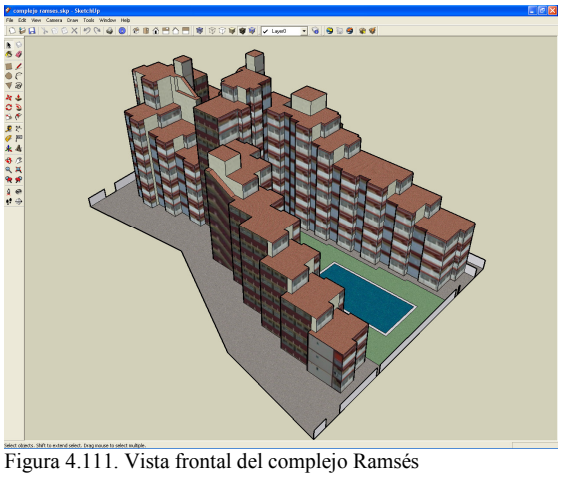


Figura 4.111. Vista frontal del complejo Ramsés

En la fase de baja calidad las fotografías de este modelo se rectificaron, se redujo la calidad de las mismas, se generaron nuevas teselas y se volvió a texturizar. Y pese a todo esto el resultado visual sigue siendo satisfactorio.

Tratamiento de texturas de *SketchUp*

Por último hay que comentar que *SketchUp* viene con un número determinado de texturas de materiales para trabajar en él. Éstas pueden ser ampliadas mediante la descarga de nuevos materiales desde su página Web. Estas texturas no son otra cosa que fotografías ya preparadas como teselas para ser aplicadas a las diferentes superficies de trabajo. Lo bueno de ellas es que la mayoría tiene buena calidad de imagen. Esto significa que su acabado final es muy bueno pero, como se ha visto a lo largo del capítulo, a costa de un gasto considerable de memoria. Es obvio que para la fase de baja calidad, cuya filosofía es la economía de la memoria, semejante gasto es impensable. En la gran mayoría de los casos se optó por escoger otros materiales del mismo *SketchUp* que consumieran menos memoria. Esto se hizo principalmente con las texturas de mayor uso en los modelos, como eran las texturas de vegetación, agua para las piscinas y pintura blanca para los muros.

En la primera fase no se entró en este tipo de consideraciones, pero una vez se comprobó la cantidad de memoria que consumía, por ejemplo, la vegetación tipo *Kudzu* en lugar del tradicional *césped*, se optó por este segundo para representar tanto el césped como cualquier otro tipo de vegetación. Así se pasó de usar dos texturas a una, que además consumía menos memoria. Un poco más adelante se sustituyó también la textura *césped* por la *Putting Green*, que es todavía más económica y da un buen resultado visual a simple vista. Lo mismo ocurrió con la textura usada al principio para el agua de las piscinas. Ésta era *Water Sparkling* y pasó a usarse *Water deep*, que consumía bastante menos memoria (Tabla 4.1.).

Textura original	Tamaño (KB)	Textura final	Tamaño (KB)
Juniper	20	Putting green	6
Grass blue	28		
Kudzu	20		
Ivy Hedera	16		
Brick colored gold	17	Concrete warm	6
Brick antique	15	Brick red mix	5
Brick english simple	6		
Brick red	10		
Asphalt rubber red	18	Brick pavers octagon	13
Metal steel textured	7	Color I20	2
Fencing chain link	9	Fencing construction	4
Fencing diamond mesh	10		
Water sparkling	14	Water deep	11
Water pool	15		
Asphalt old	16	Concrete gray	7
Asphalt old 2	17		
Cladding plaster rough	6	Cladding plaster chalk	2
Cladding stucco white	16		
Concrete aggregate small	22		
Concrete aggregate stained	17	Concrete scored red	9
Concrete scored gray	6	Concrete block smooth	3
Concrete block 8x8 gray	7		
Roofing tile spanish	23	Roofing shingles asphalt 1	2
Roofing metal seam red	3		
Roofing shingles variable	7		
Concrete squares	11	Tile squares grays	9
Tile ceramic earthy	9	Tile ceramic 2inch	4
Wood bamboo dark	14	Wood cherry original	11
Groundcover dirt red	13	Groundcover sand smooth	10
Cladding stucco brown	13		
Carpet aqua	16	Roofing metal seam green	13

Tabla 4.1. Relación de texturas originales y sus substitutas de menor tamaño

Esto es lo que se hizo en un principio para reducir la memoria usada en los modelos, pero llegó un momento que ni con estos recortes se lograba llegar a los requisitos que se habían impuesto para los modelos (Apdo. 4.3.3.2.).

SketchUp utiliza unos archivos de materiales propios, que no son más que imágenes en un formato propietario (.skm: "*SketchUp material*"), y que pueden ser exportadas como imágenes JPEG. De esta manera se comprobó que muchos de ellos ocupaban mucha memoria. Y así fue como en 7 modelos de complejos, aquellos en los que se quería ajustar más la memoria utilizada, se optó por exportar como imágenes JPEG aquellos materiales que más memoria ocupaban. Una vez exportadas se redimensionaron para que consumieran una cantidad de memoria aceptable y fueron importadas en *SketchUp* como texturas nuevas. El resultado es bastante bueno, ya que la textura es prácticamente la misma. Aparecen ligeramente difuminadas debido al efecto *antialiasing* que *SketchUp* aplica a las imágenes, pero esta manera se ha ahorrado mucha memoria.

A continuación se muestran varios ejemplos de reducción de la calidad en materiales propios de *SketchUp*:

Figura 4.112. Textura original de *SketchUp* (Teja española, 22.8KB) y textura redimensionada (1.08KB)

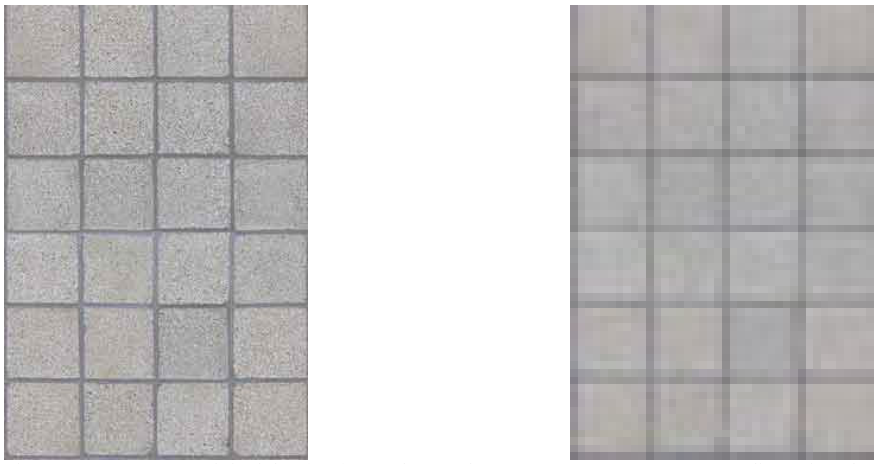


Figura 4.113. Textura original de *SketchUp* (Bloques de cemento, 10.5KB) y textura redimensionada (1.32KB)

Por último, como se comenta al principio de este apartado así como en anteriores secciones, una vez las imágenes están retocadas y se ha ajustado la cantidad de memoria que ocupan, se procede a su aplicación sobre las superficies de los edificios de los modelos 3D, dándoles su aspecto final realista y terminándose así el proceso de modelado 3D. Estos modelos así generados ya están preparados entonces para ser mostrados en *Google Earth*, donde se presentarán en conjunto para comprobar su aspecto y buen funcionamiento.

4.4. Búsqueda y tratamiento de la información

4.4.1. Introducción

En este apartado se explica cuál ha sido el proceso de búsqueda y el tratamiento de la información de los comercios y servicios que pueden encontrarse en la zona de estudio, y se presentan las aplicaciones bajo las que se ha mostrado dicha información.

En primer lugar, se presenta *Google Maps*, una aplicación que muestra mapas en 2D y permite superponer información generada por los usuarios. Cabe destacar que *Google Maps* no se ha utilizado en el desarrollo del proyecto, pero que, dado que se considera a *Google Earth* como una evolución 3D de esta aplicación, resulta interesante presentarla, explicando sus características. A continuación se presenta *Google Earth*, que muestra un modelo 3D de la Tierra y permite localizar geográficamente modelos 3D, así como superponer información alfanumérica y gráfica. Se describen sus características, el funcionamiento de su interfaz, los sistemas de navegación y demás funciones. Después se muestran las herramientas de que dispone para añadir diferentes tipos de elementos geométricos, gráficos, así como información alfanumérica. Por último, se explica cómo se ha procedido para la obtención de la información sobre los comercios de la zona de estudio y cómo se han generado todas las marcas de posición y se han maquetado sus respectivas ventanas de información.

4.4.2. Google Maps

4.4.2.1. Introducción

En este apartado se presenta *Google Maps*, una aplicación Web que permite ver mapas 2D de todo el mundo y permite realizar diferentes tipos de consultas y búsquedas de comercios. A continuación se describe el funcionamiento de su API, explicando sus componentes básicos y los elementos principales que se pueden encontrar en sus mapas.

4.4.2.2. Historia

Antes de 2005 había servicios de mapas, como *MapQuest* [130] y otros, que permitían buscar lugares, direcciones y comercios, pero sólo mostraban aquellas empresas que patrocinaban el propio servicio, quedando muchas otras imposibles de localizar.

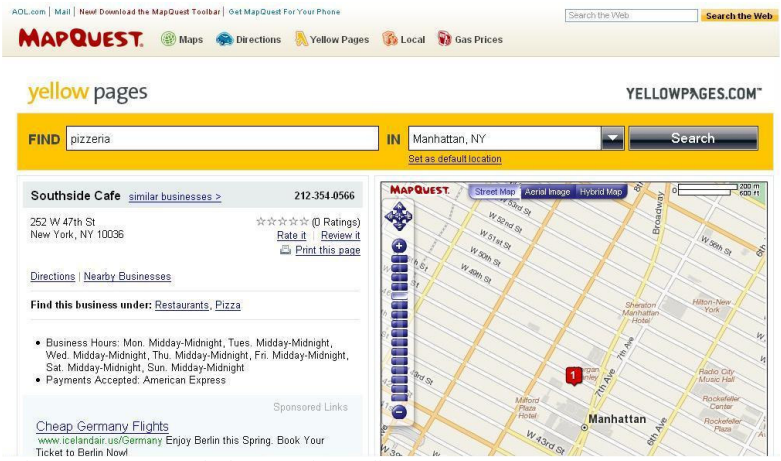


Figura 4.114. Búsqueda de comercios mediante *MapQuest*

A principios de de 2005 Google lanzó la beta de *Google Maps* [85]. Se trataba de un servicio cartográfico gratuito, a través de la página Web de su laboratorio (http://labs.google.com) [84]. El nuevo sistema de carga de mapas era más rápido que el de los antiguos servicios de mapas, permitiendo moverse de forma más dinámica por los mapas en lugar de tener que pulsar y recargar, gracias al uso de la tecnología AJAX.

La interfaz presentaba una serie de controles para navegar por el mapa, cosa que también podía hacerse con el ratón o los cursores. También era posible activar imágenes aéreas de algunas partes del mundo, incluso mostrar una vista híbrida que mezclaba las imágenes aéreas con las de los mapas.

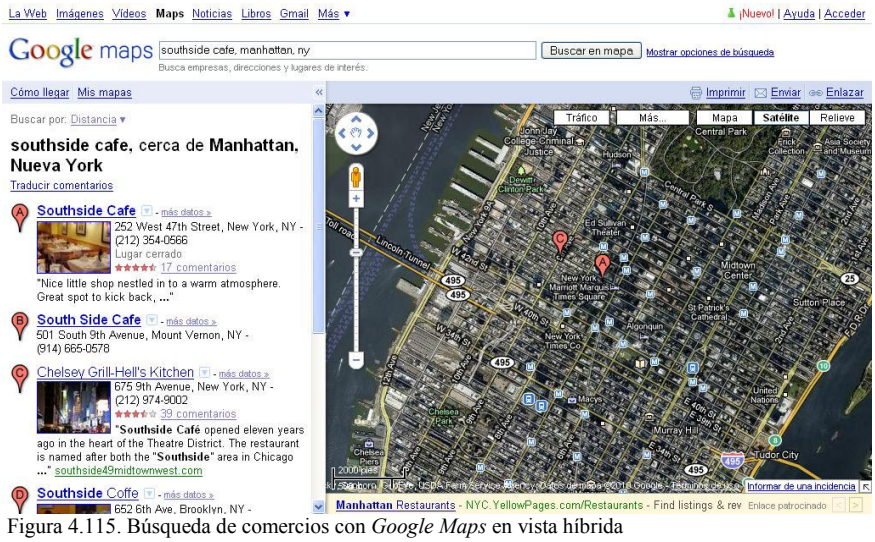


Figura 4.115. Búsqueda de comercios con *Google Maps* en vista híbrida

Desde un principio *Google Maps* fue un éxito. Fue el primer servicio de mapas gratuito que proporcionaba además fotografías aéreas de alta resolución para la mayoría de las áreas urbanas y principales ciudades del mundo. Estaba dotada de una interfaz sólida y de rápida respuesta que mostraba direcciones y mapas de ciudades y pueblos. Tenía un planificador de rutas para ir a pie, en bicicleta, coche o transporte público y un localizador de comercios para numerosos países de todo el mundo.

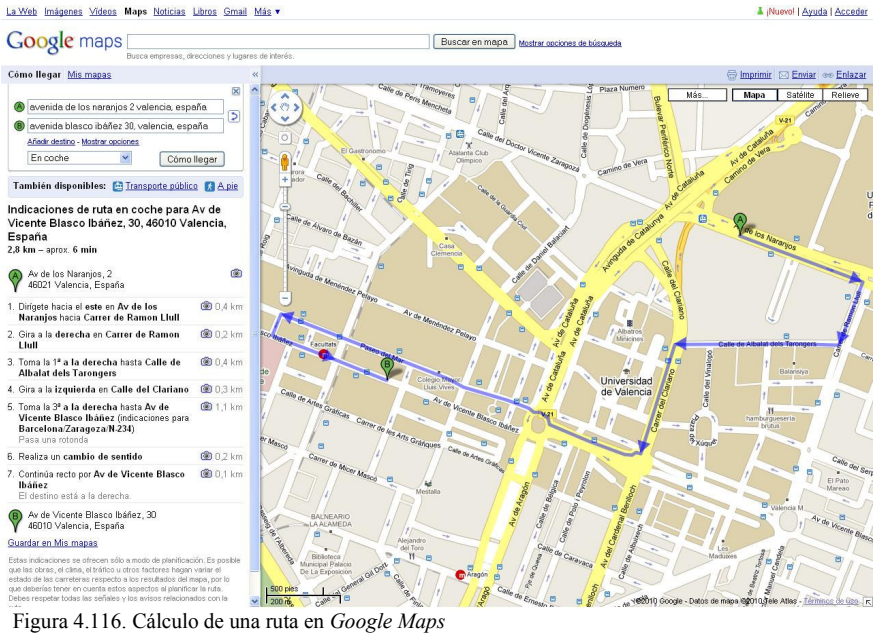


Figura 4.116. Cálculo de una ruta en *Google Maps*

Para poder localizar geográficamente todos los elementos con los que trabaja y representarlos en la pantalla Google creó su propia proyección, denominada “Esférica de Mercator” o “Proyección Google”.

Se trata de una proyección Mercator que considera la Tierra como una esfera. Aparece especificada en el estándar “EPSG: 900913” y emplea los siguientes parámetros:

```
900913=PROJCS["WGS84 / Simple Mercator",
              GEOGCS["WGS 84",
                     DATUM["WGS_1984",
                     SPHEROID["WGS_1984", 6378137.0, 298.257223563]],
                     PRIMEM["Greenwich", 0.0],
                     UNIT["degree", 0.017453292519943295],
                     AXIS["Longitude", EAST],
                     AXIS["Latitude", NORTH]],
              PROJECTION["Mercator_1SP_Google"],
              PARAMETER["latitude_of_origin", 0.0],
              PARAMETER["central_meridian", 0.0],
              PARAMETER["scale_factor", 1.0],
              PARAMETER["false_easting", 0.0],
              PARAMETER["false_northing", 0.0],
              UNIT["m", 1.0],
              AXIS["x", EAST],
              AXIS["y", NORTH],
              AUTHORITY["EPSG","900913"]]
```

No obstante, la principal característica que le hizo destacar del resto de aplicaciones, fue que podía manipularse su código para añadir nuevos puntos de interés. De hecho podía combinarse su base de datos cartográficos con una fuente de datos externa y generarse al instante mapas con puntos de información georreferenciados.

El mayor cambio se produjo en Junio de 2005, cuando se presentó oficialmente su API (interfaz de programación de aplicaciones). Ésta permitía crear miles de aplicaciones Web que permitían a los usuarios disponer de gran variedad de contenidos (noticias, imágenes, resultados de elecciones, etc.) localizados en un mapa [204].

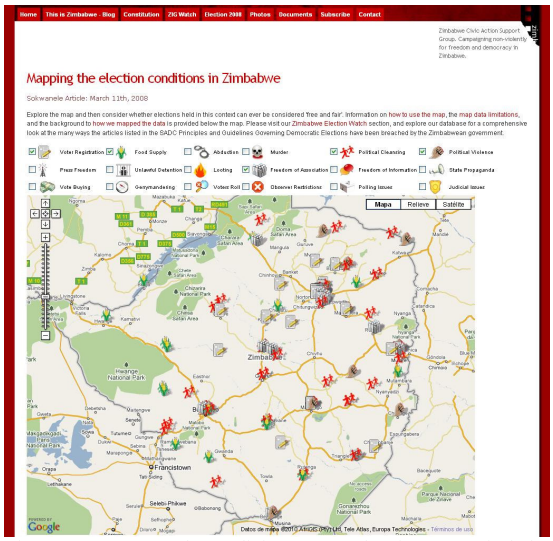


Figura 4.117. Mapa de condiciones para elecciones en Zimbabue

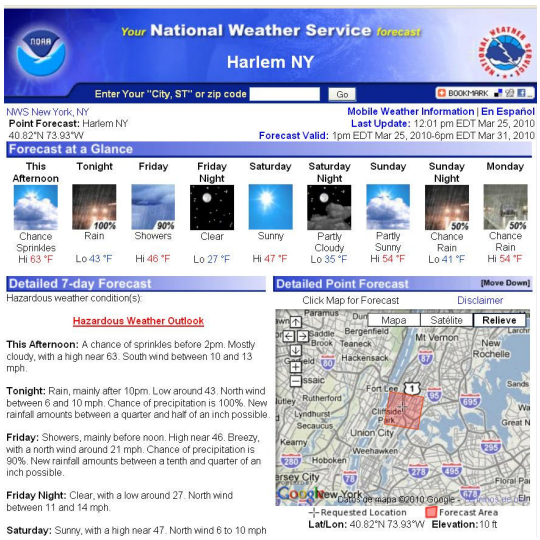


Figura 4.118. Página del Servicio de Meteorología de EEUU

4.4.2.3. La API de Google Maps

Una API es una interfaz de programación de aplicaciones, que en el caso de *Google Maps* permite colocar esta aplicación dentro de una página Web, permitiendo al usuario manipularla y realizar consultas sobre ella. Combinándola con una base de datos puede dar soporte a elementos interactivos vía Web sin tener que recargar la página o volver a mostrar partes del mapa.

Para poder utilizarla, es primero necesario registrarse en su página Web, y obtener una clave de acceso. Para conseguir una hay que tener una cuenta de correo electrónico abierta en Gmail e ir a la página Web de la API de Google (*http://code.google.com/intl/es/apis/maps/index.html*). Una vez allí se pulsa sobre el enlace "Registrarse para obtener una clave de API de Google Maps". Se teclea la dirección URL del sitio Web donde se aloja y se pulsa sobre "Generar clave de API". Ésta será enviada al correo electrónico. Cada vez que se crea una página que acceda a la API habrá que incluir dicha clave dentro de su código.

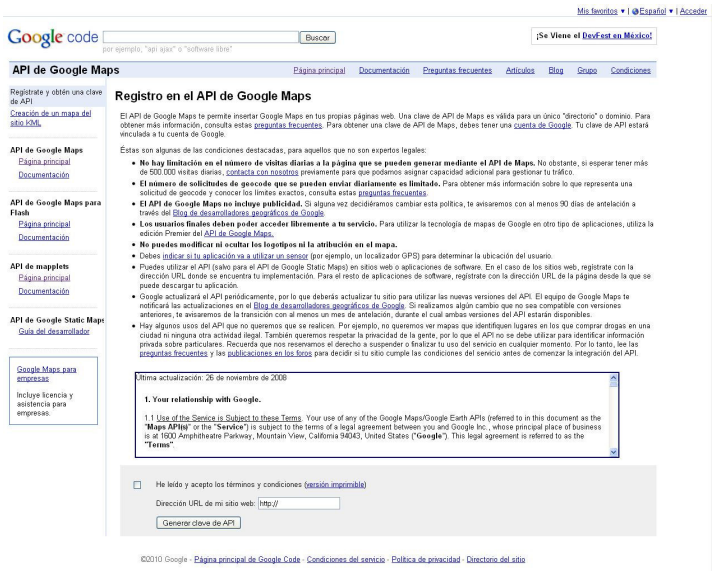


Figura 4.119. Registro de la dirección URL

Cabe destacar que la clave sólo funcionará cuando se verifique la página Web en la que la API va a estar alojada. Por tanto ésta deberá ser pública, no pudiéndose desarrollar aplicaciones a nivel local dentro de una red que no esté conectada directamente con Internet. Por otro lado, la clave sólo valdrá para un directorio específico, esto es, las aplicaciones sólo funcionarán cuando se carguen desde la URL exacta que se haya registrado. Para otros directorios por encima o debajo de ella no funcionarán. Así pues, para registrar un dominio entero que las incluya todas, éstas deberán estar almacenadas en el mismo directorio.

La API es compatible con los principales navegadores de Internet:

- *Internet Explorer* 6.0 o superior
- *Mozilla Firefox* 2.0 o superior
- *Safari* 3.0 o superior
- *Opera*
- Google *Chrome*

4.4.2.4. Componentes básicos

Su pieza clave es el componente *JavaScript*, que proporciona la interfaz para trabajar con *Google Maps*, genera el mapa en pantalla, carga las imágenes y las organiza por cuadrados. La aplicación Web se complementa con los siguientes componentes:

XHTML (HTML extensible)

Pese a funcionar en páginas HTML, se recomienda seguir el estándar XHTML. De esta manera se asegura la compatibilidad con la mayoría de navegadores.

VML (Lenguaje de Marcado de Vectores)

En el caso de que se pretenda trazar rutas o dibujar formas (polilíneas) sobre la aplicación, debe incluirse el espacio de nombre VML en las definiciones del XHTML.

Estilos y elementos

Google Maps no establece ninguna limitación o restricción sobre cómo debe ser el aspecto de la página Web que contenga la API. Por lo tanto, las hojas de estilo y maquetación de la página Web pueden diseñarse según las necesidades de la misma.

XML (Extensible Markup Language)

Utilizado para manejar enormes cantidades de datos (parcelas, subparcelas, servicios públicos, etc) el archivo XML se carga a través de la API, que muestra en el mapa los datos contenidos en él. La API también soporta transformación XSL, convirtiendo el documento XML en HTML mediante hojas de estilo.

4.4.2.5. Elementos principales de un mapa

Todas las aplicaciones de *Google Maps* comienzan con un sencillo mapa. A este mapa se le puede añadir una serie de elementos diseñados para proporcionar funcionalidad a la aplicación. Todos los mapas soportan cuatro elementos principales:

Overlays (superposiciones)

Muestran información de un lugar superpuesto al mapa. Existen dos tipos:

- Marcador. Consiste en un icono que indica la posición exacta de un comercio o lugar en el mapa.

- Polilínea. Se usa para trazar o dibujar una forma concreta en el mapa, p.ej., los límites de un término municipal o de una propiedad privada [177].

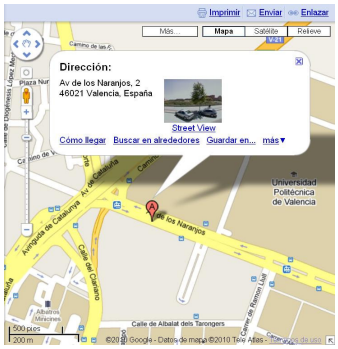


Figura 4.120. Ejemplo de marcador

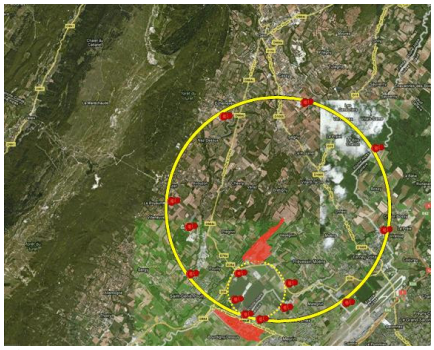


Figura 4.121. Gran colisionador de hadrones, representado mediante polilíneas

Eventos

Se utilizan para comunicar que el usuario ha realizado una acción. Un evento se activa, por ejemplo, cuando el usuario pulsa con el botón del ratón sobre un marcador, mueve el mapa, hace zoom, etc. Para aumentar la interacción se programan funciones que se ejecutan cuando uno de estos eventos se activa. Así pues, al pulsar sobre un marcador la mayoría de los mapas centran la vista en él y muestran una ventana de información.

Ventana de información

Se utiliza para ampliar la información sobre un lugar concreto o comercio y aparece superpuesta al mapa en forma de bocadillo (Fig. 4.122). Ésta suele aparecer cuando se pulsa sobre un marcador. Puede ser sencilla o complicada, contener sólo texto o código HTML, incluyendo imágenes, enlaces y otros elementos. Para ello la API dispone de una serie de funciones para mostrar y dar formato al contenido de una ventana de información.

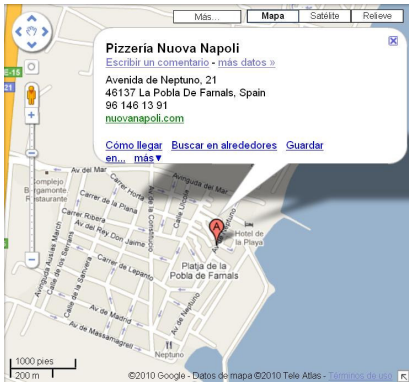


Figura 4.122. Ventana de información

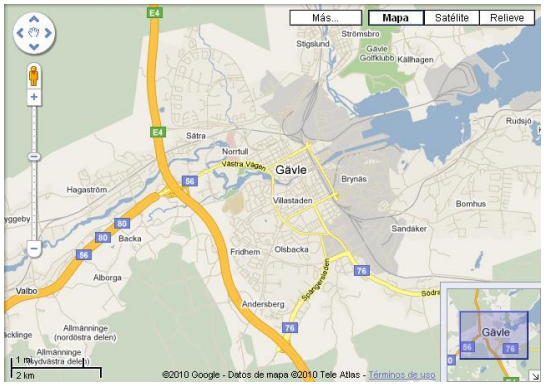


Figura 4.123. Diferentes tipos de controles en Google Maps

Controles

La API de *Google Maps* incluye una serie de controles para interactuar con el mapa (Fig. 4.123). Éstos se muestran normalmente en la parte superior izquierda y derecha del mapa e incluyen botones para moverse por el mapa, hacer zoom y elegir el tipo de vista.

Algunos de los controles existentes son:

- *GLargeMapControl*: Control de tamaño grande, con movimiento de mapa y zoom mediante barra deslizante.
- *GMapTypeControl*: Permite cambiar el tipo de vista (mapa, satélite e híbrida).
- *GScaleControl*: Muestra la escala actual en millas y kilómetros.
- *GOverviewMapControl*: Muestra un pequeño mapa de situación.

Por último, cabe resaltar que su capacidad de cargar datos dinámicamente es clave para mostrar elementos interactivos en la aplicación. La información suele estar contenida en un archivo XML almacenado en un servidor Web. Éste puede ser estático o ser generado mediante componentes dinámicos como un *script* CGI, PHP, ASP.NET o una aplicación *Java*, permitiendo así crear aplicaciones bastante avanzadas.

4.4.3. Google Earth

4.4.3.1. Introducción

En este apartado se trata la aplicación *Google Earth*. Ha sido importante aprender su manejo, conocer su funcionamiento y explotar sus capacidades, dado que más adelante se utilizará una API basada en ella para presentar todo el trabajo realizado (Apdo. 4.5.4.). Además cabe destacar que todas las marcas de posición, con sus respectivas ventanas de información relativas a los comercios de la zona, han sido generadas y maquetadas en esta misma aplicación.

Por otro lado, *Google Earth* ha servido de campo de pruebas para poder ver el aspecto final de los 46 modelos 3D generados en *SketchUp* y observar el movimiento de todo el conjunto a la vez, que como ya se explicó en el apartado 4.2.3, era inviable.

Así pues, se detallan a continuación sus principales características, su funcionamiento básico y las herramientas de que dispone para la generación de nuevos datos.

Por último se describe el proceso seguido en la creación de las marcas de posición y ventanas de información de este proyecto.

4.4.3.2. Historia

El nombre original de *Google Earth* era *Keyhole* (Fig. 4.124), y fue desarrollada en 2001 por la empresa del mismo nombre [65]. Se trataba de una aplicación que mostraba un modelo 3D de la Tierra y permitía observar cualquier lugar mediante imágenes de satélite, mapas y relieves. Además permitía superponer diferentes tipos de información geográfica (topográfica, hidrográfica, histórica y cultural, entre otras) y compartirlos con otras personas.

En 2004 Google compró la aplicación y en Mayo de 2005 pasó a llamarse *Google Earth* [78]. El programa fue lanzado en Junio de ese mismo año teniendo como principal novedad su versión gratuita. Permitía introducir el nombre de un hotel, colegio, comercio, etc., y obtener la dirección exacta, un plano y una vista del lugar, e incluso podía mostrar edificios modelados en 3D (Fig. 4.125).

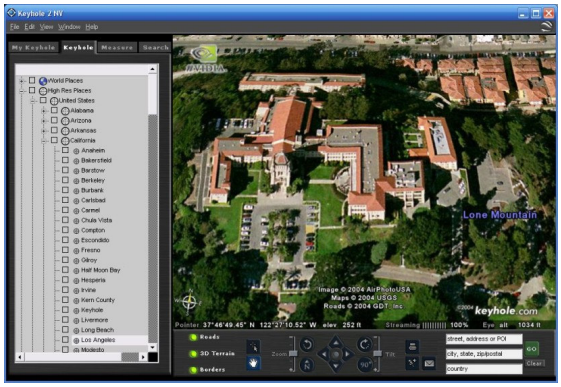


Figura 4.124. Keyhole, la aplicación original

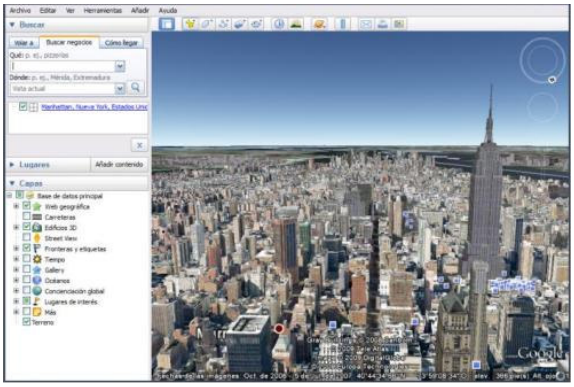


Figura 4.125. Edificios 3D en Google Earth

Más adelante Google desarrolló diversos "modos" bajo *Google Earth*:

Google Sky

Se trata de una herramienta para explorar el cielo (Fig. 4.126); fruto de un acuerdo con el Instituto de Ciencia Telescópica Espacial de Baltimore, el centro de operaciones del Telescopio Hubble, permite ver constelaciones, estrellas, galaxias y animaciones que presentan los planetas y trazan sus órbitas.

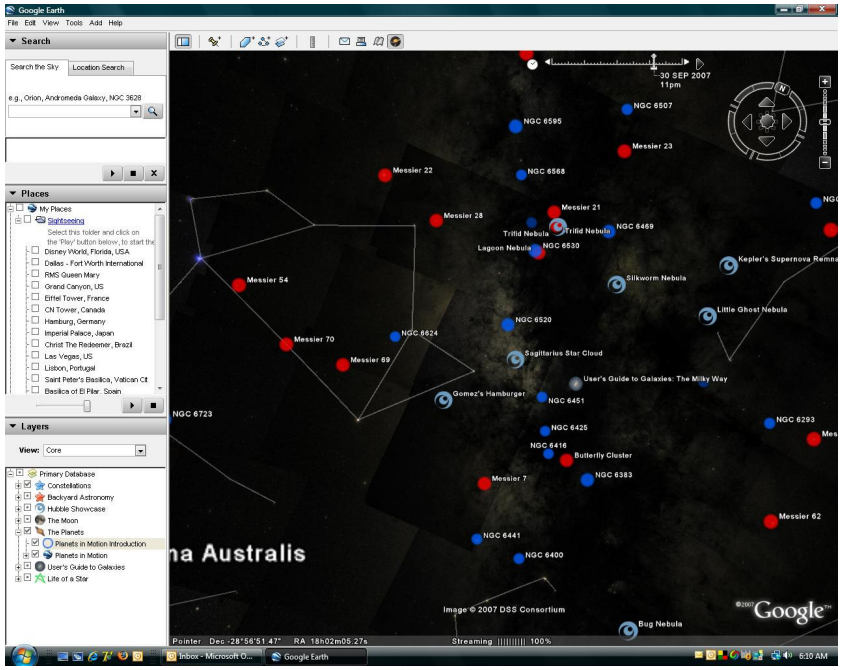


Figura 4.126. Vista del firmamento con el modo Google Sky

Google Mars

A partir de la versión 5 puede accederse a una aplicación similar a *Google Earth* basada en Marte (Fig. 4.127). Dispone de un mapa de alturas así como imágenes de alta resolución de la superficie de Marte, tanto en el espectro visible como en el infrarrojo. También pueden verse imágenes panorámicas a ras de suelo registradas por las sondas espaciales enviadas allí.

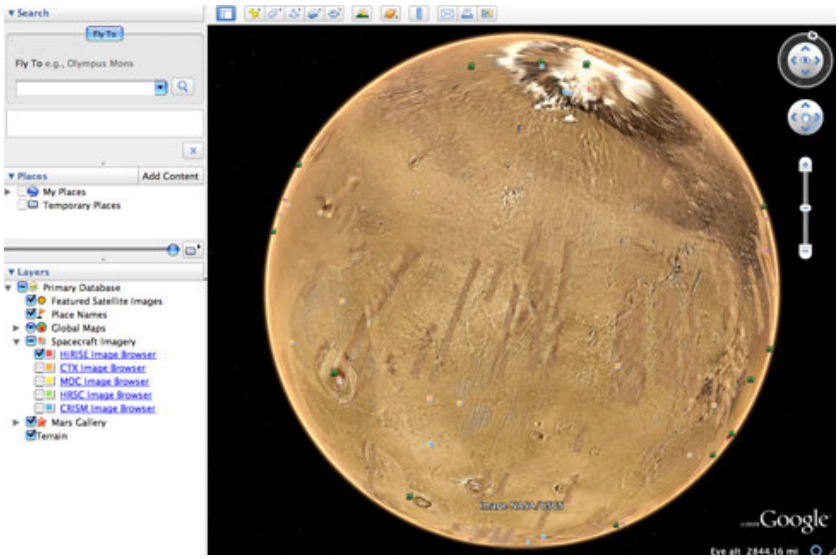


Figura 4.127. Vista de Marte con el modo Google Mars

4.4.3.3. Versiones existentes

En la actualidad pueden descargarse desde *http://earth.google.com* tres versiones de Google Earth:

Google Earth

Es la versión gratuita, para uso personal. Incorpora imágenes de satélite que abarcan la superficie completa de la Tierra. Permite medir distancias geográficas y realizar consultas sobre lugares, rutas y comercios de todo el mundo. También se pueden crear marcas de posición, polígonos, superponer imágenes al terreno y grabar rutas. Por último, toda esta información puede almacenarse y ser compartida con otras personas mediante archivos de intercambio KML.

Incorpora también una herramienta para mostrar imágenes históricas, a lo largo del tiempo. Una barra deslizante permite observar los cambios globales, la expansión de las periferias de las ciudades, las capas de hielo fundiéndose, la erosión costera, etc.

Google Earth Pro

Se trata de un producto profesional de pago, con tiempos de descarga más cortos, flujo de datos, capas ampliadas y mejoras en el sistema de anotación. Esto permite desarrollar las capas mucho más al detalle y directamente dentro de Google Earth. Accede a imágenes de mayor resolución que en su versión gratuita, y éstas pueden ser exportadas a otros documentos, presentaciones, material Web o impresos. Permite a aquellas empresas que trabajen con información georreferenciada, como inmobiliarias, aseguradoras, grupos de comunicación y otras, representar las ubicaciones resultantes de sus estudios, pudiendo importarse hasta 2500 desde una hoja de cálculo a través de sus direcciones o de sus coordenadas geográficas. El módulo de importación de datos GIS permite representar datos contenidos en archivos .shp y .tab, como planos, parcelas, listados de propiedades, datos demográficos, edificios 3D, etc.

Google Earth Enterprise

Está compuesto por varias aplicaciones que trabajan juntas para integrar, organizar y publicar aquellos datos de las empresas que estén georreferenciados. Esto permite combinar la potencia de *Google Earth* con los datos que la empresa maneje, como estadísticas, información de clientes y demás. Estas aplicaciones son:

Google Earth Fusion
Permite integrar los datos geoespaciales, las tramas (imágenes y relieve), vectores, datos KML, modelos 3D e incluso los datos almacenados en bases de datos y sistemas GIS tradicionales, y aplicarles estilos (fig. 4.128). Una vez hecho esto, se integran en el software cliente mediante el servidor de Google Earth [99]. Se pueden realizar integraciones complejas y operaciones automatizadas mediante línea de comando o interfaz gráfica y configurarse de manera que los datos se actualicen cuando existan nuevos datos disponibles.

Servidor de Google Earth
Permite almacenar datos del usuario a la vez que ofrece abundante contenido 3D y 2D propio. El usuario puede situar sus datos sobre el modelo 3D de la Tierra o generar un modelo 3D de un planeta propio e incorporar datos sobre él en forma de imágenes, relieve, vectores, etc [111]. Este nuevo planeta hecho a medida puede mostrarse mediante *Google Earth Enterprise* o en un navegador Web, en su versión 2D. Por otra parte, ofrece servicios complementarios, como la codificación geográfica, que permite unir varias bases de datos a través de una API.

Cliente de Google Earth Enterprise
Permite al usuario conectarse al planeta 3D propio de la empresa, que muestra sus datos georreferenciados y con sus propios estilos, estando todo alojado en el servidor de *Google Earth* (fig. 4.129). El complemento de búsqueda opcional permite a los usuarios consultar y buscar datos de atributo para los datos vectoriales de la empresa.

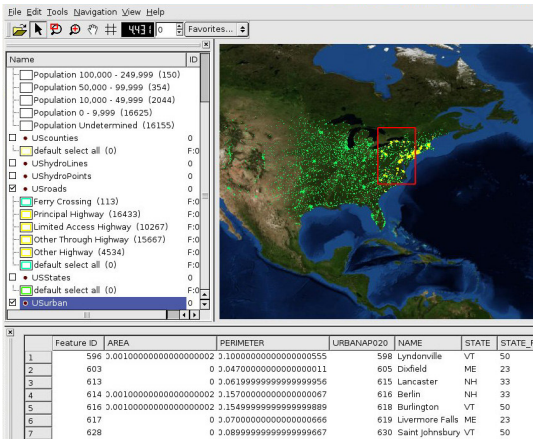


Figura 4.128. Google Earth Fusion

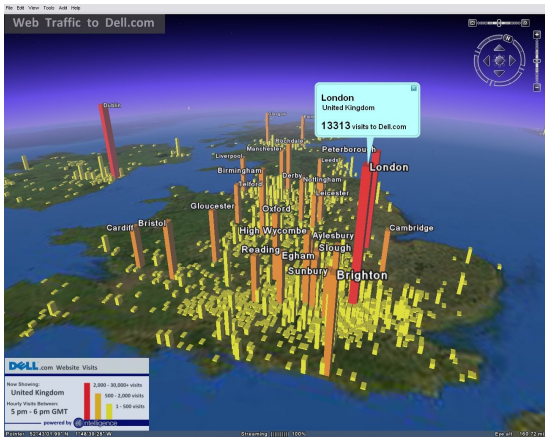


Figura 4.129. Ejemplo de Cliente de Google Earth Enterprise

4.4.3.4. Características

Su principal característica es su capacidad para poder mostrar imágenes de satélite de forma más fluida que en otras aplicaciones. Al tratarse de una aplicación independiente no presenta las limitaciones propias de un navegador Web ni de *JavaScript*, siendo así mucho más flexible y con mayor interacción con el usuario.

Sistema de referencia espacial y proyección

Para determinar la posición de un elemento, *Google Earth* utiliza como sistema de referenca espacal el WGS84, con la visualización de coordenadas en coordenadas geográficas (longitud, latitud), expresadas en grados. El datum utilizado es el World Geodetic System 1984, con elipsoide WGS 84 y meridiano central en Greenwich.

En cuanto a la altitud, todas las medidas que aparecen en *Google Earth* se toman a partir del datum vertical, que es el del geoide WGS84 EGM96, expresadas en metros.

Por último, para representar la Tierra utiliza la proyección cilíndrica simple, también conocida como Latitud / longitud WGS84.

La interfaz básica

La interfaz está dividida en tres paneles (fig. 4.130). El panel de la izquierda muestra los lugares almacenados por el usuario, las capas y una interfaz de búsqueda que permite ir a lugares concretos introduciendo información sobre cualquier pueblo, ciudad o punto de interés. El panel de la parte superior derecha, y superpuesto al mapa, muestra unos controles para moverse alrededor de la bola del mundo. Por último, el panel principal, y más grande, es la propia ventana que muestra las imágenes de satélite.

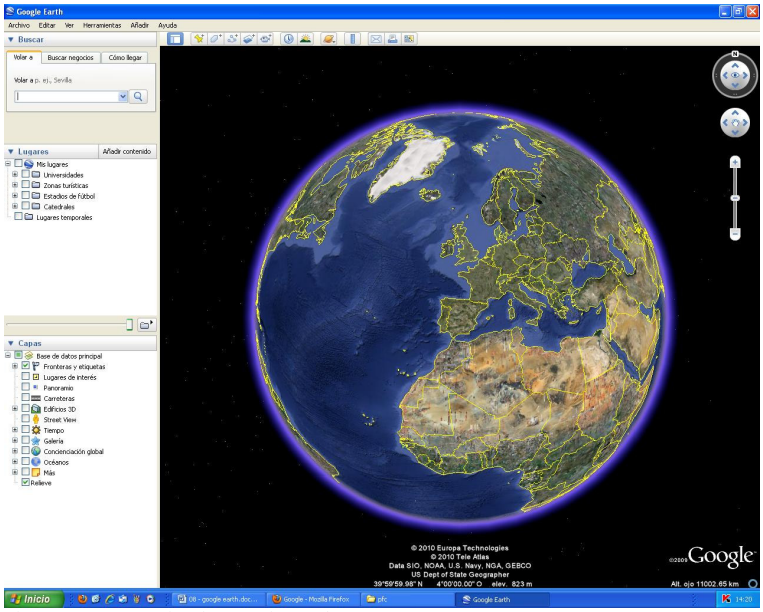


Figura 4.130. Interfaz de Google Earth

La información que aparece en la parte inferior del panel principal proporciona:

- La fecha en que fueron tomadas las imágenes y la empresa que las tomó.
- La longitud, latitud del punto central del mapa actual.
- La elevación sobre el nivel del mar del punto central.
- La altitud de la vista actual.
- El estado de las imágenes que se están descargando a la aplicación.

Navegación

La navegación por el mapa puede realizarse de diferentes maneras:

Ratón
Manteniendo pulsado el botón izquierdo del ratón puede arrastrarse el mapa en la dirección que se desee. Para detenerse en un lugar concreto basta con soltar el botón pulsado. Si esto mismo se hace rápidamente y se suelta el ratón el mapa mantendrá el movimiento en la dirección que se haya marcado. Para pararlo basta con pulsar sobre el mapa con el ratón. Mantener pulsada la tecla “Ctrl” a la vez que se pulsa el botón izquierdo del ratón permite el manejo del cabeceo de la cámara sin variar su posición. Con el botón central del ratón se maneja la rotación e inclinación del mapa. Con el botón derecho del ratón se aleja o acerca el mapa.

Teclado

Los cursores del teclado mueven el mapa en la dirección que se desee. Manteniendo pulsada la tecla "mayúsculas" y los cursores izquierda o derecha se rota el mapa en sentido horario y antihorario. Si en su lugar se pulsan los cursores arriba y abajo se varía la inclinación del mapa. Manteniendo pulsada la tecla "Ctrl" y los cursores permite manejar el cabeceo y rotación de la cámara sin variar la posición de la misma. Para reorientar el mapa hacia el Norte se pulsa la tecla "N". Para manejar el zoom en el mapa se usan las teclas "+" y "-" o "RePág" y "AvPág".

Panel de navegación

Con las flechas que llevan dibujado un ojo en su centro (fig. 4.131) se manejan el cabeceo y la rotación de la cámara sin variar su posición. Las que llevan una mano dibujada en su centro permiten moverse por el mapa. La barra deslizante permite controlar el zoom. Se puede saber la orientación del mapa en cada momento ya que hay dibujada una brújula alrededor del panel de navegación superior. Para reorientar el mapa hacia el Norte se pulsa sobre el botón "N" de la misma.

Capas

La aplicación dispone de una serie de capas (fig. 4.132) que pueden activarse y que contienen la siguiente información sobre un mismo espacio geográfico:

- Nombres de países, ciudades, pueblos y calles.
- Puntos de interés: hoteles, restaurantes, gasolineras, hospitales, etc.
- Fronteras, límites provinciales, carreteras y vías férreas.
- Volcanes, epicentros de sismos, lagos, lagunas y ríos, entre otros.
- "*Terreno*". Activa el relieve del terreno.
- Sitios relacionados con estudios de la "*National Geographic*".
- "Edificios 3D". Edificios modelados con *SketchUp*.
- "*Street view*". Imágenes esféricas a pie de calle.
- Predicción meteorológica.
- "*Galería*". Vídeos, fotografías, artículos, etc. de los usuarios.
- "*Océanos*". Contenidos de la BBC y National Geographic sobre océanos.

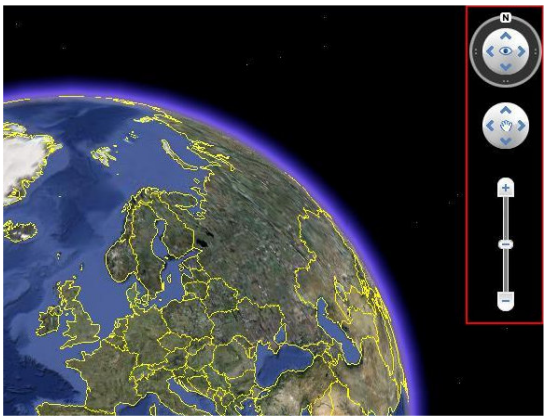


Figura 4.131. Controles de navegación en *Google Earth*

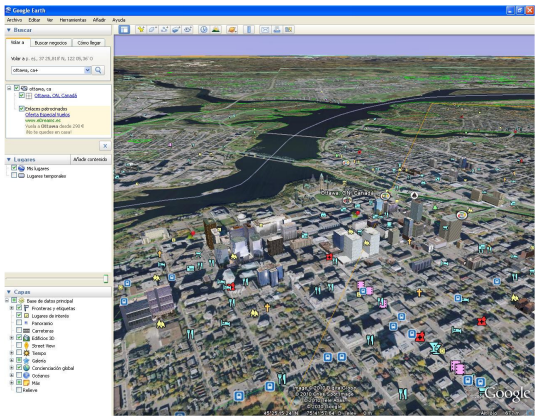


Figura 4.132. Ejemplo de diversas capas activadas

Lugares

Además de visitar los numerosos lugares importantes del mapa, que son actualizados y añadidos continuamente, es posible almacenar nuevos lugares favoritos propios y organizarlos por carpetas (fig. 4.133).

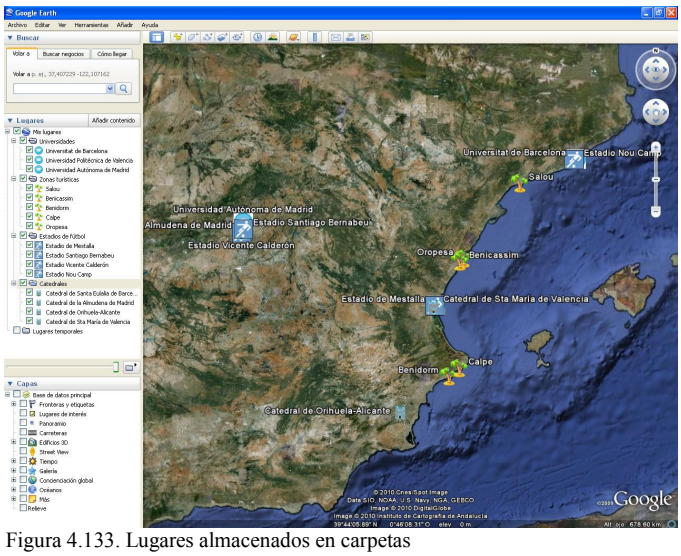


Figura 4.133. Lugares almacenados en carpetas

Buscador y calculador de rutas

Google Earth permite volar a países, ciudades y pueblos de todo el mundo introduciendo su nombre o una dirección en su buscador. También se pueden buscar comercios y lugares de interés escribiendo su nombre o especificando la localidad en la que están situados.

Por otro lado dispone de un calculador de rutas que funciona introduciendo una dirección de salida y otra de llegada o indicando las marcas de posición de ambos sitios (fig 4.134). Activando la capa "carreteras" éstas aparecerán representadas por encima de las imágenes de satélite. Si además se activa la capa "terreno" se incluirá información sobre la orografía del terreno, apareciendo las carreteras adaptadas a esta, es decir que estará en 3D.

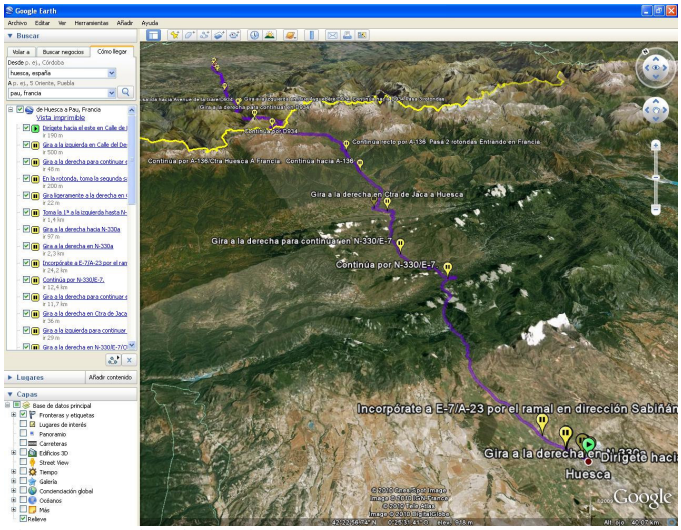


Figura 4.134. Ejemplo de cálculo de ruta (Huesca - Pau)

4.4.3.5. Añadir nueva información

La aplicación incorpora una extensa librería propia de datos, capas, puntos de interés, edificios 3D, etc., que pueden superponerse al mapa, pero también permite crear nueva información. Toda esta nueva información se almacena en archivos KML, un tipo especial de archivo XML (se tratará más adelante), que permite construir y organizar las marcas de posición, rutas, polígonos etc. y compartirlas con otras personas.

Existen dos maneras de incorporar nueva información al mapa. Una es mediante las herramientas de que dispone la aplicación en el panel superior, para crear nuevas marcas de posición, rutas, polígonos, imágenes superpuestas, etc. La otra es creando los archivos KML a través de un editor de textos externo.

Añadir marca de posición

Mediante esta herramienta se crea una nueva marca de posición sobre algún lugar que se considere de interés. Su interfaz permite introducir la posición del lugar, así como información detallada del mismo mediante texto sencillo o mediante el uso de código HTML (fig. 4.135), lo que permite mostrar imágenes que estén en Internet a través de su URL, y enlaces a páginas Web, entre otros (fig. 4.136 y 4.137).

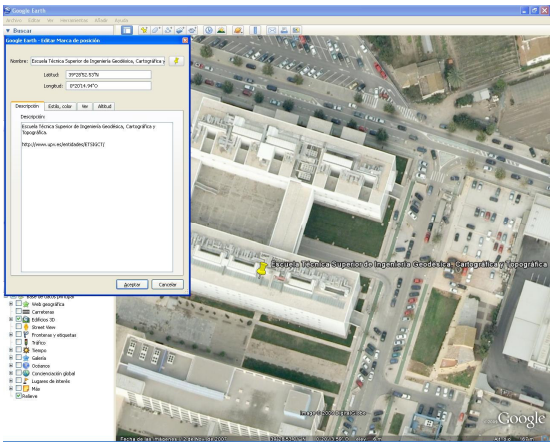


Figura 4.135. Edición de una marca de posición

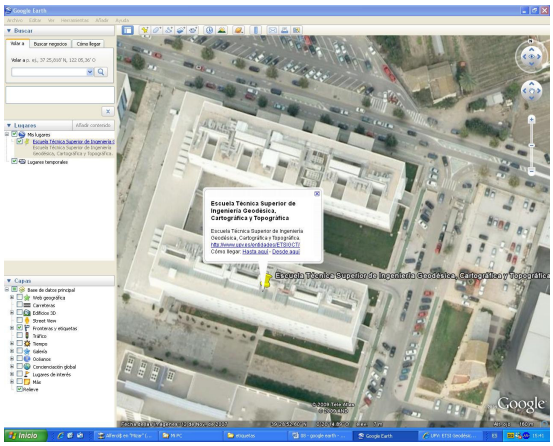


Figura 4.136. Aspecto final de una marca de posición sencilla

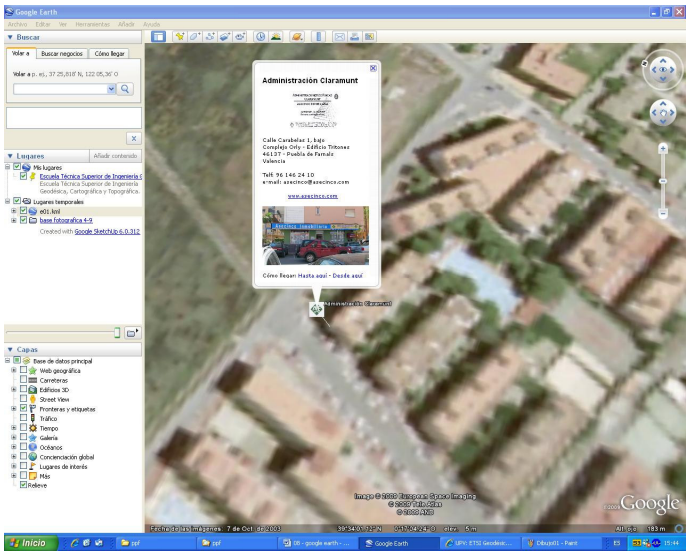


Figura 4.137. Marca de posición con código HTML

El siguiente código HTML pertenece a la marca de posición anterior:

```
<FONT FACE="verdana" SIZE=1>
<center><IMG SRC="http://personales.alumno.upv.es/alferol/farnals/ppf/imagenes/01t.jpg"
width=100></center>
<br>Calle Carabelas 1, bajo
<br>Complejo Orly - Edificio Tritones
<br>46137 - Puebla de Farnals
<br>Valencia
<br>
<br>Telf: 96 146 24 10
<br>e-mail: asecinco@asecinco.com
<br>
<br><center><a HREF="http://www.asecinco.com" TARGET="_BLANK">www.asecinco.com</a>
<br>
<br>
<IMG SRC="http://personales.alumno.upv.es/[...]/01-f.jpg" width=200></center>
```

La interfaz también permite modificar el color, escala y opacidad de la etiqueta que aparece señalando el lugar (fig. 4.138).

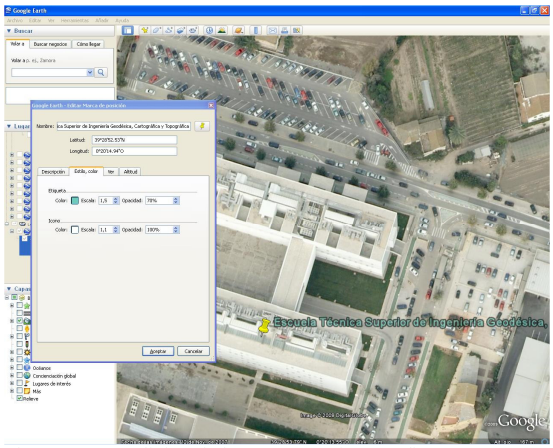


Figura 4.138. Edición de las características gráficas

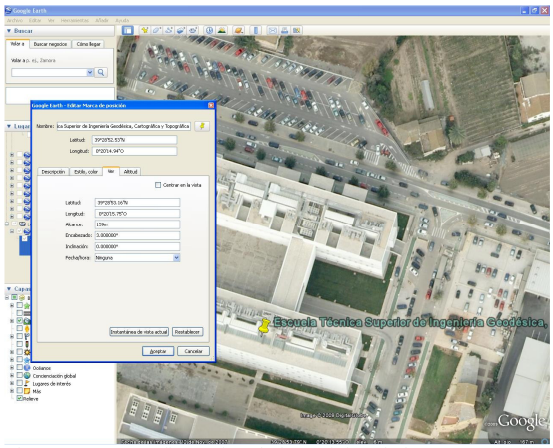


Figura 4.139. Configuración de la posición de la cámara

En toda marca de posición queda almacenada, en el momento de su creación, la configuración de la vista actual de la cámara, esto es, su posición, altitud, orientación e inclinación en ese momento, pero ésta puede ser modificada en cualquier momento a través de la interfaz (fig. 4.139). Ésta será la vista que ofrecerá la cámara cada vez que se elija esta marca de posición.

En las siguientes imágenes se especifica si el icono de la marca de posición estará sujeto al suelo o a cierta altura, relativa al suelo o absoluta:

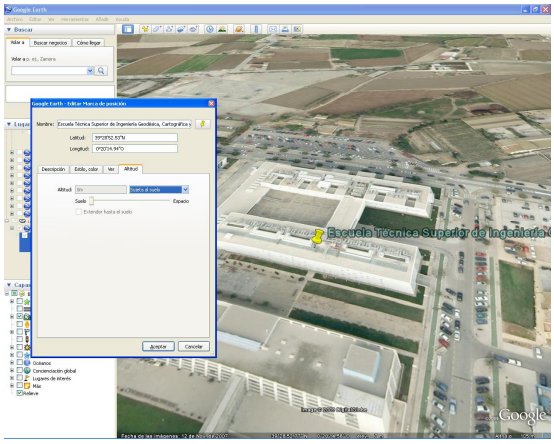


Figura 4.140. Ejemplo de icono sujeto al suelo

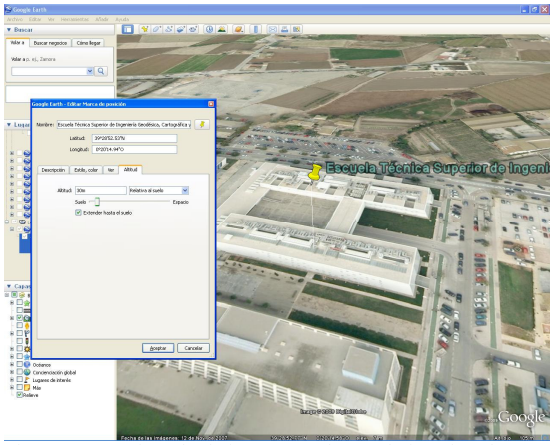


Figura 4.141. Ejemplo de icono con altura relativa al suelo

A toda marca de posición va asociado un icono. La aplicación dispone de unos cuantos predeterminados (fig. 4.142). También es posible añadir como nuevo icono una imagen que se encuentre disponible en Internet introduciendo su URL (fig. 4.143). En esta misma ventana puede modificarse su color, escala y opacidad (Fig. 4.144).

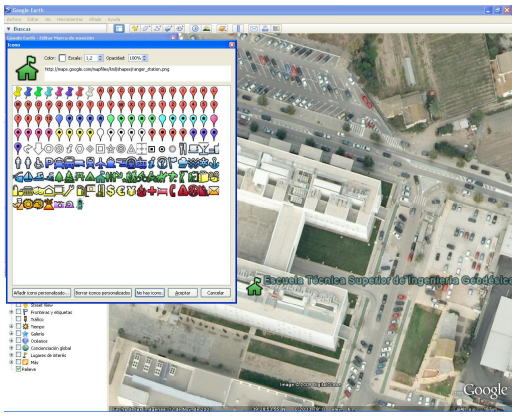


Figura 4.142. Ejemplo de icono de Google Earth

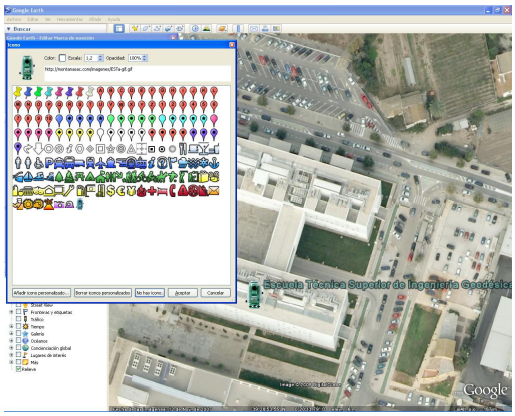


Figura 4.143. Ejemplo de icono enlazado a través de su URL

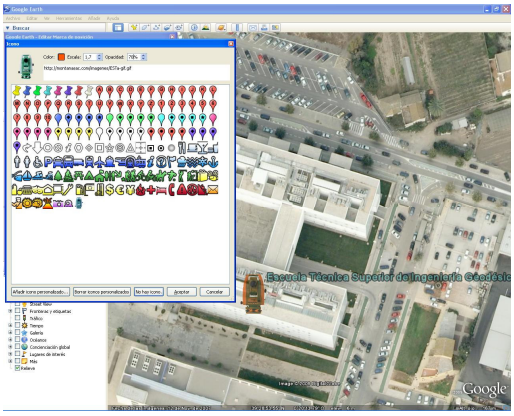


Figura 4.144. Edición de las características gráficas del icono

Añadir polígono

Esta herramienta permite dibujar polígonos, que pueden representar límites administrativos, parcelas, subparcelas, etc. En este caso se dibujará la manzana correspondiente a una zona residencial. La ventana de información asociada permite el uso tanto de texto sencillo como de HTML (Fig 4.145).

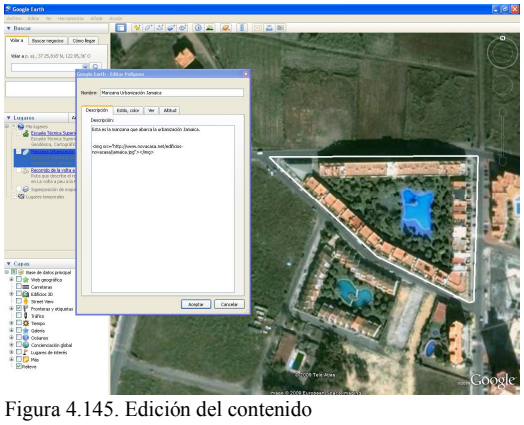


Figura 4.145. Edición del contenido


Figura 4.146. Ajuste de propiedades gráficas

Puede modificarse el grosor y el color de las líneas que lo forman (fig. 4.146).

Se puede determinar la altura a la que el polígono está situado, y si permanece tal cual o aparece extrudido, de manera que se extienda hasta alcanzar el suelo (fig. 4.147 y 4.148).

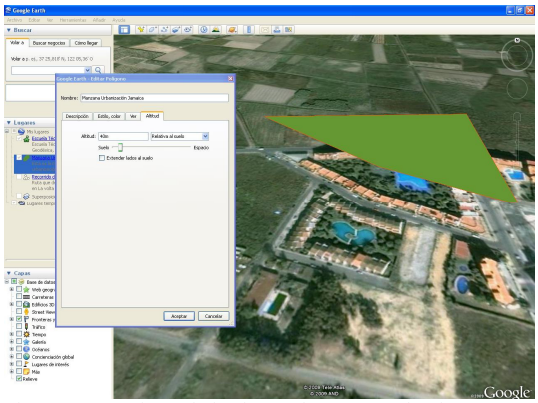


Figura 4.147. Ajuste de la altura a la que está el polígono

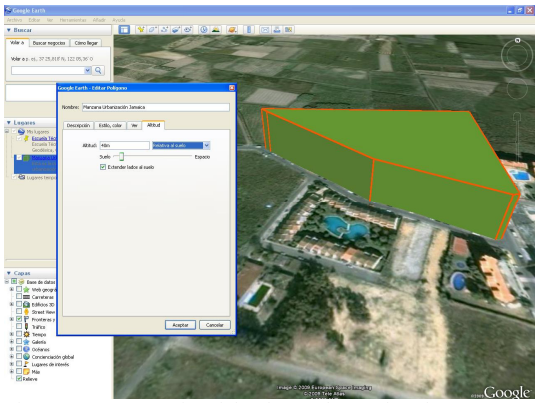


Figura 4.148. Extrusión del polígono hasta el suelo

Y por último, se puede ajustar su opacidad (fig. 4.149), dejando así ver los elementos e imágenes que estén situados por debajo de él.

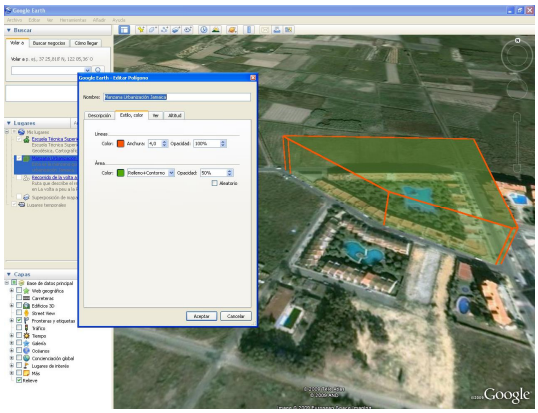


Figura 4.149. Ajuste de la opacidad del polígono

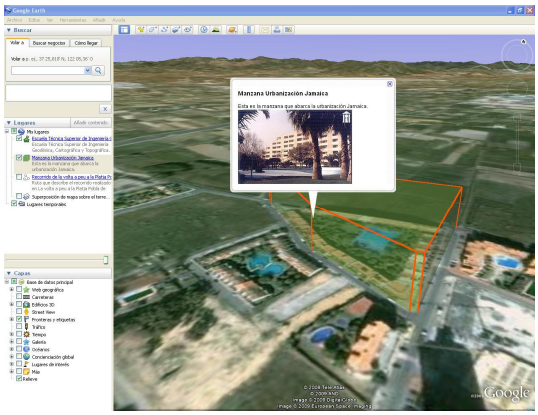


Figura 4.150. Polígono con su ventana de información

Su aspecto final, habiendo realizado los ajustes anteriores e incluyendo algo de texto sencillo y HTML con enlace a una imagen del lugar, sería el que muestra la figura 4.150.

Añadir ruta

La herramienta que se presenta a continuación está diseñada para poder trazar viajes, recorridos, rutas, caminos, senderos etc. Guarda cierto parecido con la que se acaba de explicar, con la salvedad de que la polilínea no se cierra formando un polígono; permanece abierta.

Como complemento a lo anterior, una vez terminada la ruta, puede hacerse un recorrido que desplazará la cámara a lo largo de ella de forma automática, seleccionando "reproducir ruta".

Igual que en anteriores elementos, también puede incluirse información detallada de la ruta en su ventana de información mediante texto sencillo o código HTML.

En el caso que se presenta a continuación se ha creado el recorrido de una supuesta carrera popular ficticia.


Figura 4.151. Edición de la información de la ruta


Figura 4.152. Ajuste de propiedades gráficas de la ruta

Lo primero que se hace es marcar los vértices que configuran la ruta.

Una vez hecho se pasa a ajustar las propiedades gráficas de la misma. Se establece el grosor de las líneas y el color de líneas y áreas. Estas áreas son las que aparecen al extrudir la ruta. De igual manera se configura la opacidad de ambas (Fig. 4.152).

La ruta puede representarse a cierta altura sobre el terreno y aparecer extrudida hasta éste (Fig. 4.153).

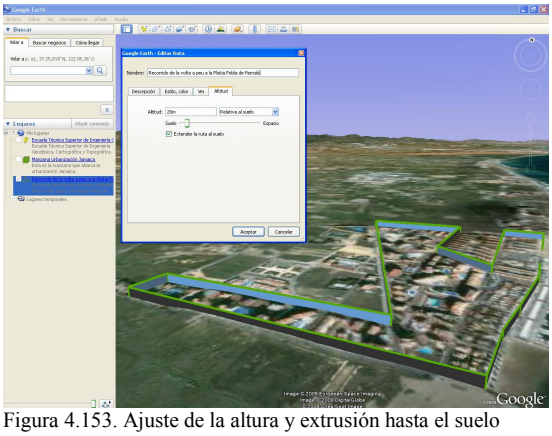


Figura 4.153. Ajuste de la altura y extrusión hasta el suelo


Figura 4.154. Ejemplo de ruta y su ventana de información

Por último, para el ejemplo que se muestra, se ha editado una ventana de información con detalles del evento. Ésta incluye tanto texto sencillo como código HTML, así como una imagen enlazada desde la red. Su aspecto final puede apreciarse en la figura 4.154.

Añadir superposición de imagen

Esta herramienta permite superponer una imagen al terreno. Ésta puede estar almacenada en un servidor y ser enlazada mediante su URL o ser cargada desde el disco duro. En el ejemplo que se muestra se ha encontrado en la Web una imagen de un mapa de carreteras de la zona de estudio que puede servir como ejemplo para superponer al terreno. De esta manera, se introduce la URL en la que se encuentra la imagen (fig. 4.155) y se carga. Ésta aparecerá ajustada a la vista actual y orientada al Norte.

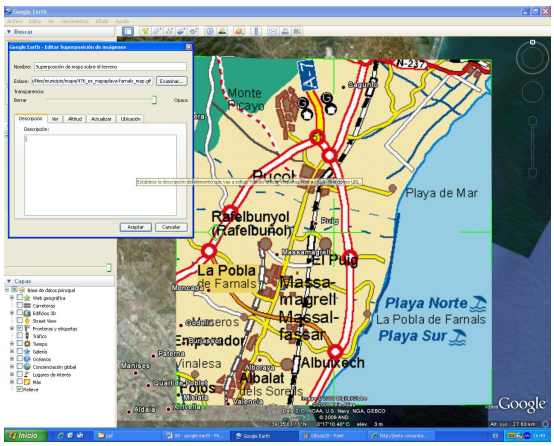


Figura 4.155. Introducción de la URL de la imagen

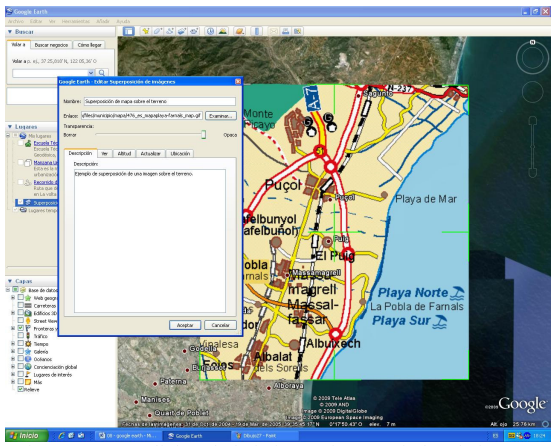


Figura 4.156. Edición de la información asociada a la imagen

Este elemento permite que se le asocie una ventana de información (Fig. 4.156). No obstante, a diferencia de otros elementos, éste no la mostrará al pulsar sobre la imagen o su marca de posición. Para acceder a su contenido será necesario desplegar el menú contextual y seleccionar en la opción “propiedades”.

A continuación se ajustarán sus características gráficas. Normalmente se busca que la imagen se amolde al relieve (Fig. 4.157) aunque también es posible mostrarla flotando por encima del suelo (Fig. 4.158), estableciendo para ello su altura y si ésta es respecto al suelo o absoluta. En este ejemplo se presentará adaptada al terreno.

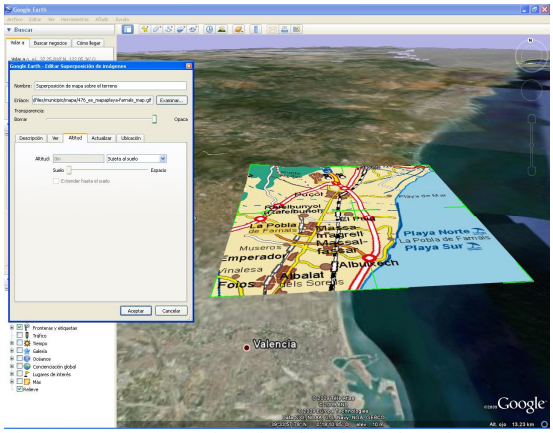


Figura 4.157. Imagen sujeta al suelo, adaptándose al terreno

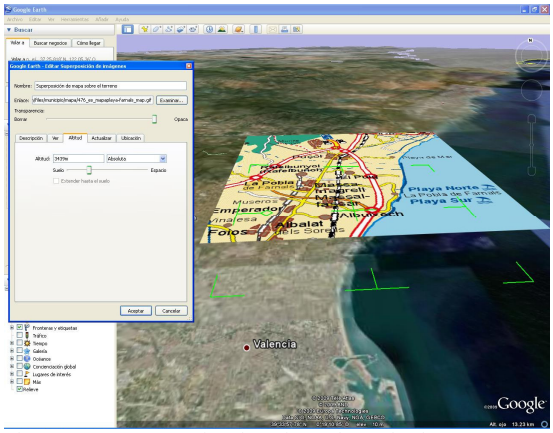


Figura 4.158. Imagen situada a cierta altura absoluta

Por otra parte, cuando se carga la imagen ésta no aparece bien posicionada ni escalada. Para hacer este ajuste y presentarla correctamente se introducirán las coordenadas de sus lados Norte, Sur, Este y Oeste (Fig. 4.159). De esta manera puede apreciarse en la figura 4.160 el aspecto final de la imagen superpuesta, una vez posicionada y adaptada al terreno.

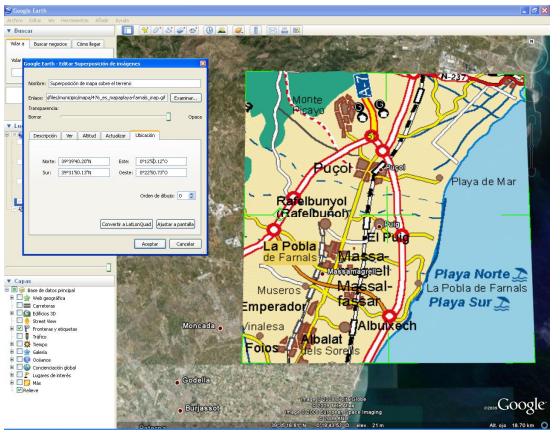


Figura 4.159. Ajuste de la posición de la imagen

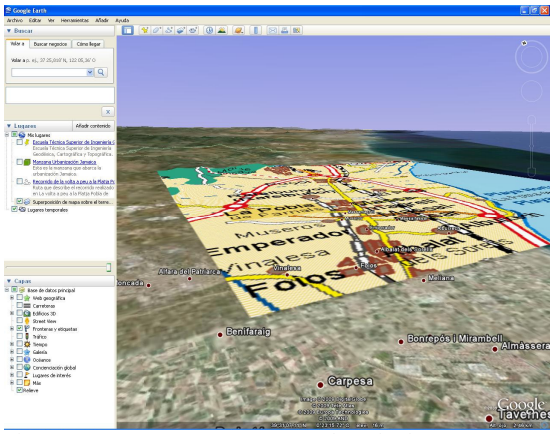


Figura 4.160. Imagen superpuesta y adaptada el terreno

Mostrar Edificios 3D

Como se comentó anteriormente en sus características, *Google Earth* dispone de una capa llamada "Edificios 3D". Ésta muestra modelos 3D de edificios, almacenados en los servidores de Google, y que han sido creados por usuarios utilizando principalmente el programa de modelado 3D *Google SketchUp*, que es el programa utilizado para la realización de modelados en este proyecto.

Esto significa que los edificios e instalaciones que se han modelado en 3D pueden mostrarse en esta aplicación. Para ello es necesario exportar los modelos desde *SketchUp* como archivos .kml para luego abrirlos en *Google Earth*. Como se ha comentado a lo largo de esta memoria, éste fue el punto, en la fase de alta calidad, en el que se comprobó que los modelos debían ser remodelados de manera que consumieran menos memoria, dado que el movimiento de todo el conjunto era lento. Afortunadamente, en la fase de baja calidad se pudo comprobar que habían sido correctamente remodelados y que en conjunto se movían rápidamente, como fue la intención desde el principio.

Por otro lado, en el primer momento de importación de los modelos se descubrió que éstos parecían no estar bien localizados a pesar de que la cartografía utilizada para su modelado estaba correctamente georreferenciada. Tras varias revisiones se tuvo la certeza de que las imágenes de satélite de *Google Earth* en algunos casos no están correctamente orientadas o solapadas. Este problema fue comunicado a Google junto con otro problema de solape erróneo de fotografías en la zona de estudio. A fecha de Mayo de 2010 el problema de solape ha sido resuelto, pero las imágenes siguen apareciendo mal colocadas. Es por eso que se optó por superponer en las coordenadas correctas una ortofoto adquirida en el Instituto Cartográfico Valenciano [113], de manera que sirviera de base para que los modelos de los complejos aparecieran definitivamente en su sitio, sin crear confusiones.

La siguiente es una vista general de la zona de estudio:

Figura 4.161. Edificios 3D en la zona de la Playa de la Puebla de Farnals

A continuación, varias vistas de urbanizaciones y complejos deportivos:

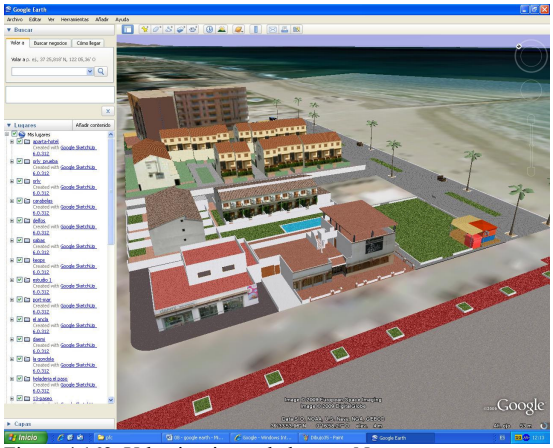


Figura 4.162. Urbanizaciones de la zona Norte

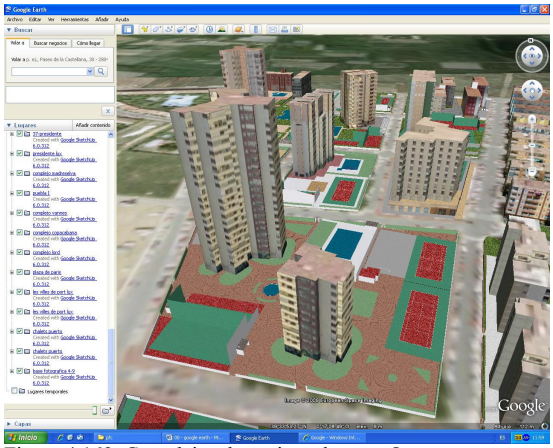


Figura 4.163. Complejos deportivos de zona Oeste

Y por último, unas imágenes más detalladas:

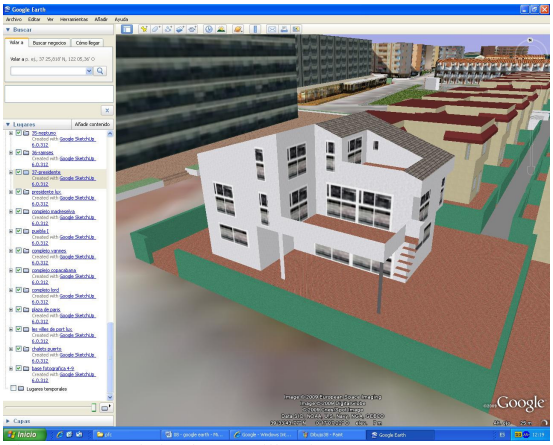


Figura 4.164. Chalet de Les Villes de Port-Lux

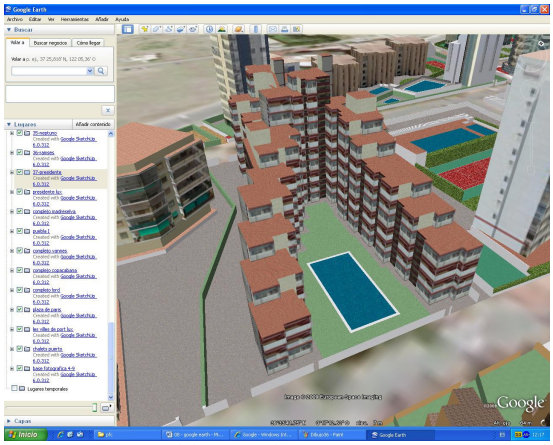


Figura 4.165. Complejo Deportivo Ramsés

4.4.3.6. Creación de marcas de posición propias

Como complemento a los diversos complejos deportivos que fueron diseñados para este proyecto se pensó agregar información sobre los comercios y los diferentes servicios públicos que se encuentran en la zona de estudio (consultorio de la Seguridad Social, oficina de Turismo, Policía Local, Posta Sanitaria, Paradas de Autobús, etc.)

Para ello se utilizó la herramienta de edición de marcas de posición de *Google Earth*, que permite no sólo señalar la posición en la que se encuentra el comercio en cuestión, sino también aportar información sobre el mismo, como es su dirección postal, teléfono, horario, descripción, tarjeta de visita, logotipo, enlace a página Web propia, fotografía del comercio, etc. mediante una ventana de información.

Búsqueda de la información

Parte de la información ha podido adquirirse en los propios establecimientos, como ha ocurrido con sus tarjetas de visita y publicidad de los mismos. Otras fuentes de información han sido publicaciones locales, que son editadas anualmente y que incorporan gran cantidad de anuncios de los comercios de la zona. En éstas aparecen principalmente las tarjetas de visita de los comercios y en ocasiones publicidad más extensa sobre los mismos. Toda esta información fue escaneada para poder incorporarla en sus marcas de posición.

Varios comercios disponen de páginas Web propias, de las que también se extrajeron imágenes e información y a las que se enlazó desde las marcas de posición.

En Internet se buscó información sobre aquellos comercios de los que no se tenía ni tarjeta de visita ni logotipo ni publicidad. En algunos casos se pudo encontrar algo de información sobre los mismos, y en otros, por tratarse de comercios muy sencillos, no se consiguió más que su dirección. Como última alternativa se consiguió un logotipo del comercio a partir de su letrero, donde suele figurar su nombre y en ocasiones un logotipo, como puede comprobarse en las siguientes imágenes:

Figura 4.166. Imagen original del comercio

Figura 4.167. Logotipo del comercio

Figura 4.168. Logotipos obtenidos a partir de las imágenes de las fachadas de diferentes comercios

Como complemento final a toda la información recabada se tomaron fotos a las fachadas de todos los comercios a pie de calle (fig. 4.169 y 4.170), para mostrar el aspecto de los mismos y facilitar su identificación.

Figura 4.169. Fachada de la tienda de náutica "Jet Spit"

Figura 4.170. Fachada de la Pizzería "Nuova Napoli"

Una vez acabado este proceso se advirtió que había información redundante de algunos comercios (fig. 4.171 y 4.172). Se tenían diferentes tarjetas de visita y publicidad variada para un mismo comercio, por lo que hubo que seleccionar la información que iba a utilizarse.

Figura 4.171. Varios diseños para la tarjeta de visita del Restaurante "La Dolce Vita"

Figura 4.172. Diferentes tipos de tarjeta de visita para la Pizzería "Nuova Napoli"

En las siguientes imágenes puede apreciarse el diseño de maquetación elegido para la ventana de información de cada marca de posición. Ésta empieza mostrando el nombre del comercio. Luego se muestra la tarjeta de visita o logotipo del mismo, si los tiene. A continuación se muestra la dirección, teléfono, fax, e-mail, enlace a página Web, descripción del comercio, etc. Por último se muestra una fotografía de la fachada del mismo.

Se optó por este diseño, que intenta ocupar el menor espacio vertical posible, para evitar la aparición de la barra de desplazamiento vertical, que no muestra las imágenes enteras y resulta poco estética. Esto ha hecho que se hayan preferido aquellas tarjetas de visita, logotipos e imágenes que eran apaisadas antes que otras alargadas. A continuación varios ejemplos de su aspecto final:

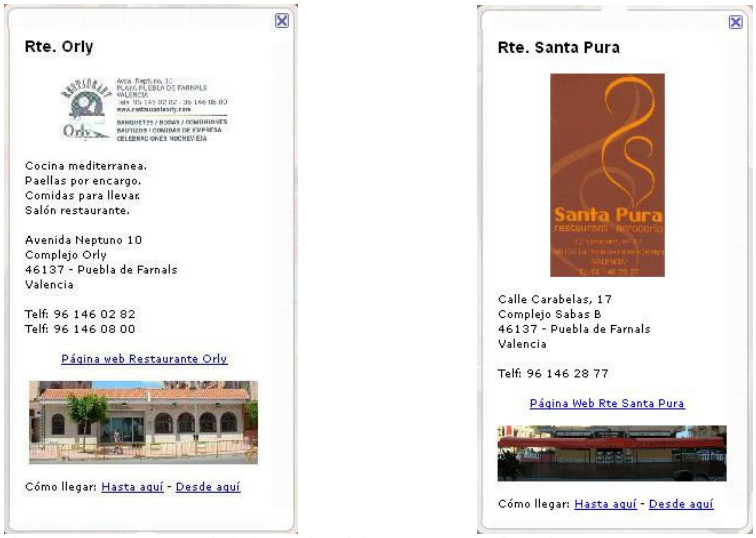


Figura 4.173. Ventanas de información del Restaurante Orly y el Restaurante Santa Pura

Búsqueda y selección de iconos

Dado que a las marcas de posición se les puede asociar un icono, en la zona de estudio de este proyecto se han clasificado los comercios en las siguientes 27 categorías con la idea de buscar para cada una un icono que la identifique:

- Administración de fincas
- Asesoría de empresas
- Bar – Restaurante
- Bazar
- Boutique – Tienda de moda
- Cajero automático
- Cibercafé
- Estanco
- Estética – Peluquería
- Falla
- Farmacia
- Feria
- Frutería
- Heladería
- Inmobiliaria
- Kiosco
- Médico
- Náutica
- Oficina de Turismo
- Parada de autobús
- Pastelería
- Pizzería
- Policía Local
- Pub
- Puerto Deportivo
- Reformas
- Supermercado

Se buscó en la red, obteniendo gran cantidad de iconos para cada categoría:

Figura 4.174. Ejemplos de iconos para el tipo de comercio “Estética / Peluquería”

A algunos comercios no fue difícil encontrarles un icono, como es el caso del colegio de administradores de fincas, los estancos, oficinas de turismo, médico o farmacia, dado cada una de sus categorías tiene un icono estándar (fig 4.175).

Figura 4.175. Comercios y servicios con iconos propios

Otros eran fácilmente reconocibles por todo el mundo:

Figura 4.176. Iconos de fácil reconocimiento

Se eligieron de entre más de 500 iconos aquéllos que mejor representaban a cada categoría, intentando que fueran lo más sencillos posible y que no dieran lugar a equívocos. Los iconos utilizados en las marcas de posición fueron los siguientes:

Figura 4.177. Iconos empleados en las marcas de posición de los diferentes comercios

Como nota, cabe destacar que también se buscó un icono que identificara los complejos deportivos.

Figura 4.178. Icono de complejo deportivo

Se trata de una categoría, la número 28, que por no tratarse de un comercio se comenta aparte. Corresponde a las marcas de posición que señalan la situación de los diferentes complejos deportivos y a los que se le asoció también un icono.

Edición y organización de las marcas de posición

En todas ellas se escribió código HTML, lo que facilitó la maquetación de la ventana de información, pudiendo incluirse enlaces a las páginas Web de los comercios y mostrar las imágenes, iconos y fotografías que se habían escaneado y que habían sido alojadas en un servidor para poder ser enlazadas. A continuación el código HTML empleado en la marca de posición del Restaurante Orly.

```
<FONT FACE="verdana" SIZE=1>
<center><IMG SRC="http://personales.alumno.upv.es/alferol/farnals/ppf/imagenes/81t.jpg"
width=150></center>
<br>Cocina mediterranea.
<br>Paellas por encargo.
<br>Comidas para llevar.
<br>Salón restaurante.<br>
<br>Avenida Neptuno 10
<br>Complejo Orly
<br>46137 - Puebla de Farnals
<br>Valencia<br>
<br>Telf: 96 146 02 82
<br>Telf: 96 146 08 00
<br>
<br><center><a href="http://www.playapuebladefarnals.com/orly.html"
target="_blank">Página web Restaurante Orly</a></center>
<br>
<center><IMG SRC="http://personales.alumno.upv.es/alferol/farnals/ppf/imagenes/81f.jpg"
width=200></center>
```

Por otro lado, se estableció la configuración de la vista de la cámara en cada una de las marcas de posición para que, cada vez que se eligiera una, ésta mostrara claramente tanto el comercio al que hace referencia como su ventana de información, como ocurre con el Restaurante Orly en la siguiente imagen:

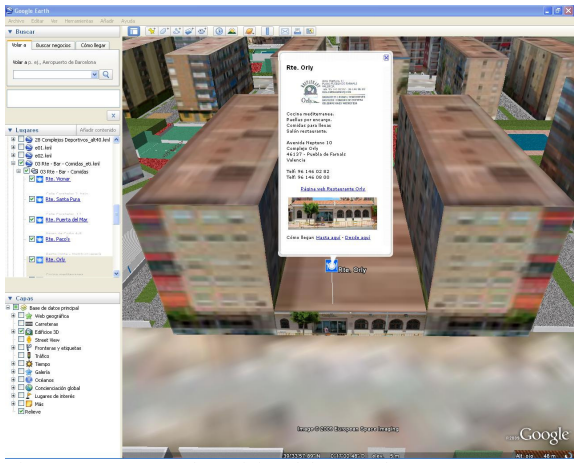


Figura 4.179. Marca de posición del Rte. Orly

A continuación se ilustran varios ejemplos de marcas de posición de diferentes comercios, con la maquetación comentada anteriormente y sus respectivas posiciones de cámara:

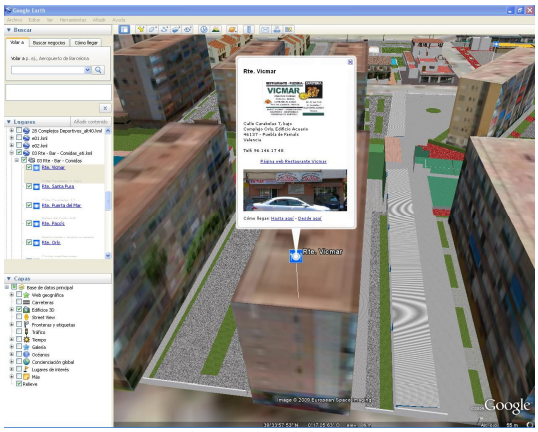


Figura 4.180. Marca de posición del Restaurante Vicmar

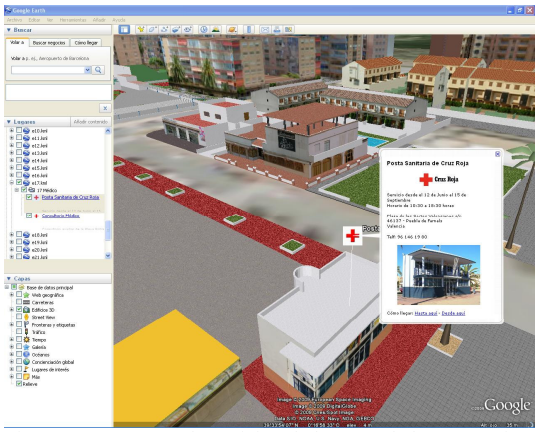


Figura 4.181. Marca de posición de la Posta de la Cruz Roja

Figura 4.182. Marca de posición del Consultorio de la S.S.

Figura 4.183. Marca de posición de la Oficina de Turismo

Figura 4.184. Marca de posición de una parada de autobuses

Figura 4.185. Marca de posición de un cajero automático

Por último, para cada categoría de comercio se ha creado una carpeta en el panel "Lugares", siendo en total 28 (fig. 4.186). Todas las marcas de posición pertenecientes a una misma categoría han sido introducidas en su carpeta correspondiente y finalmente ésta ha sido guardada en un archivo en formato .kml. Esto permitirá trabajar con él en un editor de textos, cargarlo en *Google Earth* y poder activar y desactivar cada categoría de comercio por separado.

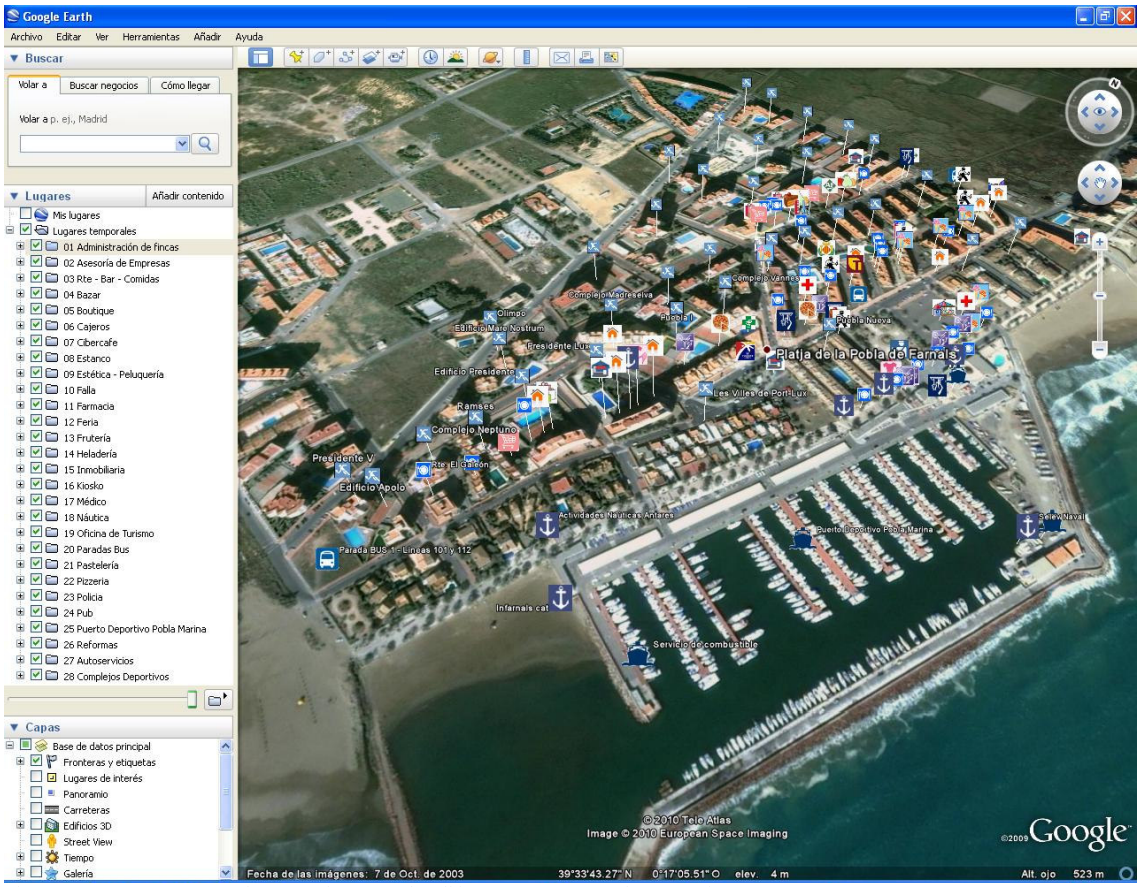


Figura 4.186. Carpetas creadas para almacenar las diferentes categorías de comercios

4.5. Desarrollo de la aplicación Web

4.5.1. Introducción

El desarrollo de la aplicación Web constituye el último apartado en la elaboración de este proyecto. En ésta se presentan los modelos 3D creados anteriormente y la información adicional asociada a ellos. Para poder trabajar con estos elementos ha sido necesario conocer el funcionamiento de cierto tipo de archivos, diseñados para almacenar este tipo de información, como son XML y KML. Así pues, antes de entrar en el desarrollo propio de la aplicación Web, se presentan primero estos dos lenguajes de marcas, describiendo sus características, métodos y tecnologías asociadas y sus aplicaciones. En el caso de KML se describen además las etiquetas que se han empleado en la edición de las marcas de posición y sus ventanas de información correspondientes, para a continuación describir el funcionamiento del archivo KMZ, que ha sido el encargado de contener los modelos 3D del proyecto.

En segundo lugar se presenta la API de *Google Earth*, gracias a la cual puede mostrarse un modelo 3D de la Tierra que represente sobre su superficie los modelos de la zona de estudio y muestre a su vez las marcas de posición. Es por eso que en este apartado se presentará esta API y se explicarán sus características y el funcionamiento de las diferentes interfaces utilizadas para desarrollar la aplicación Web de este proyecto.

A continuación, un nuevo apartado muestra el código XHTML y *JavaScript* empleado en la elaboración de la página Web, que contiene las llamadas a la API de *Google Earth*. Se comentan sus diferentes partes y la tarea que realiza cada una de las funciones que se han programado. Éstas dotarán a la página Web de sus características finales, pudiendo realizar consultas sencillas sobre los comercios de la zona y reproducir varias rutas programadas.

Por último se trata el diseño de una segunda página Web, diseñada para poder mostrar varios modelos en las fases de alta y baja calidad a la vez y comparar así el trabajo realizado en ambas.

4.5.2. XML

4.5.2.1. Introducción

Durante el desarrollo de este proyecto se ha trabajado con archivos KML y XML. Los primeros, como se explicará en el apartado 4.5.3, son una clase de archivo XML que han sido utilizados para contener tanto los propios modelos 3D de los complejos deportivos con los que se ha trabajado, como las marcas de posición que contienen la información de los comercios y servicios de la zona de estudio. Por otro lado, los archivos XML han servido de apoyo para el desarrollo de la API en la página Web, como se verá más adelante, en el apartado 4.5.4.

Así pues se ha considerado importante presentar este lenguaje de marcas, a partir del cual derivan otros de uso extendido. Es por eso que se exponen en este capítulo sus características, su funcionamiento interno y cómo son manejados por otras aplicaciones.

4.5.2.2. Lenguaje de marcas

Un lenguaje de marcas es una forma de codificar un documento de manera que, además de contener texto, incorpore una serie de etiquetas (marcas) que proporcionen información de su estructura y de cómo debe ser presentado.

4.5.2.3. Definición de XML

Se trata de un lenguaje de marcas extensible que dispone de un conjunto de reglas para definir etiquetas. Éstas se diseñan para amoldarse a las necesidades de los datos y documentos, pudiendo así estructurarlos, organizarlos e identificar diferentes partes de los mismos. A día de hoy, se considera como el formato universal *de facto* para el intercambio de documentos y datos estructurados a través de Internet.

4.5.2.4. Origen

Con el *boom* de la Web de 1997 se hicieron patentes las limitaciones del HTML en cuanto a interoperatividad y escalabilidad. Como medida temporal el W3C creó las hojas de estilo CSS (Apdo. 4.5.2.8.). Pero fue en 1998 cuando surgió XML como solución definitiva, combinando la potencia y capacidad de ampliación del SGML (lenguaje de marcado generalizado. Una especificación superior, adaptada para el almacenamiento de datos, que definía cómo debían hacerse los lenguajes de marcas).

4.5.2.5. Características

La principal característica de XML es la separación entre contenido y presentación. No tiene un marcado relativo a la presentación de los datos. Es el programa que lo maneja el que debe generar su presentación aplicando hojas de estilo CSS o XSL (Apdo. 4.5.2.8.). El propio documento describe los datos contenidos en él, su semántica y su información estructural, permitiendo que la aplicación correspondiente lo procese y manipule. Es capaz de crear un nuevo lenguaje de marcas definiendo su sintaxis y la forma en la que debe estructurar los documentos. Está escrito en un formato no propietario de texto sencillo, por lo que puede crearse un documento XML en cualquier editor de texto. Esto lo hace portátil, de fácil depuración, independiente de cualquier plataforma y resistente a la corrupción. Esto último es debido a que la pérdida de algunos bytes no corrompe la totalidad del documento, siendo por tanto ideal para el intercambio de datos.

4.5.2.6. Contenidos

En un documento XML se pueden distinguir los siguientes contenidos.

Elemento
Se trata de texto o caracteres, delimitados por una etiqueta de inicio "`<etiqueta>`" y otra de fin "`</etiqueta>`". Por ejemplo:

```
<cantidad>65</cantidad>
<title></title>
```

Atributo
Se trata de información que asocia características o propiedades a un elemento de un documento. Incluido dentro de éste, está formado por un nombre seguido de un signo "=", y un valor de atributo entre comillas. Por ejemplo:

```
<verdura clase="zanahoria" longitud='15" y media'>
<img href="http://www.....imagen.jpg">
```

Instrucción de proceso
Es una instrucción especial dirigida a una aplicación concreta. El *parser* (Apdo. 4.5.2.8.) pasa la instrucción a dicha aplicación y ésta la ejecuta. Aparece delimitada por los caracteres "<?" y "?>" y contiene un objetivo seguido de una serie de datos. El objetivo identifica la aplicación a la que va dirigida y los datos son las instrucciones que la aplicación debe ejecutar. Por ejemplo:

```
<?xml version="1.0"?>
<?xml-stylesheet href="classic.xsl" type="text/xml"?>
<?xml-stylesheet href="funky.xsl" type="text/xml" alternate="yes"?>
```

Comentario
Se trata de texto dirigido generalmente al programador, a modo informativo. Está delimitado por los caracteres "<!--" y "-->". Por ejemplo:

```
<!-- comentario dirigido al programador -->
<!-- los datos de los usuarios se actualizan cada mes -->
```

Entidad
Se utiliza para poder escribir caracteres especiales (©, <, >, &, ") sin que el *parser* los interprete como etiquetas. Se designan mediante el signo "&" seguido del nombre de la entidad y un ";". Por ejemplo:

```
&copy; para poder escribir ©.

<title>Visita virtual a la Playa Puebla de Farnals-&copy; AFO2010</title>
```

Siendo el resultado este:

Figura 4.187. Uso de la entidad `&copy;` para representar el símbolo ©

También es posible definir nuevas entidades, a las que se asocia una cadena de caracteres. Por ejemplo:

```
<!Entity nom "Alberto Villa">
```

De esta manera, cada vez que se escriba en el documento "`&nom;`", aparecerá el texto "Alberto Villa".

4.5.2.7. Documentos bien formados y documentos válidos

Un documento bien formado es aquel que cumple todas las definiciones básicas de la norma. Esto permite que un *parser* que la cumpla pueda analizarlo. Dicha norma establece que:

- Los documentos han de seguir una estructura estrictamente jerárquica. Sus etiquetas deben estar correctamente anidadas y sus elementos correctamente cerrados.
- Sólo se permite un elemento raíz.
- Los atributos deben aparecer entre comillas, simples o dobles.
- Se hace diferencia entre mayúsculas y minúsculas en las etiquetas.
- Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc.
- Las etiquetas, referencias de entidad y declaraciones, denominadas marcas, son la parte del documento que el *parser* XML espera entender. El resto son datos dirigidos al usuario.

Un documento es válido cuando, además de estar bien formado, sigue las normas de una Definición de Tipo de Documento (DTD) o un *Schema* XML (Apdo. 4.5.2.8.)

4.5.2.8. Tecnologías asociadas

XML Lleva asociado una serie de tecnologías que han ido enriqueciendo su desarrollo conforme se comprobaban sus posibilidades:

XLink

Es un lenguaje empleado para crear enlaces en documentos XML. Describe las relaciones cruzadas entre documentos, imágenes y otros archivos permitiendo:

- Crear vínculos entre varios documentos.
- Añadir a un vínculo información sobre sí mismo.
- Crear y describir vínculos a documentos almacenados en diversos sitios.

DTD (Definición de Tipo de Documento)

Define la estructura de un documento y qué elementos, atributos, entidades y notaciones pueden usarse. También define las relaciones que existen entre ellos, el significado de las marcas, cómo deben ser usadas y las restricciones a aplicar en la estructura del documento. Esto permite diseñar modelos de datos jerárquicos en forma de árbol asimétrico, más versátil que la forma tabular. De este modo el contenido es descrito con más precisión y se evitan repeticiones.

DTD sigue una normativa rígida, definida por XML, permitiendo que el documento pueda ser perfectamente interpretado por cualquier herramienta.

Cabe mencionar que una misma DTD puede aplicarse a varios documentos. De esta manera, cualquier modificación que se haga en ella se propaga automáticamente a todos los documento asociados.

Parser XML (Intérprete XML)

Se trata de una aplicación que suele funcionar como complemento o *plugin* bajo otras y que es capaz de analizar la estructura y sintaxis de un documento XML, verificar que esté bien formado y permitir el acceso a su contenido. Si además es "validante", puede comprobar si el documento es válido conforme a una Definición de Tipo de Documento (DTD).

Por otra parte, puede dar soporte a los DOM (Apdo. 4.5.2.8.), permitiendo así que los lenguajes de programación utilizados (C++, *JavaScript*, etc.) accedan y manipulen el contenido, la estructura y el estilo de documentos XML.

DOM (Modelo de Objetos de Documento)

Se trata de una Interfaz de Programación de Aplicaciones (API) diseñada para funcionar con cualquier lenguaje de programación, que permite crear, añadir, modificar o borrar elementos y contenido de documentos HTML y XML [125].

El DOM define la estructura lógica del documento mediante un modelo de árbol jerárquico, donde cada nodo es un objeto, que contiene elementos, atributos, contenidos, etc. Establece el comportamiento de los objetos, sus atributos, las relaciones entre ellos y la forma de manipularlos. Por ejemplo, a partir del siguiente código HTML:

```
<TABLE>
<TBODY>
<TR>
<TD>El grito</TD>
<TD>Edvard Munch</TD>
</TR>
<TR>
<TD>La costurera</TD>
<TD>Velazquez</TD>
</TR>
</TBODY>
</TABLE>
```

el DOM genera el siguiente modelo:

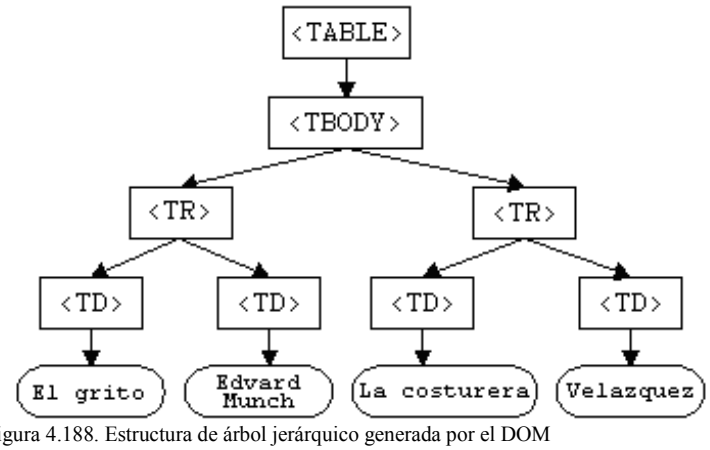


Figura 4.188. Estructura de árbol jerárquico generada por el DOM

CSS (Hoja de Estilo en Cascada)

Es un lenguaje de hojas de estilo que indica al navegador Web los estilos que debe usar para presentar los diferentes componentes del documento, como cabeceras, listas, párrafos, etc. La hoja de estilo aparece separada del texto marcado. Puede estar contenida en una sección especial, dentro del propio documento (como ocurre con HTML) o en un archivo aparte (HTML y XML). Esta separación entre contenido y presentación mejora el acceso a los datos y la definición de su aspecto final.

Igual que ocurre con las DTD, se puede aplicar una misma hoja de estilo a varios documentos, haciéndose más sencilla la estructura de éstos y evitándose repeticiones. Por otro lado, puede ocurrir que se apliquen varios estilos a un mismo elemento. En ese caso CSS establece un esquema de prioridades para determinar qué reglas de estilo se aplican y en qué orden.

XSL (Lenguaje de Hojas de Estilo Extensible)

Es un lenguaje de hojas de estilo, más potente que CSS, diseñado específicamente para documentos XML. Separa el contenido de un documento de su apariencia, mejorando de esta manera ambos.

Un mismo documento puede llevar asociados diferentes estilos, siendo el usuario el que elije su presentación e incluso selecciona qué datos quiere ver, su orden de salida, cuál quiere eliminar, etc.

XSL se divide en dos partes:

XSL-T (Transformaciones)
Transforma un documento desde un vocabulario XML a otro mediante una serie de reglas y marcas. Su uso es habitual en conversiones a HTML, texto plano, TeX entre otros. Permite reorganizar elementos, ocultar unos y mostrar otros, y puede aplicar estilos basándose en numerosos criterios, como el nombre de un elemento, sus atributos, su posición respecto a otros elementos, etc.

XSL-FO (Formato de los Objetos)
Se trata de un vocabulario XML que da formato al documento transformado. Puede tratar elementos sueltos o conjuntos, dando formato a uno o varios a la vez y pudiendo establecerse criterios de ordenación, filtrado, etc. Permite describir bloques, columnas, control de viudas, listas, etc., así como la maquetación y paginación del documento.

XPath (Lenguaje de rutas)

Se trata de un lenguaje que modela un documento XML como una estructura de árbol jerárquico y crea rutas que permiten localizar, identificar y seleccionar todos y cada uno de sus elementos, atributos o trozos de contenido.

Creado para su uso en el estándar XSL-T, se encarga de seleccionar y examinar la estructura del documento de entrada de la transformación.

XML Schema

Es un lenguaje que proporciona mayor expresividad que las DTD y mayor capacidad para definir las reglas de estructura y contenido que deben validar un documento XML. Por otra parte, permite especificar cómo trabajan juntos elementos y atributos, y las restricciones que se aplican a sus contenidos.

Hoy en día hay disponibles de forma gratuita en la red *Schemas* ya diseñados para su uso en diversos sectores (transacciones financieras, prescripciones médicas, artículos científicos, etc.).

Namespace (Nombres de Espacio)

Es un contexto que se define en un documento XML para evitar ciertas confusiones. Éstas suelen producirse cuando varias entidades hacen referencia a un mismo concepto. Por ejemplo, la etiqueta `<SALIDA>` puede usarse para describir el punto de partida de una carrera ciclista, la hora de salida de un vuelo, o la puerta de salida en un edificio.

Como solución se definen varios *namespaces* (`<xmlns>`), que aparecerán como prefijos en las etiquetas del documento, pudiendo distinguirse así aquellas que tengan nombres iguales pero un significado distinto. Por ejemplo:

```
<DOCUMENTO     xmlns:CICLISMO="http://www.teces.net/CICLISMO/"
               xmlns:VUELO="http://www.teces.net/VUELO/"
               xmlns:EDIFICIO="http://www.teces.net/EDIFICIO/">

       <CICLISMO:SALIDA>Pº Castellana 100</CICLISMO:SALIDA>
       <VUELO:SALIDA>18:00 Horas</VUELO:SALIDA>
       <EDIFICIO:SALIDA>Puerta Principal</EDIFICIO:SALIDA>
</DOCUMENTO>
```

4.5.2.9. Aplicaciones

Como se comenta al principio del apartado, XML se utiliza (entre otras cosas) para crear nuevos lenguajes de marcas. Estos se denominan aplicaciones, y mediante la sintaxis XML se define cómo deben estar escritos y cómo deben interpretarse sus peculiaridades, sin imprecisiones ni confusiones.

Hoy en día existen numerosas aplicaciones, con sus semánticas y vocabularios propios, entre las que destacan:

VRML (Lenguaje de Modelado de Realidad Virtual)
Formato normalizado para la representación de gráficos tridimensionales interactivos en la Web.

KML (Lenguaje de Marcado de Keyhole)
Utilizado para representar datos geográficos en tres dimensiones.

RSS (Sindicación Realmente Sencilla)
Usado para la sindicación a contenidos de páginas Web.

MathML (Lenguaje de Marcado Matemático)
Es una aplicación XML para representar ecuaciones matemáticas, desde aritmética escolar a cálculo y ecuaciones diferenciales.

CML (Lenguaje de Marcado Químico)
Permite la representación de estructuras y secuencias moleculares, análisis espectrográfico, cristalografía, publicación científica, bases de datos químicas, etc.

HL7 (Nivel de Salud 7)
Estándar internacional que permite la transferencia de información sobre pacientes, análisis, etc. entre sistemas informáticos médicos.

4.5.2.10. Navegadores

En la actualidad la mayoría de los navegadores Web tienen soporte para la presentación de documentos XML, entre los que se encuentran:

- *Mozilla Firefox* 3.5
- *Microsoft Explorer* 8
- *Opera* 9.64
- *Apple Safari* 4

Éstos incorporan un *parser* encargado de analizar los documentos. Si encuentra un error, lo indica, convirtiendo así a los navegadores en herramientas para desarrollo XML.

4.5.2.11. Editores

Como se comentó anteriormente, los documentos XML están escritos en texto sencillo y pueden crearse con cualquier editor de texto. Los más sencillos (*Bloc de notas*, *Vi*, etc.) no incorporan intérprete XML; los más avanzados sí, mostrando el documento mediante una estructura de árbol y permitiendo su análisis y depuración.

Los principales editores de XML en la actualidad son, entre otros:

- Altova *XML-Spy*
- Oxygen *XML Editor*
- Wattle Software *XML-Writer*
- *Notepad++*
- *Visual XML*

4.5.3. KML

4.5.3.1. Introducción

En este apartado se trata el lenguaje de marcas KML. En este proyecto los archivos kml y kmz, escritos en este lenguaje, han sido los más importantes. En ellos se han especificado la posición geográfica de las entidades que se iban a representar y la configuración de las vistas que cada una debía mostrar. Han contenido los modelos 3D creados en *SketchUp* de los diversos complejos deportivos que conforman el área de estudio, así como las marcas de posición que contenían todos los datos detallados, fotografías, enlaces, etc. de los diferentes comercios de la zona. Dada su versatilidad ha sido posible clasificar y almacenar diversos comercios de un mismo sector en su correspondiente carpeta, cosa que ha facilitado el trabajo en el posterior desarrollo de la página Web.

A continuación se describirán, dentro del amplio número de etiquetas con las que KML permite trabajar, aquéllas que han sido utilizadas para la elaboración de las marcas de posición del proyecto y algunas otras afines a éstas.

Por último se explicará la estructura interna del archivo kmz, gracias a la cual han podido almacenarse los modelos 3D creados en *SketchUp*, para su posterior representación en otras aplicaciones, como es el caso de *Google Earth*.

4.5.3.2. Historia

KML es un lenguaje de marcas creado originalmente por la compañía *Keyhole* en 2001 para la aplicación del mismo nombre, y que posteriormente pasaría a llamarse *Google Earth*.

El Open Geospatial Consortium (OGC) recibió la especificación KML 2.2 para asegurar su estatus como estándar abierto para todos los navegadores geográficos. A finales de 2007 el OGC formó un grupo de trabajo para formalizarlo y el 14 de Abril de 2008 pasó a ser estándar oficial del OGC con el nombre oficial de OpenGIS® KML Encoding Standard [149].

Con la formalización del KML como estándar surgieron diversas críticas [129].

Algunos expertos del sector, como (Jason Birch) pensaban que "*la innovación del sector se estancará con el nuevo estándar*". Otros, como Andrés Ferrate, de Cartosoft, sospechaban que Google "*quería imponer su voluntad en los estándares geoespaciales con un formato que originalmente no fue diseñado para su uso masivo*".

Desde la OGC, Tim Case argumentó que "*KML supone una suma y no una sustitución a los estándares geoespaciales*".

Por otro lado, parte de los desarrolladores alegaba que "*existen otros lenguajes de marcas geoespaciales mejores, más robustos y con más características que KML, como para adoptarlo como estándar*" (Andrés Ferrate - Cartosoft).

Existían numerosas quejas porque "*consumía muchos recursos, y el manejo de una vista con muchos KML necesitaba mucha potencia*", (Archie Belaney).

Pero finalmente, dada su amplia aceptación y uso, los miles de kml que circulan hoy en día por la red, la facilidad de trabajo con él y los numerosos "globos terráqueos virtuales" así como aplicaciones ajenas a Google (Microsoft, ESRI, NASA *World Wind* etc.) que lo utilizan, se puede afirmar que su estandarización ha valido la pena. Además esto está permitiendo que surjan competidores a *Google Earth* que incentivan la innovación y el desarrollo.

Cabe destacar finalmente que *Google Earth* fue el primer programa capaz de mostrar y editar archivos kml. No obstante, otros proyectos, como *Marble*, empezaron a desarrollar soporte para kml. *Google Maps*, por su parte, puede leer en estos momentos archivos kml o kmz.

4.5.3.3. Características

Se trata de una gramática XML diseñada para señalar lugares de la Tierra [149]. Un archivo KML contiene información relativa a la localización geográfica de diferentes entidades, como marcas de posición, imágenes, polígonos, modelos 3D, descripciones textuales, etc., con la idea de ser mostradas originalmente en *Google Earth* y hoy en día en cualquier visor 3D que tenga soporte para esta codificación.

KML está escrito en texto sencillo, por tanto, cualquier editor de textos puede servir para crear un nuevo archivo. No obstante, es recomendable usar un editor más sofisticado que permita comprobar la validez de su sintaxis y su estructura.

Sistemas de referencia

Para situar geográficamente los diversos elementos con los que se puede trabajar, KML utiliza un sistema de referencia con coordenadas geográficas 3D (longitud, latitud y altitud). Éste se divide en dos partes:

- **WGS84 con el orden de los ejes invertido**
  Para las coordenadas planimétricas se utiliza una adaptación del WGS84, con un sistema de coordenadas elipsoidal al que se ha invertido el orden de los ejes, pasando a expresarse como longitud y latitud, y con sus unidades en grados. El Datum utilizado es el World Geodetic System 1984, con elipsoide WGS 84 y meridiano central en Greenwich.

- **Geoide EGM96**
  Las altitudes se miden a partir de una "superficie de altura resultante de la aplicación del modelo de geoide EGM96 sobre el elipsoide WGS84", según el EPSG[1]. Se utiliza para ello un sistema coordenado vertical en el que se expresa la altura ortométrica (H), influenciada por el campo de gravedad terrestre, en metros. El Datum utilizado es el del geoide EGM96.

[1] EPSG: Grupo Europeo de Sondeos de Petróleo, formado por especialistas en geodesia, topografía y cartografía, con una amplia base de datos que contiene elipsoides, datums, sistemas de coordenadas, proyecciones cartográficas, etc. [148]

Usos

Los usos principales de KML son los siguientes:

- Señalar la posición geográfica de diferentes lugares del mundo mediante iconos y etiquetas.
- Preparar las vistas que muestran dichos lugares, configurando para ello la posición de la cámara virtual, a través de la cual se ven.
- Presentar imágenes superpuestas a la pantalla o adaptadas a la orografía del terreno.
- Definir la apariencia de las entidades mediante el uso de estilos.
- Aportar información acerca de los lugares señalados mediante el uso de texto sencillo y/o código HTML, incluyendo hipervínculos e imágenes.
- Agrupar entidades mediante el uso de carpetas.

Reglas sintácticas

De igual manera que ocurre con otras sintaxis XML, se basa en el uso de etiquetas, con nombres, atributos y contenido que definen al completo la posición geográfica del lugar señalado, así como las características de la vista que debe mostrarse, obedeciendo para ello las reglas de aquél:

- Todos los elementos deben tener una etiqueta de inicio y de cierre.
- Se diferencian mayúsculas de minúsculas.
- Todos los elementos deben estar correctamente anidados y ser bien-formados.
- El documento debe tener siempre un único elemento raíz.
- Los valores de atributo deben ir siempre entre comillas.

En caso de no seguirse estas reglas, *Google Earth* mostraría un aviso diciendo que se ha producido un error de formato.

Marca de posición

Es la principal forma de señalar un lugar sobre la Tierra. Suele estar formada por un punto al que va asociado un icono, aunque también puede ser una línea, un polígono, una imagen superpuesta o una figura geométrica 3D. Para establecer su configuración se definen los siguientes atributos y características:

- **Formas geométricas**
  La marca de posición puede contener simplemente un punto, una línea o un polígono o ser más compleja y contener una combinación de ellos.

- **Posición y altitud**
  Cada una de las entidades geométricas contenidas en una marca de posición tiene sus coordenadas, altitud, extrusión y relación con el suelo propias.

- **Icono**
  Puede definirse un icono que identifique un tipo de marcas de posición, estableciendo la ubicación de la imagen que quiera usarse como icono.

- **Apariencia**
  Mediante el uso de estilos se define la apariencia de todas las entidades que conforman la marca de posición, desde el aspecto de su icono y etiqueta hasta la posición y configuración de la cámara virtual.

Agrupación de entidades

Permiten la organización de la información contenida en un documento KML cuando ésta se hace más compleja, dado que es bastante normal que un documento contenga numerosas marcas de posición y éstas contengan a su vez diversos tipos de figuras geométricas. KML dispone de los siguientes tipos de agrupación:

- **Documentos**
  El elemento `<Document>` es único en un documento KML, está situado en el nivel raíz de éste y contiene al resto de elementos, además de otras entidades como carpetas, marcas de posición, imágenes para superposición, etc.

- **Carpetas `<Folder>`**
  Normalmente se emplean carpetas para estructurar grupos de entidades y capas del mismo tipo, y también para presentar una vista unificada de un grupo concreto de marcas de posición o capas.

- **Colecciones de geometría `<MultiGeometry>`**
  Es recomendable su uso cuando se está dibujando un modelo complejo. Se utiliza para hacer grupos de entidades pertenecientes a diferentes partes de un mismo modelo, permitiendo mostrarlas u ocultarlas activando o desactivando cada grupo.

4.5.3.4. Etiquetas empleadas

Hasta el momento se han podido crear marcas de posición, rutas, polígonos, etc. mediante la interfaz de *Google Earth*, pero ésta sólo permite utilizar una pequeña parte de las etiquetas y parámetros del código KML existentes [88].

A continuación se describen las etiquetas que se han generado a través de *Google Earth*, aquéllas que se han añadido mediante editor de textos, y todos los parámetros utilizados para la confección de las marcas de posición de este proyecto.

Como nota, cabe observar que algunos elementos específicos de las siguientes etiquetas aparecen sin descripción. Esto es debido a que van a ser descritos únicamente la primera vez que aparezcan, evitándose así la repetición continuada.

`<Placemark>`

Indica la posición geográfica de un elemento geométrico (punto, línea, polígono, etc.). En caso de tratarse de un punto, se muestra en el visor mediante un icono.

`<name>`

Texto asociado a diversas entidades, como marca de posición, imagen superpuesta, carpeta, enlace de red, etc. y que se muestra como etiqueta en el visor.

`<description>`

Contiene información detallada sobre el lugar indicado por la marca de posición, y aparece en una ventana de información cuando el usuario pulsa en el icono de dicha marca. Puede estar escrita en texto sencillo y HTML. Ejemplo:

```
<description><![CDATA[<FONT FACE="verdana" SIZE=1>
        <center><IMG SRC="http:[..]/83t.jpg" width=150></center>
        <br>Paseo de Colón 4-8
        <br>Puerto Deportivo Pobla Marina
        <br>46137 - Puebla de Farnals
        <br>Valencia<br>
        <br>Telf: 96 146 16 31<br><br>
        <center><a href="http:[…]"target="_blank">www.[…].com</a></center>
        <br>
        <center><img src="http:[…]/83f.jpg" width=200></center>]]>
</description>
```

Figura 4.189. Ejemplo de ventana de información

<Folder>

Se utiliza para la organización de carpetas, marcas de posición, superposición de imágenes y otras entidades. Por ejemplo:

```
<Folder>
	<name>Nombre de la carpeta </name>
	<description>Texto descriptivo</description>
	<Folder>
		<name>Nombre de SubCarpeta nº1</name>
		<description>Texto descriptivo</description>
		<Placemark> [...] </Placemark>
	</Folder>
	<Folder>
		<name> Nombre de SubCarpeta nº2<name>
		<description>Texto descriptivo</description>
		<Placemark> [...] </Placemark>
	</Folder>
</Folder>
```

Elementos geométricos

Las siguientes etiquetas permiten definir los elementos geométricos (puntos, líneas, polígonos, etc.) que forman parte de una marca de posición.

<Point>

Contiene los datos geográficos de la marca de posición. En dicha posición aparecerán el nombre de la marca y su icono asociado.

Elementos específicos de los puntos

`<extrude>`
Especifica si el punto debe conectar con el suelo mediante una línea.

`<altitudeMode>`
Indica el modo de altitud para el elemento `<coordinates>`, pudiendo ser:

- `clampToGround`
  Se ignora la altitud y el elemento se sitúa a ras de suelo.
- `relativeToGround`
  La altitud se mide respecto a la elevación del suelo.
- `Absolute`
  La altitud se mide respecto al nivel del mar.

`<gx:altitudeMode>`
Indica las altitudes relativas al fondo del mar, pudiendo ser:

- `relativeToSeaFloor`
  La altitud se mide respecto del fondo del mar.
- `clampToSeaFloor`
  Se ignora la altitud y el elemento se sitúa en el fondo del mar.

`<coordinates>`
Contiene la localización geográfica de la marca de posición. La longitud y latitud se expresan en grados y la altitud en metros. Ejemplo:

```
<Point>
        <extrude>1</extrude>
        <altitudeMode>clampToGround</altitudeMode>
        <gx:altitudeMode>clampToSeaFloor</gx:altitudeMode>
        <coordinates>-0.28364722,39.566689,15</coordinates>
</Point>
```

`<Icon>`

Establece qué imagen va a ser utilizada como icono o para una superposición.

Elementos específicos de los iconos

`<href>`
Define la URL donde está almacenada la imagen que se va a usar. Ésta puede estar en un sistema de archivos local o en un servidor Web remoto. Ejemplo:

```
<Icon>
        <href>http://maps.google.com/[…]/pushpin/ylw-pushpin.png</href>
</Icon>
```

`<LineString>`

Define un conjunto de líneas conectadas entre sí.

Elementos específicos de las cadenas de líneas

`<extrude>`

`<tessellate>`
Establece si la cadena de líneas debe seguir la orografía del terreno. Para ello el modo de altitud debe ser `clampToGround` o `clampToSeaFloor`.

`<altitudeMode>`

`<gx:altitudeMode>`

`<coordinates>`
Contiene las coordenadas geográficas de todos los puntos que conforman la cadena de líneas. Ejemplo:

```
<LineString id="ID">
        <extrude>0</extrude>
        <tessellate>0</tessellate>
        <altitudeMode>clampToGround</altitudeMode>
        <coordinates>  -0.2875913002803410,39.56004153620055,28
                       -0.2869358350924545,39.55987126893012,28
                       -0.2830699864966535,39.56680803451898,28
                       -0.2876208805238756,39.56776833242124,28
        </coordinates>
</LineString>
```

<Polygon>

Define un polígono mediante segmentos lineales. Éstos se utilizan para definir el límite exterior del polígono así como uno o varios límites interiores, si es que tiene.

Elementos específicos de los polígonos

<extrude>

<tessellate>

<altitudeMode>

<gx:altitudeMode>

<outerBoundaryIs>
Contiene el límite exterior del polígono, definido por un <LinearRing>.

<innerBoundaryIs>
Contiene un límite interior del polígono, definido por un <LinearRing>.

<LinearRing>

Define una cadena cerrada de líneas.

Elementos específicos de las cadenas cerradas de líneas

<extrude>

<tessellate>

<altitudeMode>

<gx:altitudeMode>

<coordinates>
Contiene las coordenadas de los puntos que conforman la cadena cerrada de líneas, siendo la primera y la última la misma. Ejemplo:

```
<Polygon>
	<extrude>1</extrude>
	<tessellate>1</tessellate>
	<altitudeMode>relativeToGround</altitudeMode>
	<outerBoundaryIs>
		<LinearRing>
			<coordinates>
			-0.2875913002803410,39.56004153620055,28
			-0.2869358350924545,39.55987126893012,28
			-0.2830699864966535,39.56680803451898,28
			-0.2876208805238756,39.56776833242124,28
			-0.2875913002803410,39.56004153620055,28
			</coordinates>
		</LinearRing>
	</outerBoundaryIs>
</Polygon>
```

<GroundOverlay>

Define los parámetros para la superposición de una imagen sobre el terreno.

Elementos específicos de las superposiciones sobre el terreno

<Icon>
Define qué imagen va a usarse en la superposición.

<altitude>

<altitudeMode>

<gx:altitudeMode>

<LatLonBox>
Define la posición geográfica de los lados superior <north>, inferior <south>, derecho <east> e izquierdo <west> del cuadrado que contiene la imagen que se va a superponer al terreno.
También permite aplicar un giro <rotation> a la imagen.

<ScreenOverlay>

Define una superposición de imagen en la pantalla.

Elementos específicos de las superposiciones sobre la pantalla.

<overlayXY>
Define las coordenadas del punto en la imagen que se va a superponer que será situado en las coordenadas de la pantalla que se definan más adelante.

<screenXY>
Define las coordenadas del punto en la pantalla en las que se situará la imagen a superponer, teniendo como punto de anclaje el definido en <overlayXY>.

xunits, yunits.
Unidades en las que están expresados los valores *x* e *y*, que pueden ser:

- *Pixels* (pixels).
- Fracciones del icono (fraction).
- Inserción de *pixels* (insetPixels). Referenciados desde la esquina superior derecha del icono.

Ejemplo:

```
<ScreenOverlay>
       <overlayXY x="0.5" y="0.5" xunits="fraction" yunits="fraction"/>
       <screenXY x="0.5" y="0.5" xunits="fraction" yunits="fraction"/>
</ScreenOverlay>
```

Estilos

Definen la apariencia que deben tener los iconos, etiquetas, líneas y superficies de polígonos que configuran las marcas de posición.

<StyleMap>

El icono de una marca de posición presenta dos modos de presentación. Uno normal que corresponde a su estado por defecto, y otro resaltado, que es el que se activa cuando el usuario pasa el cursor por encima del icono. `StyleMap` establece los estilos que corresponden a cada modo.

Elementos específicos de las asignaciones de estilos

`<Pair>`
Establece una dupla formada por el modo de la marca de posición y la URL del estilo asociado a aquél:

- <key> Establece el modo de la marca de posición que se está definiendo.
- <styleUrl> Indica la URL donde se encuentra el estilo asociado.

Ejemplo:

```
<StyleMap id="msn_ylw-pushpin">
	<Pair>
		<key>normal</key>
		<styleUrl>#sn_ylw-pushpin</styleUrl>
	</Pair>
	<Pair>
		<key>highlight</key>
		<styleUrl>#sh_ylw-pushpin</styleUrl>
	</Pair>
</StyleMap>
```

<Style id="…">

Contiene un grupo de estilos para diferentes entidades (iconos, etiquetas, líneas, polígonos, etc.) al que se puede referenciar mediante su identificador. Ejemplo:

```
<Style id="sn_ylw-pushpin">
	<IconStyle>
		<scale>1.1</scale>
		<Icon>
			<href>http://maps.google.com/kml/ylw-
pushpin.png</href>
		</Icon>
		<hotSpot x="20" y="2" xunits="pixels" yunits="pixels"/>
	</IconStyle>
	<LineStyle>
		<color>cc0055ff</color>
		<width>4</width>
	</LineStyle>
	<PolyStyle>
		<color>99ffaa00</color>
	</PolyStyle>
</Style>
```

<IconStyle>

Especifica el estilo que debe aplicarse al icono de una marca de posición.

Elementos específicos del estilo de icono

<color>
Establece la opacidad y el matiz de color que se aplica al icono. Los valores se expresan en notación hexadecimal. El orden de la expresión es AABBGGRR, siendo AA la opacidad, BB el color azul, GG el color verde y RR el color rojo.

<scale>
Especifica la escala del icono en *x* e *y*.

<Icon>
Determina qué imagen va a utilizarse como icono.

<href>
Contiene la URL en la que está almacenada la imagen utilizada como icono.

<hotSpot x="20" y="2" xunits="pixels" yunits="pixels">
Indica qué punto del icono va a estar situado en las coordenadas especificadas en la marca de posición.
xunits, yunits

Ejemplo:

```
<IconStyle>
	<color>7f7fff00</color>
	<scale>1.77273</scale>
	<Icon>
	<href>http://maps.google.com/kml/pushpin/ylw-pushpin.png</href>
	</Icon>
	<hotSpot x="20" y="2" xunits="pixels" yunits="pixels"/>
</IconStyle>
```

<LabelStyle>

Establece el aspecto de la etiqueta asociada a una marca de posición.

Elementos específicos del estilo de etiqueta

`<scale>`
Establece el tamaño de la etiqueta.

`<color>`

`<colorMode>`
Establece el modo de color, que puede ser `normal` (sin efecto) y `random` (aleatorio).

Ejemplo:

```
<LabelStyle>
        <color>ff0000ff</color>
        <colorMode>normal</colorMode>
        <scale>2</scale>
</LabelStyle>
```

<StyleUrl>

Indica dónde está almacenado un estilo. Si se encuentra en un archivo remoto, se especifica su URL y se hace referencia al estilo que se quiere emplear mediante "`#`" y su identificador. Si está en el propio documento, basta con utilizar la referencia "`#`" seguida de su identificador. Ejemplos:

```
<styleUrl>#myIconStyleID</styleUrl>

<styleUrl>http://someserver.com/somestylefile.xml#restaurant</styleUrl>
```

<LineStyle>

Especifica el estilo que debe aplicarse a los elementos lineales, incluyendo a las líneas usadas como contornos en los polígonos y las líneas de extrusión.

Elementos específicos del estilo de línea

`<width>`
Establece la anchura de la línea, en *pixels*.

`<color>`

`<colorMode>`

Ejemplo:

```
<LineStyle>
        <color>ffffaa55</color>
        <colorMode>normal</colorMode>
        <width>2</width>
</LineStyle>
```

<PolyStyle>

Especifica el estilo de dibujo para los polígonos, incluidos los generados en la extrusión.

Elementos específicos de los estilos de polígonos

<color>

<colorMode>

<fill>
Especifica si se debe dibujar la superficie del polígono.

<outline>
Especifica si se debe dibujar el contorno del polígono.

Ejemplo:

```
<PolyStyle id="ID">
        <color>ffffffff</color>
        <colorMode>normal</colorMode>
        <fill>1</fill>
        <outline>1</outline>
</PolyStyle>
```

Configuración de la cámara

Establece la posición y configuración de la cámara para mostrar la escena.

<LookAt>

Define la posición de la cámara a partir de la posición del objeto al que está mirando.

Elementos específicos del punto de mira.

<longitude>
Longitud del punto al que la cámara está mirando.

<latitude>
Latitud del punto al que la cámara está mirando.

<altitude>
Altitud del punto al que la cámara está mirando.

<heading>
Orientación de la cámara respecto al norte.

<tilt>
Inclinación de la cámara respecto a la normal a la superficie de la Tierra.

<range>
Distancia entre la cámara y el punto al que se está mirando.

<altitudeMode>

<gx:altitudeMode>

Ejemplo:

```
<LookAt>
        <longitude>-0.2815388889</longitude>
        <latitude>39.56269167</latitude>
        <altitude>0</altitude>
        <range>75</range>
        <tilt>50</tilt>
        <heading>81.8</heading>
        <altitudeMode>relativeToGround</altitudeMode>
</LookAt>
```

<Camera>

Define la posición de la cámara respecto a la superficie de la Tierra, así como su orientación.

Elementos específicos de la cámara

`<longitude>`
Longitud a la que se encuentra la cámara.

`<latitude>`
Latitud a la que se encuentra la cámara.

`<altitude>`
Altitud a la que se encuentra la cámara, medida desde la superficie de la Tierra.

`<heading>`
Orientación de la cámara respecto al Norte.

`<tilt>`
Giro de la cámara alrededor de su eje X.

`<roll>`
Giro de la cámara alrededor de su eje Z.

`<altitudeMode>`

`<gx:altitudeMode>`

Ejemplo:

```
<Camera id="id">
        <longitude>-0.2859256628234608</longitude>
        <latitude>39.56403818786482</latitude>
        <altitude>500</altitude>
        <range>1000</range>
        <tilt>10</tilt>
        <roll>0</roll>
        <heading>-8.020188462337538</heading>
        <altitudeMode>relativeToGround</altitudeMode>
</Camera>
```

4.5.3.5. El archivo kmz

El formato por defecto de los archivos creados por *Google Earth* es KMZ. Consiste en un archivo KML comprimido que puede ser abierto por cualquier software que soporte el formato ZIP. Su principal ventaja es que pueden almacenarse las imágenes en el propio archivo sin necesidad de hacerlo en un servidor remoto.

Los contenidos de un archivo KMZ son un simple documento KML raíz (generalmente *doc.kml*) y opcionalmente cualquier tipo de capas, imágenes, iconos, y modelos a los que se hace referencia desde el KML incluyendo otros archivos KML enlazados a través de la red. Por convención el documento KML raíz se encuentra en el nivel raíz y los archivos referenciados se encuentran en subdirectorios (p.e. imágenes para superponer).

A continuación se puede apreciar la estructura del documento *02-orly.kmz*. A la izquierda en *Google Earth* y a la derecha una vez descomprimido:

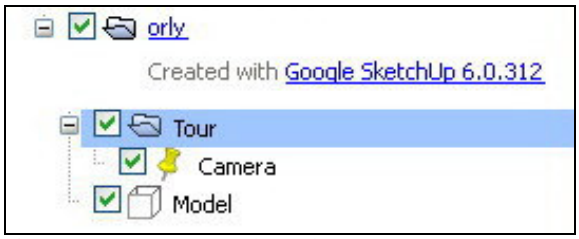


Figura 4.190. KMZ en *Google Earth*

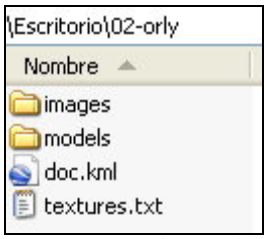


Figura 4.191. KMZ descomprimido

La subcarpeta "images" contiene las imágenes que se han empleado como textura en el modelo 3D:

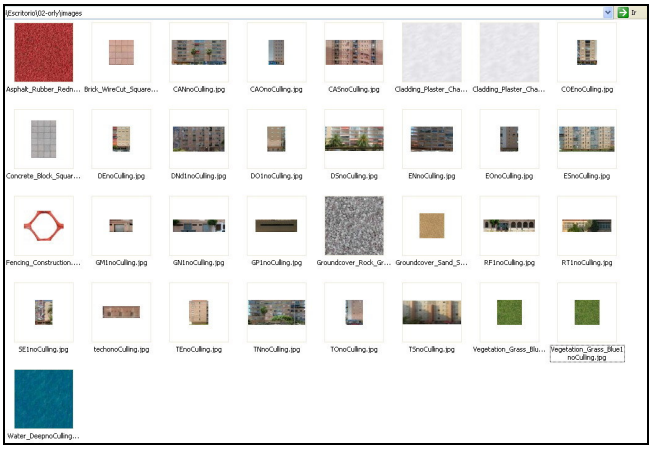


Figura 4.192. Texturas contenidas en la carpeta "images"

En la subcarpeta "models" se encuentra el archivo del modelo en formato COLLADA [54] (.dae):

Figura 4.193. Archivo del modelo 3D

El archivo *doc.kml* contiene el código KML que permite mostrar el icono, la etiqueta, el modelo 3D, la posición de cámara y demás ajustes del modelo:

```
<?xml version='1.0' encoding='UTF-8'?>
<kml xmlns='http://earth.google.com/kml/2.1'>
<Folder>
	<name>orly</name>
	<description>
		<![CDATA[Created with <a href="http://sketchup.google.com/">Google
		SketchUp 6.0.312</a>]]>
	</description>
	<DocumentSource>SketchUp</DocumentSource>
	<visibility>1</visibility>
	<LookAt>
		<heading>163.458</heading>
		<tilt>57.7158</tilt>
		<latitude>39.56985559615494</latitude>
		<longitude>-0.2855228251872715</longitude>
		<range>434.2539303135265</range>
		<altitude>259.933327269528</altitude>
	</LookAt>
	<Folder>
		<name>Tour</name>
		<Placemark>
			<name>Camera</name>
			<visibility>1</visibility>
			<LookAt>
				<heading>163.458</heading>
				<tilt>57.7158</tilt>
				<latitude>39.56985559615494</latitude>
				<longitude>-0.2855228251872715</longitude>
				<range>434.2539303135265</range>
				<altitude>259.933327269528</altitude>
			</LookAt>
		</Placemark>
	</Folder>
	<Placemark>
		<name>Model</name>
		<description><![CDATA[]]></description>
		<Style id='default'></Style>
		<Model>
			<altitudeMode>relativeToGround</altitudeMode>
			<Location>
				<longitude>-0.282739340889</longitude>
				<latitude>39.566838428448</latitude>
				<altitude>0.000000000000</altitude>
			</Location>
			<Orientation>
				<heading>0</heading>
				<tilt>0</tilt>
				<roll>0</roll>
			</Orientation>
			<Scale>
				<x>1.0</x>
				<y>1.0</y>
				<z>1.0</z>
			</Scale>
			<Link>
				<href>models/orly.dae</href>
			</Link>
		</Model>
	</Placemark>
</Folder>
</kml>
```

Puede observarse cómo el código KML llama al modelo 3D almacenado en la subcarpeta “models”, dentro del propio KMZ.

Y por último, el archivo *textures.txt*, que define la localización, dentro del archivo KMZ, de todas las texturas utilizadas en el modelo 3D.

```
<../images/Asphalt_Rubber_RednoCulling.jpg> <../images/Asphalt_Rubber_RednoCulling.jpg>
<../images/Brick_WireCut_Square1noCulling.jpg>
<../images/Brick_WireCut_Square1noCulling.jpg>
<../images/CANnoCulling.jpg> <../images/CANnoCulling.jpg>
<../images/CAOnoCulling.jpg> <../images/CAOnoCulling.jpg>
<../images/CASnoCulling.jpg> <../images/CASnoCulling.jpg>
<../images/Cladding_Plaster_Chalk1.jpg> <../images/Cladding_Plaster_Chalk1.jpg>
<../images/Cladding_Plaster_ChalknoCulling.jpg>
<../images/Cladding_Plaster_ChalknoCulling.jpg>
<../images/COEnoCulling.jpg> <../images/COEnoCulling.jpg>
<../images/Concrete_Block_Square_Gray2noCulling.jpg>
<../images/Concrete_Block_Square_Gray2noCulling.jpg>
<../images/DEnoCulling.jpg> <../images/DEnoCulling.jpg>
<../images/DNd1noCulling.jpg> <../images/DNd1noCulling.jpg>
<../images/DO1noCulling.jpg> <../images/DO1noCulling.jpg>
<../images/DSnoCulling.jpg> <../images/DSnoCulling.jpg>
<../images/ENnoCulling.jpg> <../images/ENnoCulling.jpg>
<../images/EOnoCulling.jpg> <../images/EOnoCulling.jpg>
<../images/ESnoCulling.jpg> <../images/ESnoCulling.jpg>
<../images/Fencing_Construction.png> <../images/Fencing_Construction.png>
<../images/GM1noCulling.jpg> <../images/GM1noCulling.jpg>
<../images/GN1noCulling.jpg> <../images/GN1noCulling.jpg>
<../images/GP1noCulling.jpg> <../images/GP1noCulling.jpg>
<../images/Groundcover_Rock_GraynoCulling.jpg>
<../images/Groundcover_Rock_GraynoCulling.jpg>
<../images/Groundcover_Sand_SmoothnoCulling.jpg>
<../images/Groundcover_Sand_SmoothnoCulling.jpg>
<../images/RF1noCulling.jpg> <../images/RF1noCulling.jpg>
<../images/RT1noCulling.jpg> <../images/RT1noCulling.jpg>
<../images/SE1noCulling.jpg> <../images/SE1noCulling.jpg>
<../images/techonoCulling.jpg> <../images/techonoCulling.jpg>
<../images/TEnoCulling.jpg> <../images/TEnoCulling.jpg>
<../images/TNnoCulling.jpg> <../images/TNnoCulling.jpg>
<../images/TOnoCulling.jpg> <../images/TOnoCulling.jpg>
<../images/TSnoCulling.jpg> <../images/TSnoCulling.jpg>
<../images/Vegetation_Grass_Blue1noCulling.jpg>
<../images/Vegetation_Grass_Blue1noCulling.jpg>
<../images/Vegetation_Grass_Blue1_1.jpg> <../images/Vegetation_Grass_Blue1_1.jpg>
<../images/Water_DeepnoCulling.jpg> <../images/Water_DeepnoCulling.jpg>
```

4.5.4. La API de Google Earth

4.5.4.1. Introducción

La API de *Google Earth*, ha sido otra de las partes importantes del proyecto. Gracias a ella ha sido posible que la página Web pudiera trabajar con *Google Earth*, como se verá en el apartado 4.5.5. De esta manera se ha podido representar el globo terráqueo, superponerle los modelos 3D construidos en *SketchUp*, añadir las marcas de posición de los comercios y los diferentes complejos deportivos, así como representar las ventanas de información correspondientes. Todo esto ha hecho que los resultados que se han obtenido con el uso de esta API para la solución de este proyecto hayan sido del todo satisfactorios.

En este capítulo se explican sus características principales y las diversas interfaces que ofrece para la elaboración de aplicaciones Web avanzadas. No obstante, si bien no se usó la gran mayoría de ellas, se explican al detalle aquellas que sí se utilizaron en este proyecto, aportando ejemplos de cada una.

4.5.4.2. API (Interfaz de Programación de Aplicaciones)

Durante la programación de una aplicación suele suceder que ésta no sea capaz de realizar ciertas funciones por sí sola. Existen no obstante otras aplicaciones que sí pueden realizar dichas funciones, pero que escapan a nuestro conocimiento. Para poder hacer uso de dichas funciones sin complicaciones están las API, que hacen de intermediario entre las aplicaciones, de manera que la aplicación en la que estamos programando puede hacer peticiones a la aplicación especializada a través de la API, siendo ésta la encargada de traducirle a la segunda aplicación lo que queremos obtener. La API también se encarga de devolvernos el resultado que estábamos buscando. Dichos resultados van desde dibujar ventanas o iconos a representar mapas en la pantalla. El uso de una API es una gran ventaja, ya que nos evita tener que programar todo desde cero.

Para ello la API establece un vocabulario y unas normas para realizar las peticiones a la aplicación especializada. Esto incluye especificaciones para las rutinas, estructuras de datos, clases de objetos y protocolos para la comunicación entre ambas. Cuando una aplicación hace uso de una determinada API se le denomina "implementación de dicha API".

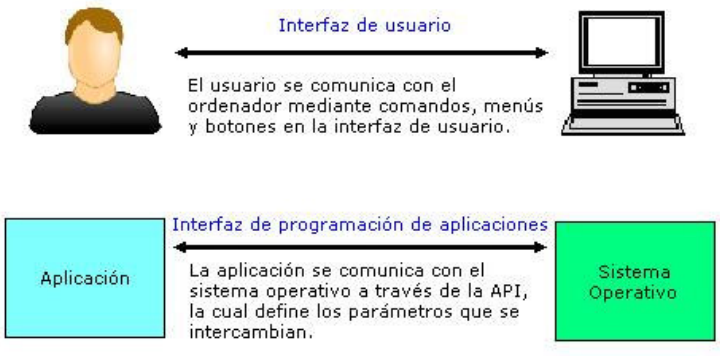


Figura 4.194. Interfaz de programación de aplicaciones

4.5.4.3. APIs en la Web

En el contexto de desarrollo de aplicaciones Web, una API suele ser la definición de un conjunto de mensajes de llamada a través de HTTP, así como de la estructura que van a tener los mensajes de respuesta. Normalmente éstos aparecen expresados en formato XML o JSON (*JavaScript* para Notación de Objetos).

La proliferación de múltiples APIs hoy en día ha permitido la aparición de mashups, i.e. nuevas aplicaciones Web que combinan contenidos de más de una fuente de información [107]. Utilizan para ello sus respectivas APIs, siendo las más importantes eBay, Amazon, Google, Yahoo!, Wikipedia, DBPedia, Youtube, BBC, National Geographic, The History Channel, Flickr y muchas otras.

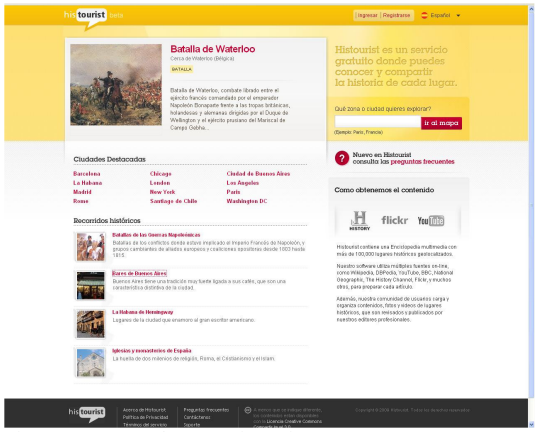


Figura 4.195. Mashup en Histourist.com

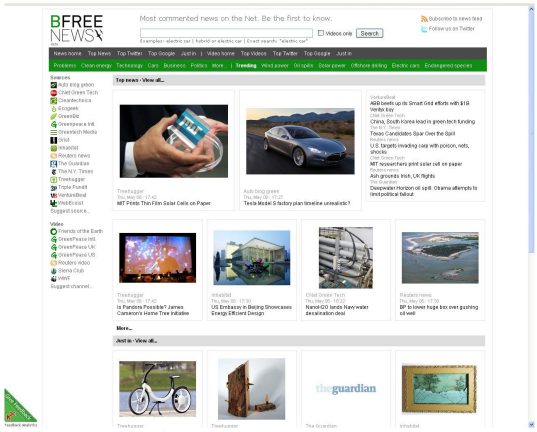


Figura 4.196. Mashup en BFreeNews.com

4.5.4.4. API de Google Earth

Esta API permite mostrar una versión de *Google Earth* en una página Web [86]. No incorpora todas las características de que dispone la aplicación original, pero con ella es posible programar aplicaciones cartográficas 3D avanzadas [80].

A día de hoy está disponible gratuitamente como complemento o *plugin* para los siguientes navegadores Web y sistemas operativos [92]:

- Microsoft Windows (2000, XP, Vista y Windows 7):
  - Google *Chrome* 1.0
  - *Internet Explorer* 6.0
  - Mozilla *Firefox* 2.0
  - *Flock* 1.0

- Apple Mac OS X 10.4 y superiores (Intel y PowerPC)
  - *Safari* 3.1
  - *Firefox* 3.0

El archivo de instalación puede descargarse desde el siguiente enlace:

http://dl.google.com/earth/plugin/GoogleEarthPluginSetup_en.exe

Igual que ocurría en el apartado 4.4.2.3. con *Google Maps*, también es necesario generar una clave para la API de *Google Earth* a través de la siguiente página Web:

http://code.google.com/intl/es-ES/apis/maps/signup.html

Esta clave debe incorporarse al código *script* de la página que se vaya a programar para que la API funcione.

Por otro lado, para una programación de la aplicación más cómoda es recomendable utilizar una herramienta de depuración de *JavaScript*. Ésta ayudará a localizar los problemas y a resolverlos según vayan apareciendo. Suelen presentarse como complementos a los propios navegadores Web, siendo las más importantes:

- *Firebug*: Complemento del navegador *Firefox*
- *Firebug Lite*: *Script* para colocar la consola de *Firebug* en una página Web
- Microsoft *Script Editor*: Complemento de *Internet Explorer*
- Microsoft *Script Debugger*: Complemento de *Internet Explorer*
- *Venkman*: Complemento del *Firefox*

Otros navegadores, como *Safari* de Apple, Google *Chrome* y Microsoft *Internet Explorer 8* ya llevan incorporados depuradores de *JavaScript* y otras herramientas de desarrollo.

4.5.4.5. Características

Igual que sucede en la aplicación original, el complemento de *Google Earth*, a través de su API en *JavaScript*, permite la creación entidades. Éstas, que ya fueron tratadas en el apartado 4.4.3., dedicado a *Google Earth*, son:

- Marcas de posición
- Ventanas de información
- Cadenas de líneas
- Polígonos, exteriores e interiores
- Superposiciones sobre el terreno
- Superposiciones sobre la pantalla
- Multigeometrías

Por otro lado permite importar y mostrar en la pantalla:

- Modelos 3D en formato COLLADA, directamente o en archivos kmz
- Archivos kml y kmz. Soporta KML 2.2 y el espacio de nombres "`gx`"
- Rutas en formato KML que pueden ser reproducidas mientras el usuario interactúa con el entorno

Acepta cadenas de código KML directamente dentro del propio *script*.

Es posible mostrar controles de navegación en la pantalla, situándolos donde mejor convenga.

Tiene acceso a varias de las capas de la aplicación original, que son:

- Fronteras y normes de países, provincias y ciudades
- Carreteras
- Relieve
- Edificios 3D fotorrealistas
- Edificios 3D en escala de grises

Puede mostrar en la ventana la siguiente información de la vista actual:

- Escala actual del mapa
- Coordenadas geográficas y altitud del punto marcado
- Meridianos y paralelos
- Mapa de referencia a pequeña escala

Es posible mostrar el Sol y la atmósfera terrestre.

Permite observar el firmamento mediante la activación del modo Cielo.

Y por último, también es capaz de mostrar, de igual manera que hace con la Tierra, el relieve y las imágenes de la superficie de Marte y la Luna.

4.5.4.6. Programación de la API

Durante la programación de la aplicación Web se ha empleado el espacio de nombre `google.earth`, que contiene las funciones que dan acceso a las diferentes interfaces de la API [97]. Por ejemplo, la generación de los objetos se realiza mediante La función `google.earth.createInstance`. Así pues, para generar el objeto que contiene el modelo 3D del globo terráqueo se escribiría:

```
google.earth.createInstance("map3d", initCallback, failureCallback);
```

Las siguientes interfaces permiten programar el acceso a las funciones internas del complemento y a otras opciones [90].

```
GEAbstractBalloon    GEGlobe             GELinearRingContainer GESchemaObjectContainer
GEEventEmitter       GEHitTestResult     GENavigationControl   GESun
GEFeatureBalloon     GEHtmlBalloon       GEOptions             GETourPlayer
GEFeatureContainer   GEHtmlDivBalloon    GEPhotoOverlayViewer  GEView
GEGeometryContainer  GEHtmlStringBalloon GEPlugin              GEWindow
```

Cabe destacar que, en la elaboración de este proyecto, como ya se vio anteriormente, las marcas de posición, las ventanas de información, los modelos 3D y demás entidades que conforman su totalidad fueron editadas con las aplicaciones *Google Earth* y *Google SketchUp*. Es por eso que esta API no ha sido utilizada para la creación de nuevas entidades, sino principalmente para mostrar la Tierra y colocar sobre ella las entidades y modelos 3D importados. No obstante, la API ha sido utilizada para la inclusión de controles de navegación, para la superposición de varios logotipos, control de la cámara y la programación de una serie de rutas.

Es por eso que, pese a disponer de todas las interfaces arriba expuestas, sólo se van a detallar a continuación aquellas que han sido empleadas en la elaboración de la página Web desarrollada para este proyecto.

Interfaz `GEWindow`

Establece el comportamiento del objeto contenido en la ventana en la que se muestra *Google Earth*.

Función

```
GEWindow.setVisibility(true/false)
```

Define la visibilidad del objeto contenido en la ventana (p.ej. el globo terráqueo).

Ejemplo:

```
ge.getWindow().setVisibility(true);
```

Interfaz `GEOptions`

Permite especificar el comportamiento de diversas opciones de *Google Earth*, como navegación, velocidad de vuelo, velocidad de la rueda del ratón, etc.

Función

```
GEOptions.setFlyToSpeed (X.x)
```

Especifica la velocidad a la que se mueve la cámara (de 0 a 5,0). Si en lugar de una cifra se escribe `SPEED_TELEPORT` los movimientos de cámara se realizan inmediatamente, sin transiciones.

Función

```
GEOptions.setStatusBarVisibility(true/false)
```

Establece la visibilidad de la barra de estado. Por defecto está oculta.

Ejemplo:

```
ge.getOptions().setFlyToSpeed(0.3);

ge.getOptions().setStatusBarVisibility(true);
```

Interfaz `GENavigationControl`

Permite configurar los controles de navegación de *Google Earth*.

Función
`GENavigationControl.setVisibility(visibilidad)`

Especifica si los controles de navegación están visibles, ocultos o si se vuelven visibles cuando el usuario intenta usarlos.

El parámetro `visibilidad` puede tener los siguientes argumentos:

- `GEPlugin.VISIBILITY_SHOW`: Los controles están visibles.
- `GEPlugin.VISIBILITY_HIDE`: Los controles están ocultos.
- `GEPlugin.VISIBILITY_AUTO`: Los controles se vuelven visibles al intentar usarlos.

Ejemplo:

```
ge.getNavigationControl().setVisibility(ge.VISIBILITY_AUTO);
```

Interfaz `GEPlugin`

Se trata del objeto que se devuelve a la aplicación *JavaScript* cuando se crea por primera vez una instancia, i.e., un objeto perteneciente a una clase determinada. Dispone de patrones de diseño para generar otros objetos (marcas de posición y demás), y también se emplea para recuperar los objetos del documento raíz.

Interfaz para creación de objetos `GEPlugin.createLookAt(id)`

Se trata de una Interfaz basada en el comando `LookAt` de KML. Crea un nuevo elemento `LookAt`. Establece la posición de la cámara a partir de la posición del objeto al que se observa.

Función
`KmlLookAt.set (latitud, longitud, altitud, modo de altitud, orientación, inclinación, distancia)`

Establece la latitud, longitud, altitud, modo de altitud, orientación, inclinación y distancia para la cámara.

Ejemplo:

```
ge.createLookAt('').set(lat,lon,alt,alt_mode,head,tilt,range)
```

Interfaz para creación de objetos `GEPlugin.createCamera(id)`

Se trata de una Interfaz basada en el comando `Camera` de KML. Crea un nuevo elemento `Camera`. Este elemento también establece la posición de la cámara, pero esta vez a partir de sus coordenadas y altitud respecto a la superficie terrestre. Además define la dirección de la vista.

Función
`KmlCamera.set (latitud, longitud, altitud, modo de altitud, orientación, inclinación, balanceo)`

Establece la latitud, longitud, altitud, modo de altitud, orientación, inclinación y balanceo de la cámara.

Ejemplo:
```
ge.createCamera('').set(lat,lon,alt,alt_mode,head,tilt,roll)
```

Interfaz para creación de objetos `GEPlugin.createNetworkLink(id)`

Se basa en el comando `NetWorkLink` de KML. Permite crear un enlace a la red. Hace referencia a un archivo KML o KMZ en una red local o remota.

Función
`KmlNetworkLink.setLink(enlace)`

Especifica la localización de:

- Archivos KML recogidos por enlaces de red.
- Archivos de imagen empleados por iconos en estilos de icono, superposiciones, y superposiciones sobre la pantalla.
- Archivos de modelos empleados en el objeto modelo.

Ejemplo:
```
ge.createNetworkLink("").setLink(http://[...])
```

Interfaz para creación de objetos `GEPlugin.createLink(id)`

Se basa en el comando `KmlLink` de KML. Permite crear un enlace. Un enlace especifica la localización de archivos KML, archivos de imagen empleados en superposiciones o archivos de modelos 3D, entre otros.

Función
`KmlLink.setHref(cadena)`

Especifica una URL (tanto una dirección HTTP como un archivo local). Puede tratarse de un archivo KML, un archivo COLLADA o una imagen para superposición.

Ejemplo:
```
ge.createLink("").setHref(http://[...])
```

Interfaz para creación de objetos `GEPlugin.createIcon(id)`

Se basa en el comando `KmlIcon` de KML y comparte funciones heredadas de `KmlLink`. Crea un icono. Un icono define una imagen asociada con un estilo de icono y también la imagen que se utilizará para realizar una superposición.

Función.
`KmlLink.setHref(cadena)`

Establece una URL, tanto local como remota. Puede tratarse de un archivo KML, un archivo COLLADA o una imagen para superposición.

Ejemplo:
```
ge.createIcon('').setHref('http://[...]')
```

Interfaz para creación de objetos `GEPlugin.createScreenOverlay('')`

Se basa en el comando `KmlScreenOverlay` de KML. Superpone una imagen en la ventana que contiene *Google Earth*.

Función.
`KmlOverlay.setIcon(icono)`

Define la imagen asociada con la superposición.

Ejemplo:
```
ge.createScreenOverlay('').setIcon('http//[…]')
```

Función
`KmlOverlay.getOverlayXY()`

Define un punto respecto al origen de coordenadas de la ventana que servirá de referencia para la colocación de la imagen a superponer.

Subfunción
```
.setXUnits(unidades)
.setYUnits(unidades)
```

Establece en qué unidades están expresados los valores:

- `GEPlugin.UNITS_PIXELS`: *pixels* con origen en la esquina inferior izquierda de la pantalla.
- `GEPlugin.UNITS_FRACTION`: fracciones de pantalla.
- `GEPlugin.UNITS_INSET_PIXELS`: *pixels* con origen en la esquina superior derecha de la pantalla.

Ejemplo:
```
KmlOverlay.getOverlayXY().setXUnits(ge.UNITS_FRACTION)
KmlOverlay.getOverlayXY().setYUnits(ge.INSET_PIXELS)
```

Subfunción

```
.setX(valor)
.setY(valor)
```

Indica la coordenada *x* e *y*.

Ejemplo:

```
KmlOverlay.getOverlayXY().setX(100)
KmlOverlay.getOverlayXY().setY(90)
```

Función

```
KmlOverlay.getSize()
```

Define el tamaño de la imagen para la superposición en la ventana:

- Un valor de -1 indica usar la dimensión nativa de la imagen.
- Un valor de 0 indica mantener la relación de aspecto.
- Un valor de *n* establece el valor de la dimensión.

Subfunciones

```
.setXUnits(unidades)
.setYUnits(unidades)
.setX(valor)
.setY(valor)
```

Ejemplo:

```
KmlOverlay.getSize().setXUnits(ge.UNITS_PIXELS)
KmlOverlay.getSize().setYUnits(ge.UNITS_PIXELS)
KmlOverlay.getSize().setX(100)
KmlOverlay.getSize().setX(90)
```

Interfaz de miembro público `GEPlugin.getGlobe()`

Controla el comportamiento del Globo Terráqueo. Configura la interfaz **`GEGlobe`** dentro de la interfaz **`GEPlugin`**.

Interfaz `GEGlobe`

Se trata de una clase que contiene el objeto del globo terráqueo, restringe el acceso a él y determina el comportamiento de los sucesos asociados a él.

Función

```
GEGlobe.getFeatures()
```

Establece las entidades que se encuentran en el modelo 3D de la Tierra. Configura la interfaz **`GEFeatureContainer`** dentro de la interfaz **`GEGlobe`**.

Interfaz `GEFeatureContainer`

Se trata de un tipo de contenedor que almacena una o más entidades y permite la creación de jerarquías anidadas. Comparte funciones heredadas de **`GESchemaObjectContainer.`**

Función

`GeSchemaObjectContainer.appendChild(objeto KML)`

Añade un nodo al final de la lista de "child" de una entidad especifica y devuelve el objeto añadido (*`objeto KML`*)

Ejemplo:

```
ge.getGlobe().getFeatures().appendChild(objeto KML)
```

Interfaz `GEView.`

Establece el comportamiento de la cámara que muestra la escena.

Función

`GEView.setAbstractView(`**`view`**`)`

Configura la cámara que muestra la escena.

El parámetro `view` puede estar definido por la función `createLookAt('').set([…])`

Ejemplo:

```
la=ge.createLookAt('').set(39,0,0,ge.ALTITUDE_RELATIVE_TO_GROUND,0,5,99)
ge.getView().setAbstractView(la)
```

4.5.5. Programación de la página Web

4.5.5.1. Introducción

En este apartado va a tratarse el último paso en la elaboración de este Proyecto Final de Carrera, que ha sido la creación de una página Web que contuviera y presentara toda la información generada anteriormente y que aportara una serie de funciones que la hicieran interactiva con el usuario final.

Así pues, se comentará a continuación el código que se ha programado para dotarla de sus funciones interactivas, así como la necesidad que hubo al final del todo de confeccionar una segunda página Web para poder apreciar las diferencias entre modelos que se crearon en un principio en alta calidad y sus versiones remodeladas en baja calidad.

4.5.5.2. Código XHTML y JavaScript

XHTML es el sucesor de HTML. Basado en XML, se trata de una versión nueva y mejorada que los navegadores pueden manejar más fácilmente [187]. Su programación se ha complementado con el uso de *scripts*. Éstos incorporan numerosas funciones que permiten ejecutar multitud de acciones en la API de *Google Earth*, como se vio en el apartado anterior.

Por otro lado, para asegurar su buen funcionamiento en los principales navegadores Web que soportan instrucciones XHTML 1.0, se han seguido sus especificaciones [187]. Gracias a esto la página Web ha podido ser validada, realizándose este proceso mediante la aplicación "*W3C Markup Validation Service*" o "*Servicio de validación de marcado del W3C (Consorcio de la World Wide Web)*"[200].

Así pues, siguiendo las especificaciones, lo primero que se ha hecho ha sido programar el código de manera que el navegador sepa con qué se va a encontrar. Para ello se declara el tipo de documento, mediante el elemento `<!DOCTYPE […]>`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Esto establece que se trata de un documento XHTML 1.0 Transicional, esto es, que incorpora todos los elementos estructurales de XHTML, así como todos los elementos de presentación.

El siguiente paso consiste en añadir el espacio de nombre XHTML. Se trata de una colección de nombres empleados por los elementos y atributos de un documento XML. XHTML emplea una colección especial de nombres, siendo necesario declarar su espacio de nombre correspondiente para poder usarlos. La declaración se hace dentro del elemento `<HTML>`, que es el que define el documento como HTML.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="EN">
```

Cabecera `<head>`

El elemento `<head>` contiene la cabecera del documento. En esta parte del código se incluye información interna y funciones de la página Web que permanecen ocultas a los ojos del usuario, a excepción del título.

Dentro de la cabecera, y mediante el elemento `<title>`, se define el título que se asigna al documento, que es el que se muestra en la barra de título del navegador Web.

Por otro lado se utiliza el elemento `<meta>`, que contiene información sobre la propia página Web. Esta información se describe mediante pares de atributos. El primero se utiliza para establecer el tipo de información que se va a proporcionar y el segundo contiene la información en sí. De esta manera el atributo `http-equiv` indica que a continuación se va a determinar un tipo de contenido para la página Web, y el atributo `content` que se trata de una aplicación y código XHTML y XML y que el juego de caracteres empleado es el ISO-8859-3, utilizado en el Sur de Europa.

```
<head>
<title>Visita virtual a la Playa La Puebla de Farnals</title>
<meta http-equiv="Content-Type" content="application/xhtml+xml;
      charset=ISO-8859-3" />
```

A continuación, las parejas de atributos `name` y `content` establecen una serie de palabras clave (*`keyword`*) relacionadas con la página, que son: "`sketchup, farnals, modelos 3D`". De esta manera, cualquier navegador Web, cuando realice sus búsquedas, asociará esta página Web con dichas palabras clave. De igual manera, un segundo atributo `name` indica que se va a proporcionar una descripción de la página Web dentro del atributo `content`.

```
<meta name="keywords" content="sketchup, farnals, modelos 3D" />
<meta name="description" content="Modelo 3D de la Playa de Puebla de Farnals" />
```

Una vez establecidos el título y la información referente a la página, llega el momento de programar los *scripts* que se van a ejecutar. Para esta aplicación Web se han programado dos, cada uno contenido en su correspondiente elemento `<script>`.

El primer *script* es muy sencillo y se encarga de proporcionar la clave de la API de *Google Earth*, obligatoria para poder trabajar con ella.

```
<script src="http://www.google.com/jsapi?key=ABQIAAAAX2kgDSvnBBTiZa4stXNIoxQ-
3ueX12H_HJjd80O76zFaykCSwxSv_jgVPm9scUBi8wxJqmq0jNQeZA" type="text/javascript">
</script>
```

El segundo *script* es mucho más extenso que el anterior e incorpora todas las funciones de llamada a la API de *Google Earth*, que hacen posibles todas las acciones que pueden ejecutarse en la página Web. A continuación, dada su extensión, se analizará por partes.

El atributo `type` indica que el *script* está escrito en texto sencillo y en *JavaScript*.

```
<script type="text/javascript">
```

El comando `google.load` es de los pocos pertenecientes al lenguaje AJAX que se utiliza en la API de *Google Earth* [89]. Este tipo de comando permite que la vista actual se actualice sin tener que refrescar la página entera. Este comando concreto carga la API correspondiente al modelo de la Tierra (`Earth`) en su versión 1.

```
google.load("earth", "1");
```

A continuación se declaran las siguientes variables globales:

- `ge`, se trata del objeto que almacena el modelo 3D de la Tierra, y a través de él se emplearán todas las interfaces que se han comentado en el capítulo anterior.
- `la`, contiene la posición y configuración de la cámara.
- `r`, almacenan los datos sobre las rutas (`r1-r17`), que se programarán más adelante.
- `m`, contienen los modelos 3D de los complejos urbanísticos (`m0-m46`) y las marcas de posición de todos los establecimientos comerciales (`m47-m75`), catalogadas por tipo de comercio.

```
var ge = null;
var la = null;
var r1 = null;
[…]
var r17 = null;
var m0 = null;
[…]
var m75 = null;
```

Una vez declaradas las variables se programan todas las funciones necesarias para la página Web. Unas se encargarán de gestionar el comportamiento del modelo de la Tierra que muestra *Google Earth*, otras cargarán los modelos y los situarán sobre la superficie de la Tierra. También se han programados funciones para manejar las superposiciones sobre la pantalla, para reproducir rutas programadas o para poder hacer consultas sobre los comercios de la zona de estudio. A continuación se presentan todas y se describen sus características y funcionamiento.

Función `init`

Contiene la función que va a ejecutarse nada más cargarse la página Web. Así pues, lo primero que se hace es crear el objeto que contiene el modelo 3D de la Tierra con el que tendremos que trabajar. Se le asigna un identificador "`map3d`" y, si no surge ningún problema, se llama a la función de `initCallback`. En caso contrario, si se produjera un error, se llamaría a la función `failureCallback`.

```
function init() {google.earth.createInstance("map3d", initCallback, failureCallback);}
```

Función `initCallback`

Por definición, esta es la función a la que llama la función `init`, una vez se ha ejecutado correctamente.

Dado que se sabe que el modelo 3D de la Tierra ha cargado correctamente, se pasa a configurar diferentes opciones. Esta función gestiona todo lo que tiene que mostrarse en la ventana de la API, tanto sobre el modelo 3D de la Tierra como sobre la misma ventana. Para ello, primero se activa la visibilidad del modelo 3D, se establece la velocidad de movimiento de la cámara por defecto, se activan los controles de navegación en modo automático y se hace visible la barra de estado.

```
function initCallback(object) {
ge = object;
ge.getWindow().setVisibility(true);
ge.getOptions().setFlyToSpeed(0.3);
ge.getNavigationControl().setVisibility(ge.VISIBILITY_AUTO);
ge.getOptions().setStatusBarVisibility(true);
```

A continuación se cargan todos los modelos 3D y las marcas de posición, contenidos en archivos kmz y kml respectivamente. Para ello se llama a la función `getNL` y se le proporciona la URL donde estos están almacenados.

```
m0 = getNL('http://personales.alumno.upv.es/[…]/base-fotografica.kmz');
m1 = getNL('http://personales.alumno.upv.es/[…]/01-aparta-hotel.kmz');
[…]
m74 = getNL('http://personales.alumno.upv.es/alferol/farnals/ppf/e27.kml');
m75 = getNL('http://personales.alumno.upv.es/alferol/farnals/ppf/e28.kml');
```

Luego se inicializa la variable `la`, que es la encargada de contener los datos relativos a la posición de la cámara, a través de la interfaz `createLookAt`. Se establecen los datos iniciales de posición y configuración de la cámara, que van a mostrar una vista general de la zona de estudio, y por último se manda trasladar la cámara a dicha posición mediante la interfaz `getView`.

```
la = ge.createLookAt('');
la.set(39.56188889, -0.285022222, 0, ge.ALTITUDE_RELATIVE_TO_GROUND, 0, 50, 630);
ge.getView().setAbstractView(la);
```

Por último, la función `initCallback`, llama a dos funciones, que serán explicadas más adelante.

```
overlay();
updateOptions();}
```

Función `failureCallback`

Al contrario que la anterior, ésta es la función a la que la función `init` llama en caso de producirse un error.

En este caso la función se ha programado para que simplemente muestre en la pantalla un mensaje diciendo que se ha producido un error.

```
function failureCallback(object) {alert("Error en Callback");}
```

Función `getNL`

Recibe las URL donde se encuentran almacenados los diferentes modelos 3D y marcas de posición. A continuación los carga y los presenta en la ventana de la API. La variable `link` almacena la dirección URL, y la variable `nl` la establece como un enlace remoto.

```
function getNL(kmlURL){
var nl = ge.createNetworkLink("");
var link = ge.createLink("");
link.setHref(kmlURL);
nl.setLink(link);
```

Por último se añaden dichos modelos y marcas de posición al modelo de la Tierra y se presentan en la ventana.

```
ge.getGlobe().getFeatures().appendChild(nl);
return nl;}
```

Función `overlay`

Se ha programado para que superponga el escudo de la ETSIGCyT en una de las esquinas de la ventana de la API. Para ello lo primero que se hace es crear un icono, proporcionando para ello la URL en la que está almacenada la imagen del escudo. A continuación se establece que ésa es la imagen que va a utilizarse en la superposición. Se configura su posición en la ventana, el tamaño que va a tener y por último se presenta en la ventana.

```
function overlay(){
var icon = ge.createIcon('');
icon.setHref('http://personales.alumno.upv.es/[…]/topo.gif');
var screenOverlay = ge.createScreenOverlay('');
screenOverlay.setIcon(icon);
screenOverlay.getOverlayXY().setXUnits(ge.UNITS_FRACTION);
screenOverlay.getOverlayXY().setYUnits(ge.UNITS_FRACTION);
screenOverlay.getOverlayXY().setX(0.077);
screenOverlay.getOverlayXY().setY(0.895);
screenOverlay.getSize().setXUnits(ge.UNITS_PIXELS);
screenOverlay.getSize().setYUnits(ge.UNITS_PIXELS);
screenOverlay.getSize().setX(100);
screenOverlay.getSize().setY(90);
ge.getFeatures().appendChild(screenOverlay);}
```

Función `updateOptions`

En la página Web se muestran unos cajetines junto a los nombres de los distintos tipos de comercios. Dichos cajetines pueden activarse o desactivarse para presentar sobre el modelo de la Tierra las marcas de posición correspondientes a cada uno de ellos. Esta función se encarga de comprobar el estado de todos los cajetines y según estén activados o desactivados, establece la visibilidad de las marcas de posición correspondientes.

```
function updateOptions() {
m48.setVisibility(document.capas.e01.checked);
m49.setVisibility(document.capas.e02.checked);
[…]
m74.setVisibility(document.capas.e27.checked);
m75.setVisibility(document.capas.e28.checked);}
```

Función `ruta`

La página Web dispone de un menú desplegable que permite escoger entre diversas rutas. Cada una muestra diferentes partes de la zona de estudio del proyecto de forma automática. Para ello se ha programado esta función, que reconoce la ruta que se ha elegido en la página Web y la reproduce. Cada ruta está formada por un conjunto de posiciones geográficas y configuraciones de cámara propias a las que se irá trasladando la vista conforme va siendo reproducida.

Todas las rutas utilizan las mismas 17 variables (`r1 - r17`) para almacenar las posiciones y configuraciones de cámara que las conforman. Pero no todas emplean las 17. Algunas emplean sólo 7 y otras hasta 17. De esta forma puede ocurrir que, al escoger una ruta que usa un menor número de variables que la escogida con anterioridad, quedaran variables residuales que seguirían ejecutando parte de la ruta anterior, interfiriendo así con la actual. Para evitar esto lo primero que hace la función antes de poner en marcha alguna ruta es borrar todas las variables.

```
function ruta(){
ge.getOptions().setFlyToSpeed(0.1);
la = ge.createLookAt('');
clearTimeout(r1);
[…]
clearTimeout(r17);
```

La variable `tipo_ruta` almacena el valor contenido en la opción escogida dentro del menú desplegable de la página Web. Dependiendo de cuál sea dicho valor pondrá en marcha una ruta u otra.

```
var tipo_ruta = document.getElementById("opciones").value;
```

Una vez dentro de una ruta se asigna a cada variable `r` una función `setTimeout`. Dicha función llama a su vez a la función `changeLookAt`, proporcionándole a ésta los parámetros de posición y configuración de la cámara en ese punto de la ruta. La peculiaridad de la función `setTimeout` es establece en qué momento debe ser ejecutada la función que contiene, en este caso `changeLookAt`. Este dato aparece al final de la función expresado en milisegundos. De esta manera se pueden establecer las pausas en la ruta.

Por último se define la velocidad de movimiento de la cámara para la ruta actual. Puede observarse que dicha velocidad varía entre ruta y ruta dadas las diferentes características de cada una.

```
if(tipo_ruta == "ruta1") {
r1=setTimeout("changeLookAt(ge,la,39.5664,-0.2840, 26.0601,49.0850,175)",    0);
r2=setTimeout("changeLookAt(ge,la,39.5665,-0.2842,326.06  ,49.0856,175)", 2500);
[…]
r6=setTimeout("changeLookAt(ge,la,39.5664,-0.2841, 27.3791,56.7324, 63)",10000);
r7=setTimeout("changeLookAt(ge,la,39.5664,-0.2840, 26.0601,49.0850,1750)",13000);
ge.getOptions().setFlyToSpeed(0.6);}
[…]
if(tipo_ruta == "ruta10") {
r1=setTimeout("changeLookAt(ge,la,39.5665,-0.2849,206.5000,85.5000, 90)",    0);
r2=setTimeout("changeLookAt(ge,la,39.5657,-0.2854,206.5000,85.5000, 90)", 3000);
[…]
r6=setTimeout("changeLookAt(ge,la,39.5624,-0.2875,204.0000,85.5000, 90)",15000);
r7=setTimeout("changeLookAt(ge,la,39.5641,-0.2865, 26.0000,72.0000,300)",18000);
ge.getOptions().setFlyToSpeed(0.6);}}
```

Función changeLookAt

Recibe y gestiona los datos de posición y configuración de la cámara, y a continuación la traslada hasta dicha posición.

```
function changeLookAt(ge,la,lat,lng,head,tilt,range){
la.set(lat,lng,0,ge.ALTITUDE_RELATIVE_TO_GROUND,head,tilt,range);
ge.getView().setAbstractView(la);}
```

Función parar

La página Web dispone de un botón con un cuadrado dibujado dentro de él que sirve para detener la ruta que se está reproduciendo. Esta función es la encargada de detener dicha ruta, programándose para ello el borrado de todas las variables que la configuran.

```
function parar() {
clearTimeout(r1);
[…]
clearTimeout(r17);}
```

Función actualiza_listas

La página Web permite buscar un comercio concreto, ver su situación en el modelo, su marca de posición y una ventana de información propia. Para ello se ha creado un menú desplegable que permite elegir entre las diferentes categorías de comercios existentes. Una vez elegida, se rellena un segundo menú desplegable, situado justo debajo del anterior, con el listado de comercios pertenecientes a dicha categoría. Esto recibe el nombre de "listas dependientes".

Esta función fue programada para leer el valor de la categoría escogida en el primer menú desplegable y pasárselo a la función rellena.

```
function actualiza_listas() {
var categ = document.getElementById("categoria").value;
rellena(categ);}
```

Función rellena

Esta función está programada para leer el archivo que contiene los comercios pertenecientes a una determinada categoría y devolver un listado de dichos comercios, mostrándolo en el segundo menú desplegable. Para ello la función analiza el valor que le pasó la función `actualiza_listas`, y que aquí pasa a llamarse `archivo`. Según sea este valor, le asigna una variable `n` que se utilizará más adelante para la gestión de las capas de comercios.

```
function rellena(archivo) {
if (archivo == "e01") {var n=0}
        if (archivo == "e02") {var n=1}
        […]
        if (archivo == "e28") {var n=27}
```

El siguiente comando borra cualquier contenido residual que pudiera haber almacenado en el segundo menú desplegable (`comercio`).

```
comercio.options.length = 0;
```

El archivo del que podría extraerse el listado de comercios de una determinada categoría podría ser el archivo KML que contiene las marcas de posición de dicha categoría. El problema que se presenta es que el analizador sintáctico (*parser*) de los navegadores no permite cargar archivos con extensión KML. Es por eso que hay que crear archivos XML, que sí son aceptados por el *parser* para su análisis.

Lo más sencillo hubiera sido crear nuevos archivos XML a partir de los KML, cambiando sencillamente su extensión de archivo, sin modificar nada. Una mejor opción fue modificar los nuevos archivos XML dejando únicamente la información necesaria para este propósito: el nombre del comercio y la posición de cámara asociada a cada uno. De esta manera se ha ahorrado memoria, ya que los archivos se han visto reducidos notablemente, y a su vez la lectura por parte del *parser* es más rápida.

Para analizar el código XML primero debe crearse un DOM. Éste permite acceder y trabajar con los datos contenidos en él. Cada navegador Web tiene su propio comando específico para crear el DOM. Es por eso que la función comprueba en qué navegador se está ejecutando la página Web, y según cuál sea éste, crea el DOM con su comando específico. Una vez creado se carga el archivo XML correspondiente (`archivo+".xml"`), y se almacena en el DOM.

```
if (navigator.appName == "Netscape") {
        var xmlDoc = document.implementation.createDocument("","",null);
        xmlDoc.async = false;
        xmlDoc.load(archivo+".xml");
}

if (navigator.appName == "Microsoft Internet Explorer") {
        var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
        xmlDoc.async = false;
        xmlDoc.load(archivo+".xml");}
```

El *parser* empieza entonces a buscar las etiquetas "`Placemark`" en el código XML. La información contenida entre las diferentes etiquetas `<Placemark>` y `</Placemark>` se almacena en la serie de variables `markers (markers[0]-markers[n])`.

Luego se inicializa la variable `contador`, que lleva la cuenta del número de opción que se está generando en el segundo menú desplegable. Ésta aumenta desde 0 hasta el número total de etiquetas `<Placemark>`. De esta manera, si hay 9 comercios en el archivo XML (9 `placemark`), se generarán 9 opciones.

```
var markers = xmlDoc.getElementsByTagName("Placemark");
var contador = 0;
```

Después se pone en marcha una rutina que busca dentro de las etiquetas `<Placemark>` la etiqueta `<name>`, que contiene el nombre del comercio.

Se inicializa para ello la variable `i`, que incrementará su valor desde 0 hasta el número de `markers` generados anteriormente, esto es, el número de etiquetas `<Placemarks>` contenidas en el código XML.

Se inicializa también la variable `etiqueta`, que contiene el valor almacenado dentro de las etiquetas `<name>` (nombre del comercio).

A continuación se añade una nueva opción al segundo menú desplegable, y se le asigna el valor de la variable `etiqueta`, esto es, el nombre del comercio, que será el que muestre dicho menú.

Después se incrementa el valor de la variable `contador`. Este incremento se producirá mientras sigan encontrándose etiquetas `<Placemark>`.

```
for (var i = 0; i < markers.length; i++) {
var etiqueta=markers[i].getElementsByTagName("name")[0].childNodes[0].nodeValue;
comercio.options[contador]=new Option(etiqueta,etiqueta);
contador++;}
```

Como se ha comentado al principio de este apartado, la variable `n` se utiliza para la gestión de las capas de comercios. El propósito del siguiente código es asegurar que, cuando se elige un tipo de comercio en el menú desplegable, la capa que lleva asociada esté activa para que puedan verse sus marcas de posición. Así pues, establece la condición de que, si la capa elegida está desactivada, la active. Si no, no tiene que hacer nada y la deja como está, activada.

```
if (document.capas.elements[n].checked == false) {
document.capas.elements[n].click()}}
```

Función `localiza_nombre`

Una vez se ha elegido el comercio del segundo menú desplegable, la página Web dispone de un botón con la palabra "Ir" que está programado para trasladar la vista a hasta dicho comercio, mostrándose la situación del mismo, su marca de posición y una ventana de información si se pulsa sobre esta última.

Esta función se ha programado para saber qué comercio se ha elegido en el segundo menú desplegable y buscar su posición de cámara. Dicha información está almacenada en el archivo xml correspondiente a la categoría a la que pertenece dicho comercio. Este archivo es el mismo usado en la función `rellena`.

Primero de todo, y dado que cabe la posibilidad de que se estuviera reproduciendo una ruta antes de elegir un comercio y pulsar en el botón "ir", la función detiene cualquier ruta que pudiera estar reproduciéndose.

```
function localiza_nombre() {

clearTimeout(r1);
[…]
clearTimeout(r17);
```

A continuación averigua qué categoría y qué comercio han sido elegidos en el primer y segundo menú desplegable. La variable `archivo` almacena el nombre del archivo xml (sin extensión), correspondiente a dicha categoría. La variable `nombre` almacena el nombre del comercio elegido.

```
var archivo = document.getElementById("categoria").value;
var nombre = document.getElementById("comercio").value;
```

Después, se comprueba en qué navegador se está ejecutando la página Web, se crea el DOM, y se carga el archivo xml correspondiente.

```
if (navigator.appName == "Netscape") {
        var xmlDoc = document.implementation.createDocument("","",null);
        xmlDoc.async = false;
        xmlDoc.load(archivo+".xml");}

if (navigator.appName == "Microsoft Internet Explorer") {
        var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
        xmlDoc.async = false;
        xmlDoc.load(archivo+".xml");}
```

Se almacenan los contenidos de las etiquetas `<Placemark>` en la variable `markers`.

```
var markers = xmlDoc.getElementsByTagName("Placemark");
```

Se asigna un valor a `n` según sea el archivo cargado.

```
if (archivo == "e01") {var n=0}
[…]
if (archivo == "e28") {var n=27}
```

Se inicia a continuación una rutina que busca de entre todos los nombres contenidos dentro de las etiquetas `<Placemark>` el que coincide con el del comercio que se ha elegido en el segundo menú desplegable. Una vez encuentra la coincidencia, lee la información sobre la posición de la cámara de dicho comercio y la asigna a las variables `nlng, nlat, nrange, ntilt, nhead`.

```
for (var i = 0; i < markers.length; i++) {
var label = markers[i].getElementsByTagName("name")[0].childNodes[0].nodeValue;
if (label == nombre) {
   var nlng = markers[i].getElementsByTagName("longitude")[0].childNodes[0].data;
   var nlat = markers[i].getElementsByTagName("latitude")[0].childNodes[0].data;
   var nrange = markers[i].getElementsByTagName("range")[0].childNodes[0].data;
   var ntilt = markers[i].getElementsByTagName("tilt")[0].childNodes[0].data;
   var nhead = markers[i].getElementsByTagName("heading")[0].childNodes[0].data;
```

Se define la velocidad de movimiento de la cámara. Se establece la nueva posición de cámara y se traslada la cámara hasta dicha posición.

```
ge.getOptions().setFlyToSpeed(0.6);
la = ge.createLookAt('');
la.set(eval(nlat),eval(nlng),0,ge.ALTITUDE_RELATIVE_TO_GROUND,eval(nhead),
eval(ntilt), eval(nrange));
ge.getView().setAbstractView(la);}}
```

Por último se comprueba si la capa a la que pertenece el comercio elegido está desactivada. Si es así, se activa para poder ver las marcas de posición correspondientes. Y si ya estaba activada, se deja cómo estaba.

```
if (document.capas.elements[n].checked == false)
{document.capas.elements[n].click()}}
```

Y con la definición de esta última función terminarían los *scripts*. Así pues, una vez concluidos se cierra la parte de la cabecera (`<head>`) y se pasa a la parte encargada de la presentación de la página Web.

Cuerpo de la página `<body>`

Este elemento contiene toda la información del "cuerpo" de la página Web, esto es, todo aquello que aparece en la ventana del navegador.

Dentro del elemento `<body>` el atributo `onload` llama a la función `init` del *script* para que se ejecute una vez se haya cargado toda la página Web en el navegador. El atributo `id` determina un identificador para el cuerpo de la página y `bgcolor` establece un color de fondo para la página.

```
<body onload='init()' id='body' bgcolor="#488AC7">
```

Una vez definidas sus características, se procede a la maquetación de la página. Para ello se utilizan una serie de elementos `<div>`. Éstos permiten definir fácilmente la posición y características de cada una de las secciones en las que se divide la página, conteniendo cada una de ellas un elemento concreto de la aplicación.

Sección "Título"

Esta sección muestra un texto con el título de la página Web. Para ello, primero se define la posición de la sección en la ventana del navegador. Una vez hecho esto, se establece la fuente, el tamaño y el color del texto y se configura de manera que aparecerá centrado en la sección.

```
<div style='position: absolute; left: 270px; top:10px'>
<center>
   <font face="Verdana" size="4" color="white">
      <b>Visita virtual a la Playa de la Puebla de Farnals</b>
   </font>
</center>
</div>
```

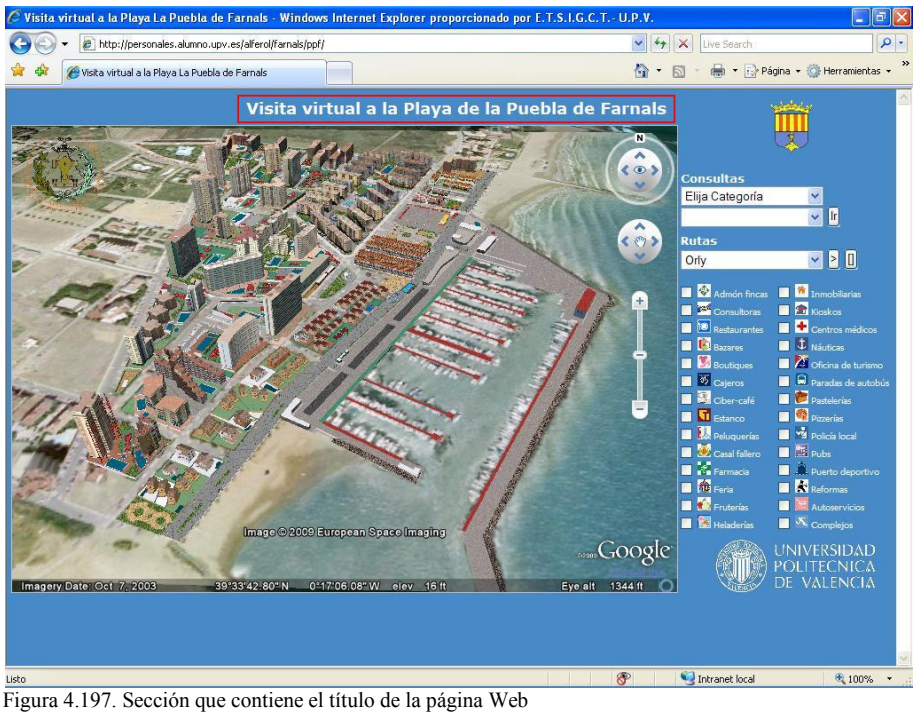


Figura 4.197. Sección que contiene el título de la página Web

Sección "API Google Earth"

La sección que se define a continuación contiene una ventana que a su vez contiene la API de Google Earth, donde se muestran todos los modelos del proyecto.

A la sección se le asocia un identificador único (`map3d_container`). Se define su altura y anchura, su posición absoluta respecto a la ventana del navegador y el grosor de su marco. Dentro de ella se crea otra subsección, que es la que contiene la API de *Google Earth* en sí. A esta subsección se le asocia también un identificador único (`map3d`), que es al que hace referencia la función `init` en el *script*. Por último se le da un tamaño del 100%, adaptándose así al de la sección principal, en la cual está incluida (Fig. 4.198).

```
<div id='map3d_container'
    style='border:1px solid silver; height:525px; width:750px; position:absolute;
           left:5px; top:40px'>
    <div id='map3d' style='height: 100%;'>
    </div>
</div>
```

Sección "Escudo del Ayuntamiento de La Puebla de Farnals"

En la siguiente sección se muestra el escudo del Ayuntamiento de la Puebla de Farnals. Para ello se indica la posición de la sección, el tamaño con que debe mostrarse la imagen y la URL en la que está almacenada.

```
<div style='position: absolute; left:860px; top:12px'>
<img  height="60" width="50" src="http://personales.alumno.upv.es/[...]/ayto.gif"
alt="icono ayto" />
</div>
```

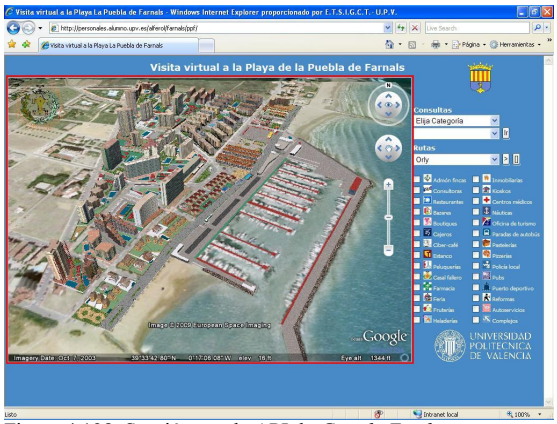


Figura 4.198. Sección con la API de *Google Earth*

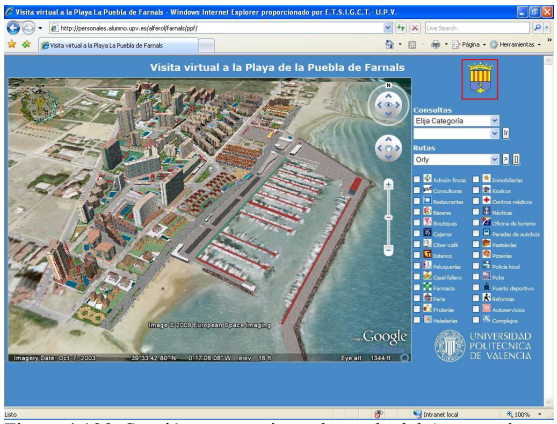


Figura 4.199. Sección que contiene el escudo del Ayuntamiento

Sección "Consultas"

La sección que se describe a continuación es la que contiene los menús desplegables que mostrarán las categorías de comercio y los nombres de éstos. Para ello, lo primero que se hace, de igual forma que se ha hecho con las secciones anteriores, es definir su posición en la ventana del navegador. Una vez hecho esto, se muestra un texto que servirá de título para esta sección, definiendo para éste una fuente, tamaño y color de texto.

A continuación, los elementos `<select>` definirán los menús desplegables. Cada uno lleva asociado un identificador único, que servirá para que el *script* pueda hacer consultas sobre ellos. En el primer menú desplegable se muestran todas las categorías de comercio existentes y se establece que, una vez elegida una categoría se llame a la función `actualiza_listas`. Cada categoría lleva asociado un valor (`e01 – e28`), que corresponde al nombre del fichero xml (sin extensión) que contiene la información de los comercios de dicha categoría. El segundo menú desplegable se deja preparado para que la función `rellena` lo rellene y muestre así los comercios pertenecientes a la categoría elegida. Por último se incluye un botón con el texto "Ir" para que, una vez elegido un comercio en el segundo menú desplegable, llame a la función `localiza_nombre`, la cual trasladará la cámara hasta el lugar donde se encuentre dicho comercio.

```
<div style='position: absolute; left:760px; top:90px'>
   <b><font face="Verdana" size="2" color="white">Consultas</font></b>
   <br />
   <select style="width: 161px" name="categoria" id="categoria"
                  onchange="actualiza_listas()">
      <option >Elija Categor&iacute;a</option>
      <option value="e01">Adm&oacute;n de fincas</option>
      <option value="e02">Consultoras</option>
      […]
      <option value="e27">Autoservicios</option>
      <option value="e28">Complejos deportivos</option>
   </select>
   <br />
      <select style="width: 161px" name="comercio" id="comercio">
      <option /></select>
      <input type="button" value="Ir" onclick="localiza_nombre()" />
</div>
```

En las siguientes imágenes puede verse el rellenado del segundo menú desplegable una vez elegida la categoría y la vista del comercio elegido en dicho menú después de haber pulsado el botón "Ir", mostrándose su situación y marca de posición.

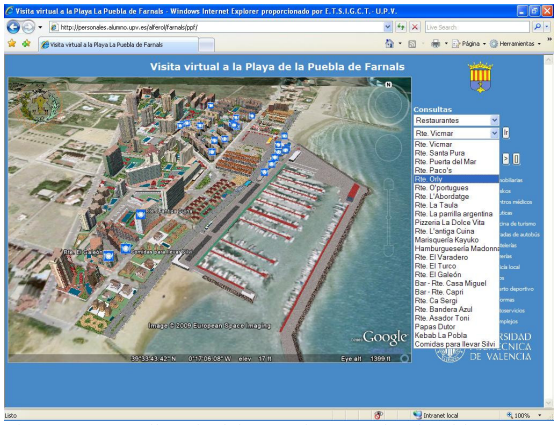


Figura 4.200. Rellenado del segundo menú desplegable

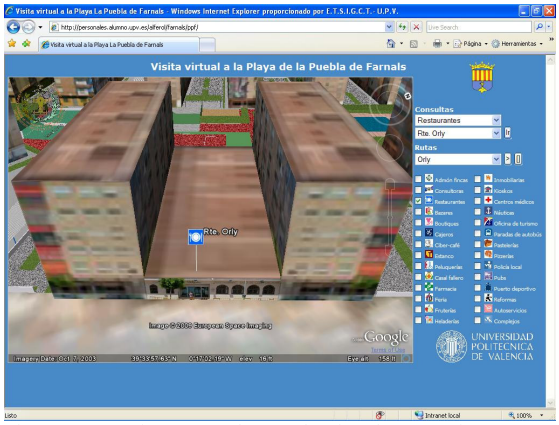


Figura 4.201. Vista generada tras elección de comercio

Sección "Rutas"

Esta sección contiene un menú desplegable que muestra las diferentes rutas que han sido programadas para ser reproducidas en la página Web.

Primero de todo se define la posición de la sección en la ventana del navegador. A continuación, igual que se hizo en la sección "Consultas", se presenta un texto con el título de esta sección. Se define para éste su fuente, tamaño y color.

Una vez hecho esto se define el elemento `<select>`. Se establece la anchura del menú desplegable y se le asocia un nombre y un identificador único. A continuación se muestran todas las opciones de rutas. Cada opción lleva asociado un valor determinado, correspondiente a un número de ruta. Dicho valor permite a la función `ruta` determinar la ruta que se ha elegido y reproducirla.

Al final de esta sección se han incluido dos botones. El primero lleva el signo ">" y se ha programado para que, una vez se ha elegido una ruta, se pulse sobre él y éste llame a la función `ruta`, la cual reproducirá aquella que se haya elegido. El segundo botón lleva el signo "[]" y se ha programado para que al pulsarlo llame a la función `parar`, la cual detiene la ruta que esté siendo reproducida en ese momento.

```
<div id='lista' style='position: absolute; left:760px; top:160px'>
   <b><font face="Verdana" size="2" color="white">Rutas</font></b>
   <br />
   <select style="width: 161px" name="opciones" id="opciones">
      <option value="ruta1">Orly</option>
      <option value="ruta2">Ambassador</option>
      […]
      <option value="ruta9">Avenida Neptuno</option>
      <option value="ruta10">Avenida Carabelas</option>
   </select>

   <input type="button" value=">"  onclick="ruta()" />
   <input type="button" value="[]" onclick="parar()" />
</div>
```

Las siguientes imágenes muestran la elección de una ruta y su posterior reproducción. En este caso la ruta elegida corresponde al Complejo Ramsés, que es mostrado al detalle (Fig. 4.203).

Figura 4.202. Elección de la ruta del Complejo Ramsés

Figura 4.203. La ruta muestra el complejo Ramsés

Sección "Gestor de categorías"

En esta sección se presentan todas las categorías de comercios existentes en la zona de estudio. Cada una dispone de un cajetín que permite activarla o desactivarla, mostrando u ocultando así las marcas de posición correspondientes a dicha categoría. La selección es múltiple, pudiendo llegar a mostrarse todas a la vez.

La presentación de las diferentes categorías se ha realizado mediante la inserción de una tabla con dos columnas dentro de un formulario. Cada celda muestra un cajetín de activación / desactivación, el icono que representa a dicha categoría y el nombre de la misma. Al pulsar sobre un cajetín, se llama a la función `updateOptions()`. Dicha función analiza todos los cajetines y según sea el estado de éstos, muestra u oculta las marcas de posición de las categorías correspondientes.

```
<div style='position: absolute; left:753px; top:215px'>
<form name="capas" action="javascript:updateOptions()">
   <table width="248">
   <tr>
      <td width ="110" valign="top" align="left">
      <font face="Tahoma" size="1" color="white">
      <input type="checkbox" onclick='updateOptions()' name="e01" value="ON" />
      <img height="15" width="15"
       src="http://personales.alumno.upv.es/[...]/icono 01.gif"
       alt="icono admon" />
      <a> Adm&oacute;n fincas</a>
      <br />
      […]
      <input type="checkbox" onclick='updateOptions()' name="e28" value="ON" />
      <img height="15" width="15"
       src="http://personales.alumno.upv.es/[...]/icono 28.gif"
       alt="icono complejos" />
      <a> Complejos</a>
      <br />
      </font>
      </td>
   </tr>
   </table>
</form>
</div>
```

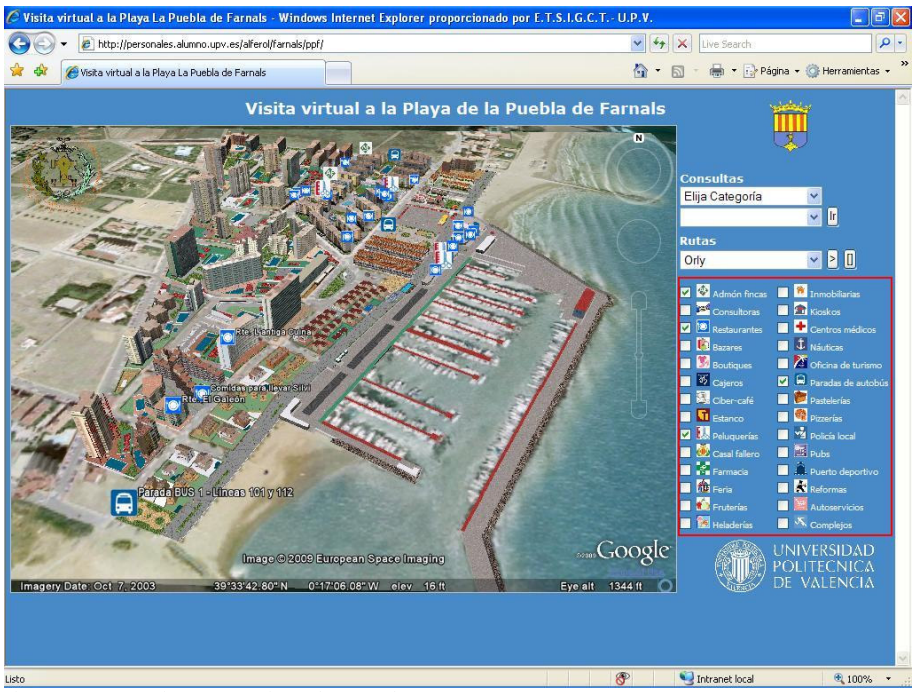


Figura 4.204. Sección que contiene el gestor de categorías

Sección "Logotipo de la Universidad Politécnica de Valencia"

Esta última sección muestra el logotipo de la Universidad Politécnica de Valencia. Para ello, igual que se hizo en la presentación del escudo del Ayuntamiento, primero se establece la posición de la sección en la ventana del navegador. Después se define el tamaño con que debe mostrarse el logotipo y se indica la URL en la que está almacenada la imagen.

```
<div style='position: absolute; left:780px; top:507px'>
   <img  height="60" width="200"
    src="http://personales.alumno.upv.es/alferol/farnals/ppf/imagenes/upv.gif"
    alt="icono UPV" />
</div>
```

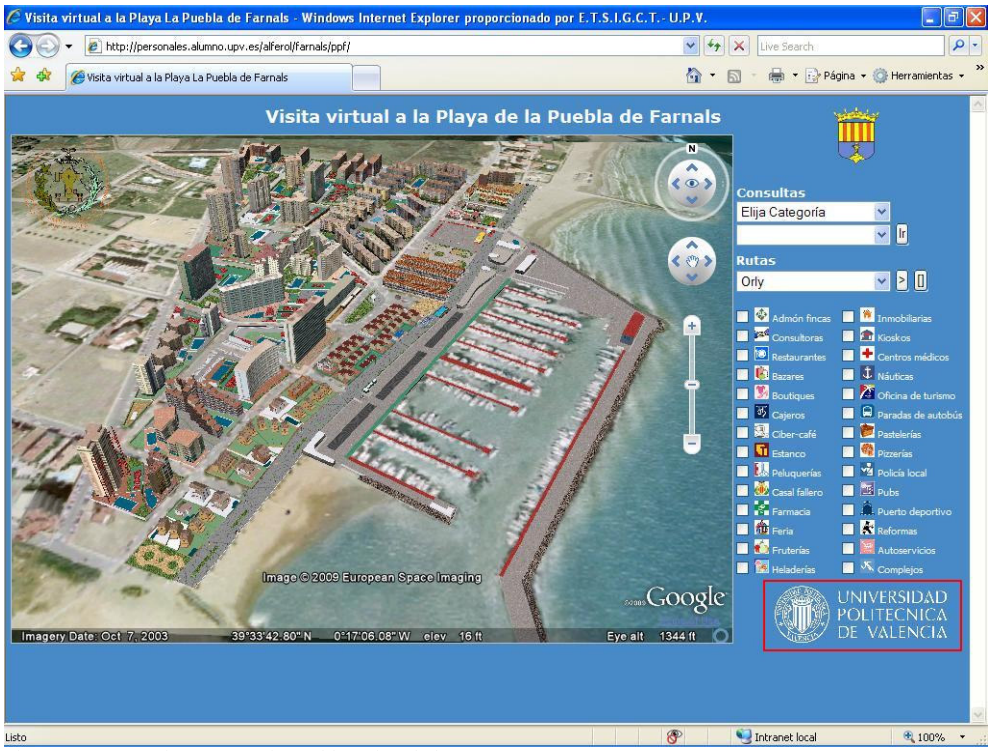


Figura 4.205. Sección que contiene el logotipo de la UPV

De esta manera, habiéndose terminado de definir todas las secciones, se cierra el cuerpo de la página Web y el código XHTML.

```
</body>

</html>
```

Por último, la página Web y todos los archivos necesarios para su correcto funcionamiento han sido almacenados en el servidor de la UPV, siendo accesible desde la siguiente dirección:

```
http://personales.alumno.upv.es/alferol/farnals/ppf/index.html
```

4.5.5.3. Creación de una segunda página Web

Como ya se ha explicado anteriormente, la carga de todos los modelos en alta calidad exige muchos recursos de procesador, memoria de ordenador, tarjeta gráfica y ancho de banda. Para evitar este inconveniente se hizo una segunda serie de modelos en baja calidad, que son los que se presentan en la anterior página Web.

No obstante, con el objeto de poder mostrar los modelos en alta calidad se pensó en programar una segunda página Web, esta vez desprovista de casi todas las funciones de la original, que mostrara tanto los modelos en baja como los de alta calidad para poder apreciar así las diferencias entre ellos. Tampoco se muestran todos los modelos que se hicieron en alta calidad, sino sólo los tres más característicos, debido a que el tiempo de carga podría ser algo largo. Hay que tener en cuenta que estos modelos ocupan bastante memoria. De hecho, los tres modelos que se presentan en esta segunda página Web ocupan más de la mitad de la memoria que ocupan los 46 modelos presentados en la anterior página Web (Tabla 4.2.).

Modelo	Alta calidad (KB)	Baja calidad (KB)	% reducción	Modelo	Alta calidad (KB)	Baja calidad (KB)	% reducción
Aparta-Hotel	442	76	83	Torre Estudio II	1314	147	89
Orly	1633	138	92	Torre Sol	820	152	81
Complejo Carabelas	1030	138	87	Príncipe II	860	117	86
Complejo Delfos	823	160	81	Edificio San Pablo	486	115	76
Complejo Sabas	848	176	79	Ambassador	1695	156	91
Complejo Keops	1041	158	85	Torre Estudio III	922	136	85
Estudio I	1007	99	90	Olimpo	600	88	85
Urbanización Port-Mar	849	117	86	Edificio Mare Nostrum	1078	161	85
Urbanización El Ancla	750	131	83	Parque poligono 46	202	31	85
Inmobiliaria Daemi	250	78	69	Presidente V	885	149	83
Restaurante La Góndola	187	80	57	EdificioApolo	519	88	83
Horchatería El Paso	531	97	82	Complejo Neptuno	667	91	86
Paseo Marítimo	1493	164	89	Ramsés	708	113	84
Plaza de Las CCVV	690	154	78	Edificio Presidente	1028	193	81
Parking Plaza CCVV	410	122	70	Presidente Lux	743	117	84
Puerto D. Pobla Marina	1796	129	93	Complejo Madreselva	657	135	79
Puebla Nueva	1240	183	85	Puebla I	725	118	84
Zodiaco	660	157	76	Complejo Vannes	957	131	86
Parque Granell	702	149	79	Complejo Copacabana	871	148	83
Aries III	745	206	72	Complejo Lord	392	81	79
Edificio Madeira	756	145	81	Plaza de Paris	179	78	56
Playa-Mar	766	148	81	Villes de Port-Lux	1436	106	93
Edificio Pleamar	488	198	59	Chalets Puerto	1742	97	94

Tabla 4.2. Relación de memoria empleada en los modelos de alta y baja calidad

La página muestra dos ventanas con la API de *Google Earth*. La primera de ellas muestra los tres modelos elegidos en baja calidad. A su lado, otra ventana muestra los modelos en alta calidad, pudiéndose apreciar las diferencias entre la primera serie de modelos y la segunda.

Esta segunda página Web ha sido creada a partir del código XHTML de la descrita en este capítulo, dejando tan sólo la opción de reproducir tres rutas, las cuales muestran estos tres complejos desde diferentes posiciones de cámara.

La peculiaridad del reproductor de rutas en esta página Web es que reproduce la misma ruta sincronizada en las dos ventanas, pudiéndose así apreciar claramente las diferencias entre modelos de forma automática. Para ello se tuvo que crear un objeto contenedor para cada API y enviar la misma orden de reproducción de ruta a cada uno. El resultado ha sido muy satisfactorio.

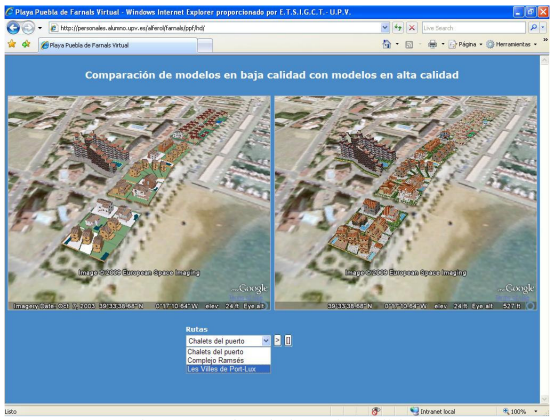


Figura 4.206. Vista general en baja y alta calidad

Figura 4.207. Vista detallada en baja y alta calidad

De igual manera que ocurre con la anterior página Web, los archivos necesarios para el correcto funcionamiento de esta segunda página Web han sido almacenados en el servidor de la UPV, siendo accesible desde la siguiente dirección:

`http://personales.alumno.upv.es/alferol/farnals/ppf/hd/index.html`

Capítulo 5

Presupuesto

5.1. Introducción

En este capítulo se muestra el presupuesto correspondiente a los trabajos realizados en la totalidad del proyecto.

Dada la división de trabajo establecida por las diferentes fases del proyecto se ha desglosado el presupuesto en capítulos y apartados. Cada apartado contiene una explicación de los trabajos realizados y el tiempo empleado en cada uno. A continuación y según el tipo de trabajo realizado, se han asignado unos honorarios por hora a cada apartado. De esta manera se han obtenido los diferentes costes parciales, coste total por capítulo y el coste final del proyecto.

Cabe destacar que durante la realización de la mayor parte del proyecto (a excepción de la creación de la página Web) no se realizó ninguna medición de tiempo. Además, los tiempos empleados al principio de este proyecto (sobretodo en el apartado de modelado 3D), dada la inexperiencia del autor, hubieran diferido mucho de los empleados hacia el final del mismo, cuando el trabajo ha ido bastante más rápido.

Por tanto, para calcular aproximadamente el tiempo necesario para acabar un apartado, se ha optado por repetir parte de dicho proceso y medir el tiempo empleado en su ejecución. Una vez conocido este dato, se ha calculado el tiempo que se hubiera empleado para acabar la totalidad del apartado.

Así, en el capítulo "Trabajos fotográficos" se han repetido los procesos de tratamiento panorámico, retoque fotográfico, rectificación y teselado para una cantidad pequeña de fotos, midiéndose el tiempo empleado en cada uno. A partir de estos datos y de la cantidad total de fotos correspondiente a cada apartado se ha calculado el tiempo aproximado que sería necesario para acabar cada uno de ellos.

De igual manera, en el capítulo "Generación de los modelos 3D", se volvió a modelar un complejo desde cero. Se midió el tiempo empleado, y a partir de él se calcularon los tiempos necesarios para el modelado de cada uno de los complejos restantes.

El modelo elegido para ser modelado desde cero fue el del Complejo Orly. Dicho complejo es un buen modelo de referencia por disponer de elementos y componentes presentes en muchos complejos y ocupar una cantidad de memoria aceptable, representando así el modelo medio.

Cabe destacar que cada complejo presenta unas características diferentes a nivel de modelado. Algunos, tienen muchos polígonos y poco trabajo de texturizado. Otros, al contrario, tienen pocos polígonos y mucho trabajo de colocación de fotografías de fachadas. Así pues, para poder asignar tiempos lo más rigurosos posibles a cada complejo, se dividió el trabajo de modelado en tres fases, atendiendo éstas a las principales características de los complejos y midiéndose un tiempo para cada una:

1) Modelado sin texturas

Se trata del trabajo de modelado partiendo de la base cartográfica hasta llegar al modelo completo, sin texturas e incluyendo todas las estructuras y componentes.

En esta fase se ha tomado como referencia el número de polígonos que tiene el modelo al final del proceso y el tiempo empleado en generarlos. Una vez conocido este dato, se calcula el tiempo necesario para modelar cada uno de los complejos restantes según su número de polígonos.

2) Colocación de las fotografías de las fachadas

En esta fase se ha contado el número de polígonos sobre los que hay que colocar una fotografía y se ha medido el tiempo empleado en su colocación. Así pues, a partir de estos datos y del número de polígonos que llevan colocadas fotografías en cada uno de los complejos restantes, se han calculado sus respectivos tiempos de colocación.

3) Texturizado del resto del modelo

En esta fase, igual que en la anterior, se ha contado el número de polígonos que han sido texturizados y se ha medido el tiempo empleado en dicho proceso. A partir de estos datos y del número de polígonos texturizados en cada uno de los complejos restantes, se han calculado sus respectivos tiempos de texturizado.

Por otro lado, y sin entrar en detalles, en el capítulo “Creación de las marcas de posición” se ha procedido de igual manera a la explicada anteriormente.

Y por último, en el capítulo “Creación de la página Web” cabe destacar que sí se hicieron mediciones de los tiempos empleados en cada uno de los apartados.

5.2. Resumen del presupuesto

			Parcial	Total	Total+IVA	
Capítulo 1. Trabajos fotográficos						
1.01 Ortofoto digital			36.00€			
1.02 Fotografía de fachadas	11h00’	9€/h	99.00€			
1.03 Tratamiento panorámico	6h05’	6€/h	36.50€			
1.04 Retoque fotográfico	4h12’	9€/h	37.80€			
1.05 Rectificación y redimensión	22h25’	9€/h	201.75€			
1.06 Creación de teselas	1h20’	9€/h	12.00€			
				423.05€	490.74€	
Capítulo 2. Generación de modelos 3D						
2.01 Modelo Apart-Hotel ~ 2.46 Modelos mobiliario	43h40’	12€/h	524.00€			
				524.00€	607.84€	
Capítulo 3. Marcas de posición						
3.01 Salidas a campo	1h20’	6€/h	8.00€			
3.02 Recogida de datos	4h00’	6€/h	24.00€			
3.03 Búsqueda de información	22h15’	6€/h	133.50€			
3.04 Tratamiento de la información	14h50’	6€/h	89.00€			
3.05 Creación de las marcas	51h55’	9€/h	467.25€			
				721.75€	837.23€	
Capítulo 4. Creación de la página Web						
4.01 Edición página Web	25h00’	12€/h	300.00€			
4.02 Edición de rutas	10h00’	9€/h	90.00€			
4.03 Contratación dominio y host	1 año	150€/a	150.00€			
				540.00€	626.40€	
						Coste final 2562.21€

5.3. Presupuesto

Capítulo 1. Trabajos fotográficos

1.01 Ortofoto digital

Compra de ortofoto en formato digital, de la serie ODCV50, de la zona de estudio en el Instituto Cartográfico Valenciano.

		Parcial
1 ortofoto		36.00€

1.02 Fotografías de fachadas

Salida a campo para la toma de fotografías de las fachadas, que luego son usadas como texturas de los correspondientes modelos de complejos deportivos.

Desplazamiento al lugar de estudio

3 salidas	40'/salida	2h		
Toma de fotos				
3 sesiones	3h/sesión	9h		Parcial
	t. total:	11h	9€/h	99.00€

1.03 Tratamiento panorámico

Montaje de la fotografía de una fachada a partir de varias fotografías de la misma dada la imposibilidad de obtenerla en una sola toma.

				Parcial
73 fotografías	5'/foto	6h05'	6€/h	36.50€

1.04 Retoque fotográfico

Retoque fotográfico realizado a las fotografías de fachadas por motivos estéticos.

				Parcial
21 fotografías	12'/foto	4h12'	9€/h	37.80€

1.05 Rectificación y redimensión de todas las fotos

Todas las fotos son rectificadas/normalizadas así como redimensionadas para una mayor economía en la memoria empleada.

				Parcial
269 fotografías	5'/foto	22h25'	9€/h	201.75€

1.06 Creación de teselas

Preparación de fotos para su uso como teselas en las fachadas de aquellos edificios de los que no se tiene fotografías de las mismas al completo.

				Parcial
40 fotografías	2'/foto	1h20'	9€/h	12.00€

Total	Total+IVA
423.05€	490.74€

Capítulo 2. Generación de los modelos 3D

La creación de los modelos 3D de los diferentes complejos deportivos se compone de tres fases: Modelado 3D de todo el complejo, colocación de las fotografías de las fachadas en sus correspondientes edificios y, por último, texturizado del resto del modelo.

2.01 Modelado Apart-hotel

Modelado 3D	18’			
Colocación fotos de fachadas	2’			
Texturizado	10’			Parcial
tiempo total:		30’	12€/h	6€

2.02 Modelado Orly

Modelado 3D	27’			
Colocación fotos de fachadas	23’			
Texturizado	12’			Parcial
tiempo total:		62’	12€/h	12.40€

2.03 Modelado Carabelas

Modelado 3D	27’			
Colocación fotos de fachadas	8’			
Texturizado	14’			Parcial
tiempo total:		49’	12€/h	9.80€

2.04 Modelado Delfos

Modelado 3D	22’			
Colocación fotos de fachadas	6’			
Texturizado	11’			Parcial
tiempo total:		39’	12€/h	7.80€

2.05 Modelado Sabas

Modelado 3D	23’			
Colocación fotos de fachadas	17’			
Texturizado	10’			Parcial
tiempo total:		50’	12€/h	10.00€

2.06 Modelado Keops

Modelado 3D	38’			
Colocación fotos de fachadas	11’			
Texturizado	20’			Parcial
tiempo total:		69’	12€/h	13.80€

2.07 Modelado Estudio I

Modelado 3D	13’			
Colocación fotos de fachadas	5’			
Texturizado	7’			Parcial
tiempo total:		25’	12€/h	5.00€

2.08 Modelado Port-Mar

Modelado 3D	49’			
Colocación fotos de fachadas	14’			
Texturizado	26’			Parcial
tiempo total:		89’	12€/h	17.80€

2.09 Modelado Urbanización El Ancla

Modelado 3D	27’			
Colocación fotos de fachadas	3’			
Texturizado	15’			Parcial
tiempo total:		45’	12€/h	9.00€

2.10 Modelado Inmobiliaria Daemi

Modelado 3D	9’			
Colocación fotos de fachadas	2’			
Texturizado	5’			Parcial
tiempo total:		16’	12€/h	3.20€

2.11 Modelado Restaurante La Góndola

Modelado 3D	31’			
Colocación fotos de fachadas	5’			
Texturizado	17’			Parcial
tiempo total:		53’	12€/h	10.60€

2.12 Modelado Heladería El Paso

Modelado 3D	13’			
Colocación fotos de fachadas	5’			
Texturizado	6’			Parcial
tiempo total:		24’	12€/h	4.80€

2.13 Modelado Paseo Marítimo

Modelado 3D	24’			
Colocación fotos de fachadas	2’			
Texturizado	13’			Parcial
tiempo total:		39’	12€/h	7.80€

2.14 Modelado Plaza de Las Cortes Valencianas

Modelado 3D	34’			
Colocación fotos de fachadas	1’			
Texturizado	20’			Parcial
tiempo total:		55’	12€/h	11.00€

2.15 Modelado Parking de la Plaza de Las Cortes Valencianas

Modelado 3D	9’			
Colocación fotos de fachadas	6’			
Texturizado	4’			Parcial
tiempo total:		19’	12€/h	3.80€

2.16 Modelado Puerto Deportivo

Modelado 3D	26’			
Colocación fotos de fachadas	38’			
Texturizado	8’			Parcial
tiempo total:		72’	12€/h	14.40€

2.17 Modelado Puebla Nueva

Modelado 3D	80’			
Colocación fotos de fachadas	81’			
Texturizado	33’			Parcial
tiempo total:		194’	12€/h	38.80€

2.18 Modelado Zodiaco

Modelado 3D	18’			
Colocación fotos de fachadas	6’			
Texturizado	9’			Parcial
tiempo total:		33’	12€/h	6.60€

2.19 Modelado Parque Granell

Modelado 3D	7’			
Colocación fotos de fachadas	2’			
Texturizado	4’			Parcial
tiempo total:		13’	12€/h	2.60€

2.20 Modelado Aries III

Modelado 3D	10’			
Colocación fotos de fachadas	2’			
Texturizado	5’			Parcial
tiempo total:		17’	12€/h	3.40€

2.21 Modelado Edificio Madeira

Modelado 3D	35’			
Colocación fotos de fachadas	13’			
Texturizado	18’			Parcial
tiempo total:		66’	12€/h	13.20€

2.22 Modelado Edificio Playa-Mar

Modelado 3D	17’			
Colocación fotos de fachadas	7’			
Texturizado	8’			Parcial
tiempo total:		32’	12€/h	6.40€

2.23 Modelado Edificio Pleamar

Modelado 3D	9’			
Colocación fotos de fachadas	3’			
Texturizado	4’			Parcial
tiempo total:		16’	12€/h	3.20€

2.24 Modelado Estudio II

Modelado 3D	18’			
Colocación fotos de fachadas	7’			
Texturizado	9’			Parcial
tiempo total:		34’	12€/h	6.80€

2.25 Modelado Torre Sol

Modelado 3D	24’			
Colocación fotos de fachadas	3’			
Texturizado	13’			Parcial
tiempo total:		40’	12€/h	8.00€

2.26 Modelado Edificio Príncipe II

Modelado 3D	16’			
Colocación fotos de fachadas	12’			
Texturizado	7’			Parcial
tiempo total:		35’	12€/h	7.00€

2.27 Modelado Edificio San Pablo

Modelado 3D	10’			
Colocación fotos de fachadas	15’			
Texturizado	3’			Parcial
tiempo total:		28’	12€/h	5.60€

2.28 Modelado Ambassador

Modelado 3D	31’			
Colocación fotos de fachadas	19’			
Texturizado	14’			Parcial
tiempo total:		64’	12€/h	12.80€

2.29 Modelado Torre Estudio III

Modelado 3D	16’			
Colocación fotos de fachadas	5’			
Texturizado	8’			Parcial
tiempo total:		29’	12€/h	5.80€

2.30 Modelado Complejo Olimpo

Modelado 3D	20’			
Colocación fotos de fachadas	16’			
Texturizado	8’			Parcial
tiempo total:		44’	12€/h	8.80€

2.31 Modelado Complejo Mare Nostrum

Modelado 3D	27’			
Colocación fotos de fachadas	30’			
Texturizado	10’			Parcial
tiempo total:		67’	12€/h	13.40€

2.32 Modelado Príncipe V

Modelado 3D	19’			
Colocación fotos de fachadas	7’			
Texturizado	10’			Parcial
tiempo total:		36’	12€/h	7.20€

2.33 Modelado Edificio Apolo

Modelado 3D	16’			
Colocación fotos de fachadas	7’			
Texturizado	8’			Parcial
tiempo total:		31’	12€/h	6.20€

2.34 Modelado Complejo Neptuno

Modelado 3D	34’			
Colocación fotos de fachadas	17’			
Texturizado	17’			Parcial
tiempo total:		68’	12€/h	13.60€

2.35 Modelado Complejo Ramsés

Modelado 3D	32’			
Colocación fotos de fachadas	73’			
Texturizado	6’			Parcial
tiempo total:		111’	12€/h	22.20€

2.36 Modelado Edificio Presidente

Modelado 3D	28’			
Colocación fotos de fachadas	7’			
Texturizado	15’			Parcial
tiempo total:		50’	12€/h	10.00€

2.37 Modelado Presidente Lux

Modelado 3D	24’			
Colocación fotos de fachadas	2’			
Texturizado	14’			Parcial
tiempo total:		40’	12€/h	8.00€

2.38 Modelado Complejo Madreselva

Modelado 3D	22’			
Colocación fotos de fachadas	16’			
Texturizado	10’			Parcial
tiempo total:		48’	12€/h	9.60€

2.39 Modelado Puebla I

Modelado 3D	27’			
Colocación fotos de fachadas	10’			
Texturizado	14’			Parcial
tiempo total:		51’	12€/h	10.20€

2.40 Modelado Complejo Vannes

Modelado 3D	20’			
Colocación fotos de fachadas	5’			
Texturizado	10’			Parcial
tiempo total:	35’	12€/h		7.00€

2.41 Modelado Complejo Copacabana

Modelado 3D	22’			
Colocación fotos de fachadas	31’			
Texturizado	7’			Parcial
tiempo total:	60’	12€/h		12.00€

2.42 Modelado Complejo Lord

Modelado 3D	17’			
Colocación fotos de fachadas	15’			
Texturizado	7’			Parcial
tiempo total:	39’	12€/h		7.80€

2.43 Modelado Plaza de París

Modelado 3D	7’			
Colocación fotos de fachadas	6’			
Texturizado	3’			Parcial
tiempo total:	16’	12€/h		3.20€

2.44 Modelado Les Villes de Port Lux

Modelado 3D	64’			
Colocación fotos de fachadas	21’			
Texturizado	33’			Parcial
tiempo total:	118’	12€/h		23.60€

2.45 Modelado Chalets Puerto

Modelado 3D	62’			
Colocación fotos de fachadas	49’			
Texturizado	28’			Parcial
tiempo total:	139’	12€/h		27.80€

2.46 Modelado Mobiliario

Parking cubierto (10 plazas)	45’			
Parking cubierto (5 plazas)	45’			
Columnas soporte Príncipe V	60’			
Paellero	60’			
Pista de pádel	60’			
Pista de tenis	60’			Parcial
tiempo total:	330’	12€/h		66.00€

Total	Total+IVA
524.00€	607.84€

Capítulo 3. Creación de las marcas de posición

3.01 Salida a campo para recogida de información

Desplazamiento hasta la zona de estudio.

Desplazamiento al lugar de estudio

				Parcial
2 salidas	40'/salida	1h20'	6€/h	8.00€

3.02 Recogida de datos de los diversos establecimientos

Visita a cada uno de los comercios para obtener tarjetas de visita, publicidad y demás información referente los mismos y sus servicios.

				Parcial
2 sesiones	2h/sesión	4h	6€/h	24.00€

3.03 Búsqueda de información complementaria

Recopilación, a través de los principales motores de búsqueda de internet, así como en diferentes revistas locales y llibrets falleros, de información complementaria de los establecimientos, de sus logotipos y de iconos que representen los diferentes tipos de comercios.

				Parcial
89 comercios	15'/comercio	22h15'	6€/h	133.50€

3.04 Tratamiento de la información complementaria

Escaneo y arreglo de las tarjetas de visita, publicidad, logotipos, iconos, etc. recopilados anteriormente.

				Parcial
89 comercios	10'/comercio	14h50'	6€/h	89.00€

3.05 Creación de la marca de posición

Diseño y generación de todas las marcas de posición, que incluyen una imagen de la tarjeta de visita o logotipo del comercio, una descripción, su dirección y una fotografía del mismo. Internamente se establece la posición geográfica de la cámara y la configuración de la misma para que muestre el comercio correctamente.

				Parcial
89 comercios	35'/comercio	51h55'	9€/h	467.25€

Total	Total+IVA
721.75€	837.23€

Capítulo 4. Creación de la página Web

4.01 Edición página Web

Maquetación y diseño página Web. Edición código XHTML. Edición código *JavaScript*. Corrección de errores para validación por W3C. Testeo de la página en varios navegadores Web.

			Parcial
Creación página Web	25h	12€/h	300.00€

4.02 Edición de rutas

Diseño de las 10 rutas disponibles y establecimiento de la posición geográfica y configuración de la cámara en cada uno de los puntos que configuran cada ruta.

				Parcial
10 rutas	1h/ruta	10h	9€/h	90.00€

4.03 Contratación dominio Web y hosting

El dominio es la dirección de la página web (p.ej. *http://www.pueblafarnals.es*). El hosting o alojamiento web es el servicio que provee un sistema para poder almacenar la información, imágenes y demás contenido accesible vía Web.

Contratación del dominio	1año	100€/a	100.00€
Contratación del host	1año	50€/a	50.00€

	Parcial
	150.00€

Total	Total+IVA	Coste final
540.00€	626.40€	2562.21€

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Como ocurre al terminar cualquier proyecto, hay cuestiones que quedan pendientes de un mayor desarrollo, a la vez que se ven nuevas líneas que se podrían seguir para un mejor desarrollo futuro de la aplicación. A su vez se va comprobando que existen mejores métodos y técnicas para llevar a cabo el trabajo que, aunque a posteriori parecen obvias, al principio, con la falta de experiencia en el manejo de los diferentes programas y con el desconocimiento de la implicación de ciertas acciones y decisiones, no se veían de forma tan clara.

Así pues, se ha comprobado a lo largo de la memoria como en una primera fase, de alta calidad, se fue modelando mientras se iba aprendiendo la técnica sin tener en cuenta ningún límite de memoria. Se utilizaban todas las herramientas y técnicas de que disponía *SketchUp* sin tener en consideración el tamaño de memoria de los modelos generados. De esta manera el acabado final de estos modelos era muy bueno. Del mismo modo se actuó con el tratamiento de las imágenes que servirían como texturas en las fachadas de los modelos 3D. Cuanto mayor era el tamaño de las fotos mayor era el realismo de los modelos, obviamente. Pero, como ya se explicó, el movimiento de todo el conjunto bajo *Google Earth* era muy lento. Así se concluyó que era necesario establecer unas limitaciones, que implicaban un remodelado y una reducción en la calidad de las imágenes para que la memoria empleada fuera mucho menor. Además, conforme los modelos iban siendo terminados eran mostrados en conjunto bajo *Google Earth* para comprobar que la aplicación era capaz de moverlos rápidamente, cumpliéndose así de forma correcta con la parte de los objetivos referente a los modelos 3D de los edificios e instalaciones y servicios de la zona de estudio.

En lo referente a las marcas de posición y sus ventanas de información, puede concluirse que han sido creadas de manera que su apariencia fuera lo más correcta posible con las características actuales de la aplicación y la información disponible. De este modo se han cumplido los objetivos referentes a la identificación de los nombres de los complejos urbanísticos, así como la señalización de los comercios de la zona de estudio y la aportación de información complementaria de los mismos, que aparece incluida en las ventanas de información.

Por otra parte, de la página Web cabe destacar que ha sido programada para cumplir los objetivos que tenía marcados. Así pues muestra los modelos 3D, las marcas de posición y las ventanas de información asociadas, tiene la capacidad de realizar consultas sobre los comercios de la zona y la posibilidad de reproducir vuelos que muestran diferentes partes de la zona de estudio de forma automática. Para ello fue necesario el aprendizaje en programación en XHTML, JavaScript y la API de *Google Earth*. Por último la aplicación fue puesta a disposición de un grupo reducido de usuarios que proporcionaron *feedback* informal en cuanto a usabilidad y *bugs* en distintos navegadores.

6.2. Trabajo futuro

Como trabajo futuro sería interesante modelar en 3D la otra mitad del núcleo urbano de la Playa de Puebla de Farnals, que complementa al presentado en este proyecto, así como el núcleo urbano interior, correspondiente al pueblo. De esta forma se tendría modelado todo el término municipal. Se podría establecer contacto con la

asociación de comerciantes del pueblo para tener información de todos los comercios del municipio y de esta manera crear sus marcas de posición y ventanas de información correspondientes. Así se le podría dar mayor presencia al municipio en Internet, promocionando así el turismo en la zona y su mejor conocimiento.

En cuanto al trabajo combinado de *SketchUp* y *Google Earth* sería de esperar que esta compañía desarrollara mejoras en ambas aplicaciones, dotándolas de motores gráficos mucho más potentes para que fuera posible un modelado más elaborado sin sacrificar el posterior rendimiento de los modelos en la segunda aplicación.

Por otra parte, es obvio que, dada la versatilidad de la API de *Google Earth* y la posibilidad de programar aplicaciones mucho más elaboradas, sería perfectamente posible desarrollar una que incluyera un mayor número de tipos de consultas, de forma más abierta, incluso con la capacidad de aportar nuevos datos a los ya existentes, dado que este tipo de aplicaciones se presta a ello. A su vez, con la aparición de nuevas versiones de la API, van apareciendo nuevas características que pueden ir siendo explotadas para dotar a la aplicación Web de más características.

Por último un estudio sobre el uso de la aplicación por parte de un grupo más amplio de usuarios con procedimientos más sistemáticos de recopilación de *feedback* (p.ej. cuestionarios para rellenar) permitiría incluir mejoras que en este momento pudieran no tenerse en consideración.

Capítulo 7

Bibliografía y documentación

7.1. Bibliografía

[1] Brown, Martin C. *Hacking Google Maps and Google Earth.* 1ª Ed. Indianapolis, IN. Wiley Publishing, Inc. 2006. 373 p. ISBN: 978-0-471-79009-9. <http://www.wiley.com>.

[2] Cárdenas Luque, Lola. *Curso de JavaScript.* 1ª Ed. Madrid, España. Ocio Networks S.L. 2002. 177 p. ISBN: Sin ISBN.. <http://rinconprog.metropoliglobal.com>.

[3] Carey, Rikk. Bell, Gavin. *The annotated VRML 2.0 reference manual.* 1ª Ed. Reading, MA. Addison-Wesley Developers Press. 1997. 501 p. ISBN: 0-20141-974-2. <http://www.vrml.org/Specifications/VRML97>.

[4] Chopra, Aidan. *Google SketchUp for dummies.* 1ª Ed. Hoboken, NJ. Wiley Publishing, Inc. 2007. 451 p. ISBN: 978-0-470-13744-4. <http://www.wiley.com>.

[5] Danesh, Arman. *JavaScriptTM in 10 Simple Steps or Less.* 1ª Ed. Indianapolis, IN. Wiley Publishing, Inc. 2004. 647 p. ISBN: 0-7645-4241-9. <http://www.wiley.com>.

[6] Davis, Scott. *Google Maps API, V2 Adding Where To Your Applications.* 1ª Ed. Dallas, TX. The Pragmatic Programmers. 2006. 76 p. ISBN: Sin ISBN.. <http://www.pragprog.com/titles/sdgmapi2/google-maps-api-v2>.

[7] Deitel, Harvey M. *XML : how to program.* 1ª Ed. Upper Saddle River, NJ. Prentice Hall. 2001. 934 p. ISBN: 0-13028-417-3. <http://www.prenticehall.com>.

[8] Douglas Olson, Steven. *Ajax on Java.* 1ª Ed. Sebastopol, CA. O'Reilly. 2007. 211 p. ISBN: 978-0-596-10187-9. <http://www.oreilly.com>.

[9] Dykes, Lucinda. Tittel, Ed. *XML For Dummies.* 4ª Ed. Hoboken, NJ. Wiley Publishing, Inc. 2005. 366 p. ISBN: 978-0-7645-8845-7. <http://www.wiley.com>.

[10] Elisabeth, Castro. *HTML for the World Wide Web visual quickstart guide : with XHTML and CSS.* 5ª Ed. Berkeley, CA. Peachpit Press. 2003. 480 p. ISBN: 0-321-13007-3. <http://www.peachpit.com>.

[11] Erle, Schuyler. Gibson, Rich. *Google Maps Hacks.* 1ª Ed. Sebastopol, CA. O'Reilly. 2006. 366 p. ISBN: 978-0-59-610161-9. <http://www.oreilly.com>.

[12] Flanagan, David. *JavaScript: The definitive guide.* 5ª Ed. Sebastopol, CA. O'Reilly. 2006. 1018 p. ISBN: 978-0-59-610199-2. <http://www.oreilly.com>.

[13] Goodman, Danny. *JavaScript Bible.* 1ª Ed. New York, NY. Hungry Minds Inc. 2001. 1516 p. ISBN: 0-7645-4718-6. <http://www.hungryminds.com>.

[14] Goodman, Danny. *JavaScript Examples Bible.* 1ª Ed. Indianapolis, IN. Hungry Minds Inc. 2001. 631 p. ISBN: 0-7645-4855-7. <http://www.hungryminds.com>.

[15] Goodman, Danny. *JavaScript & DHTML Cookbook.* 1ª Ed. Sebastopol, CA. O'Reilly. 2003. 540 p. ISBN: 0-596-00467-2. <http://www.oreilly.com>.

[16] Kennedy, Bill. Musciano, Chuck. *HTML & XHTML: The Definitive Guide.* 5ª Ed. Sebastopol, CA. O'Reilly. 2002. 670 p. ISBN: 0-596-00382-X. <http://www.oreilly.com>.

[17] Keogh, Jim. *Java Demystified.* 1ª Ed. New York, NY. Mc Graw Hill / Osborne. 2005. 375 p. ISBN: 978-0-07-226134-X. <http://www.mhprofessional.com/templates/112-computing.php>.

[18] Lemay, Laura. Couch, Justin. Murdock, Kelly. *3D graphics & VRML 2.0.* 1ª Ed. Indianapolis, IN. Sams Publishing. 1996. 462 p. ISBN: 1-57521-143-2. <http://www.informit.com/imprint/index.aspx?st=61091>.

[19] Matsuba, Stephen. Roehl, Bernie. *Special Edition Using VRML.* 1ª Ed. Indianapolis, IN. QUE Corporation. 1996. 768 p. ISBN: 0-7897-0494-3. <http://www.quepublishing.com>.

[20] Murdock, Kelly L. *Google SketchUp and SketchUp Pro 7 Bible.* 1ª Ed. Indianapolis, IN. Wiley Publishing, Inc. 2009. 526 p. ISBN: 978-0-470-29229-7. <http://www.wiley.com>.

[21] Pfaffenberger, Brian. Schafer, Steven M. et al. *HTML, XHTML, and CSS Bible.* 3ª Ed. Indianapolis, IN. Wiley Publishing, Inc. 2004. 790 p. ISBN: 0-7645-7718-2. <http://www.wiley.com>.

[22] Pimpler, Eric. *Google Maps API The New World of Web Mapping Version 2: Updated April 2006.* 1ª Ed. San Antonio, TX. Geospatial Training & Consulting, LLC. 2006. 44 p. ISBN: Sin ISBN.. <http://www.geospatialtraining.com>.

[23] Posadas, Marino. *XML: Introducción al lenguaje.* 1ª Ed. Madrid, España. Grupo Eidos. 2000. 85 p. ISBN: 84-88457-02-2. <http://www.eidos.es>.

[24] Purvis, Michael. Sambells, Jeffrey. Turner, Cameron. *Beginning Google Maps Applications with PHP and Ajax: From Novice to Professional.* 1ª Ed. Berkeley, CA. Apress. 2006. 358 p. ISBN: 978-1-59059-707-1. <http://www.apress.com>.

[25] Roskes, Bonnie. *The SketchUp book.* 5ª Ed. West Caldwell, NJ. Conceptual Product Development, Inc. 2005. 492 p. ISBN: Sin ISBN.. <http://www.docwalt.com>.

[26] Rusty Harold, Elliotte. *XML 1.1 Bible.* 3ª Ed. Indianapolis, IN. Wiley Publishing, Inc. 2004. 1022 p. ISBN: 0-7645-4986-3. <http://www.wiley.com>.

[27] Skonnard, Aaron. Gudgin, Martin. *Essential XML quick reference : a programmer's reference to XML, XPath, XSLT, XML Schema, SOAP, and more.* 1ª Ed. Boston, Lincolnshire. Addison-Wesley. 2002. 402 p. ISBN: 0-20174-095-8. <http://www.pearson.com>.

[28] Tennison, Jeni. *Beginning XSLT 2.0 : from novice to professional.* 1ª Ed. Berkeley, CA. Apress. 2005. 797 p. ISBN: 1-59059-324-3. <http://www.apress.com>.

[29] Tittel, Ed. Noble, Jeff. *HTML, XHTML & CSS For Dummies.* 6ª Ed. Indianapolis, IN. Wiley Publishing, Inc. 2008. 384 p. ISBN: 978-0-470-23847-9. <http://www.wiley.com>.

[30] Williams, Kevin et al. *Professional XML Databases.* 1ª Ed. Birmingham, UK. Wrox Press. 2000. 1007 p. ISBN: 1-86100-358-7. <http://www.wrox.com>.

[31] Young, Michael. *Google Maps Mashups with Google Mapplets.* 1ª Ed. Berkeley, CA. Apress. 2008. 116 p. ISBN: 978-1-4302-0995-9. <http://www.apress.com>.

7.2. Direcciones de Internet

[32] A. Frenkel, Karen. Therapists Use Virtual Worlds to Address Real Problems. Scientific American. Disponible en Web: <http://www.scientificamerican.com/article.cfm?id=therapists-use-virtual-worlds>. [Consulta: 15 de febrero de 2010].

[33] A. Richard, Glenn et al. Using Google Earth to Study Global Concerns. Earth Science Educational Resource Center. Disponible en Web: <http://www.eserc.stonybrook.edu/GEO311/GoogleEarth/index.html>. [Consulta: 5 de abril de 2010].

[34] Abraham, Beth. Virtual Rehab. Beth Abraham Family of health services. Disponible en Web: <http://www.bethabe.org/Virtual_Rehab325.html>. [Consulta: 16 de febrero de 2010].

[35] Adabeltriellis. Cybertown. Wordpress. Disponible en Web: <http://adabeltriellis.wordpress.com/2009/10/01/cybertown/>. [Consulta: 16 de febrero de 2010].

[36] Advisory Group on Computer Graphics. VRML Browsers. Advisory Group on Computer Graphics. Disponible en Web: <http://www.agocg.ac.uk/train/vrml2rep/part1/guide2.htm>. [Consulta: 16 de febrero de 2010].

[37] Alternative Fuel Systems. Gaseous Fuel Injectors. Alternative Fuel Systems. Disponible en Web: <http://www.afsglobal.com/products/gaseous-fuel-injectors.html>. [Consulta: 23 de febrero de 2010].

[38] Álvarez, Miguel Ángel. Marcar o desmarcar todos los checkbox de un formulario con Javascript. desarrolloweb.com. Disponible en Web: <http://www.desarrolloweb.com/articulos/2291.php>. [Consulta: 2 de marzo de 2009].

[39] Ames, Andrew. Radiosity Primer. Jack Baskin School of Engineering. University of California, Santa Cruz. Disponible en Web: <http://www.soe.ucsc.edu/classes/cmps161/Winter04/projects/aames/index.htm>. [Consulta: 25 de febrero de 2010].

[40] Andaur, Claudio et al. Volumetric Effects. Blender's SoftBody System. Disponible en Web: <http://download.blender.org/documentation/htmlI/ch24.html>. [Consulta: 25 de febrero de 2010].

[41] Andersson, Mikael. Advanced 3D for games: workflow progress. FlatPress. Disponible en Web: <http://blogs.gscept.com/2008/ip5/index.php/2008/index.html>. [Consulta: 23 de febrero de 2010].

[42] ArcGIS Desktop Help. Plate Carrée. Environmental Systems Research Institute, Inc. Disponible en Web: <http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=Plate_Carree>. [Consulta: 30 de julio de 2009].

[43] Ardalani, Sarah et al. Informe de homicidios en Los Ángeles. CA mapa de homicidios. Los Angeles Times. Disponible en Web: <http://projects.latimes.com/homicide-report/map/>. [Consulta: 25 de marzo de 2010].

[44] Autodesk. Pouring Water with nParticles. Autodesk Inc. Disponible en Web: <http://area.autodesk.com/blogs/duncan/pouring_water_with_nparticles>. [Consulta: 24 de febrero de 2010].

[45] B. Antón. El diseñador que pinta Galicia en 3D en Internet. La Voz de Galicia. Disponible en Web: <http://www.lavozdegalicia.com/sociedad/2010/05/26/0003_8508218.htm>. [Consulta: 1 de junio de 2010].

[46] Bfree News. Bfree News. Atman Systems. Disponible en Web: <http://www.bfreenews.com/bfn/home?n=1>. [Consulta: 6 de mayo de 2010].

[47] Bing. Bing Maps. Microsoft. Disponible en Web: <http://www.bing.com/maps/>. [Consulta: 22 de marzo de 2010].

[48] Bitmanagement. BS Contact. Bitmanagement Software GmbH. Disponible en Web: <http://www.bitmanagement.com>. [Consulta: 16 de febrero de 2010].

[49] Blender. Blender 3D. Blender. Disponible en Web: <http://www.blender.org/>. [Consulta: 16 de febrero de 2010].

[50] Brown, Shays. The Studio Enviroment. Design Droplets. Disponible en Web: <http://designdroplets.com/articles/product-visualisation-studio-environment-vol2/>. [Consulta: 25 de febrero de 2010].

[51] Brutzman, Don. Borsky, Mirek et al. The Xj3D project. Web 3D Consortium. Disponible en Web: <http://www.xj3d.org/>. [Consulta: 16 de febrero de 2010].

[52] Buttussi F., Chittaro L., Nadalutti D. H-Animator. Human-Computer Interaction Laboratory. Disponible en Web: <http://hcilab.uniud.it/h-animator/>. [Consulta: 16 de febrero de 2010].

[53] Callieri, Marco. Cignoni, Paolo et al. Pushing Time-of-Flight scanners to the limit. Visual Computing Lab. Istituto di scienza e tecnologie dell'informazione "A. Faedo". Disponible en Web: <http://vcg.isti.cnr.it/Publications/2009/CCDS09/>. [Consulta: 16 de febrero de 2010].

[54] COLLADA Working Group. COLLADA - Digital Asset and FX Exchange Schema. Khronos.org. Disponible en Web: <https://collada.org/mediawiki/index.php/Main_Page>. [Consulta: 30 de septiembre de 2008].

[55] Cortona Group. Cortona3D Viewer. Parallel Graphics Co. Disponible en Web: <http://www.cortona3d.com/cortona>. [Consulta: 16 de febrero de 2010].

[56] Cortona Group. Cortona VRML Client 5.1. Wareseeker. Disponible en Web: <http://wareseeker.com/screenshot/cortona-vrml-client-5.1.exe/366810>. [Consulta: 16 de febrero de 2010].

[57] Damian Yerrick et al. Bezier curve. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/Bezier_curves>. [Consulta: 25 de marzo de 2010].

[58] De la Flor, Mike. The Carrara Studio. Subdivision surfaces. webreference.com. Disponible en Web: <http://www.webreference.com/3d/cararra/3.html>. [Consulta: 23 de febrero de 2010].

[59] Departmente of Physics. Diffraction Grating. Georgia State University. Disponible en Web: <http://hyperphysics.phy-astr.gsu.edu/hbase/phyopt/grating.html>. [Consulta: 2 de marzo de 2010].

[60] Dirección General del Catastro. La cartografía catastral en internet. Dirección General del Catastro. Disponible en Web: <http://www.catastro.meh.es/servicios/wms/wms.htm>. [Consulta: 30 de julio de 2009].

[61] Dlagnekov, Louka. Final Rendering Project. Computer Graphics Lab. University of California. Disponible en Web: <http://graphics.ucsd.edu/courses/rendering/2005/ldlagnekov/index.html>. [Consulta: 25 de febrero de 2010].

[62] Dysprosia. Level set. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/Level_set>. [Consulta: 23 de febrero de 2010].

[63] Edward Lyme. Pr. Dr. Non-Photorealistic Rendering. Plaid Creature. Disponible en Web: <http://www.plaidcreature.com/2009/06/11/non-photorealistic-rendering-101/>. [Consulta: 25 de febrero de 2010].

[64] El Mundo. El Bosque Modelo de Urbión, en la provincia de Soria, podrá visitarse de manera virtual. El Mundo. Disponible en Web: <http://www.elmundo.es/elmundo/2010/05/28/castillayleon/1275036481.html>. [Consulta: 5 de junio de 2010].

[65] Elmer. Keyhole 3D Earth viewers. SYIX. Disponible en Web: <http://www.syix.com/elmer/Cool%20Stuff%203/EarthViewer%203D.htm>. [Consulta: 5 de abril de 2010].

[66] European Virtual Engineering. Virtual engineering. European Virtual Engineering Technological Centre. Disponible en Web: <http://w3.euve.org:81/en/cad.asp>. [Consulta: 16 de febrero de 2010].

[67] Exchange 3D. 3D models and 3D Grpahics in Exchange 3D. Exchange3D USA. Disponible en Web: <http://www.exchange3d.com/cubecart/index.php>. [Consulta: 24 de febrero de 2010].

[68] F. White, Stephen. White Dune VRML editor. Stuttgart University. Disponible en Web: <http://vrml.cip.ica.uni-stuttgart.de/dune/>. [Consulta: 16 de febrero de 2010].

[69] Fakultät für Maschinenbau. Maschinenbau der Uni Paderborn in TOP-Position – Platz 4 von 30 Maschinenbaufakultäten an deutschen Universitäten. Universität Paderborn. Disponible en Web: <http://www.uni-paderborn.de/mitteilung/2123/>. [Consulta: 16 de febrero de 2010].

[70] Fong, Julian et al. Rendering (computer graphics). Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/Rendering_(computer_graphics)>. [Consulta: 15 de mayo de 2009].

[71] Fraunhofer Institut Graphische Datenverarbeitung. Instant Player. Federal Ministry of Education and Research. Disponible en Web: <http://www.instantreality.org/home/>. [Consulta: 16 de febrero de 2010].

[72] Funnell, W. Robert J. Dynamic Anatomy Visualization in 3-D. Auditory Mechanics Laboratory. McGill University. Disponible en Web: <http://audilab.bmed.mcgill.ca/~funnell/davis3d/>. [Consulta: 16 de febrero de 2010].

[73] García Rojas, Alejandro. Ontology for virtual humans. Vitual Reality Lab. Ecole Polytechnique Fédérale de Lausanne. Disponible en Web: <http://vrlab.epfl.ch/~alegarcia/VHOntology/long.html>. [Consulta: 24 de febrero de 2010].

[74] Giger, Matt. KML, libkml and the "standard" mistake. EarthBrowser blog. Disponible en Web: <http://blog.earthbrowser.com/2008/04/kml-libkml-and-standard-mistake.html>. [Consulta: 26 de abril de 2010].

[75] Gilbertson, Scott. Google's Keyhole Mapping Language is Now an Open Standard. Webmonkey. Disponible en Web: <http://www.webmonkey.com/2008/04/google_s_keyhole_mapping_language_is_now_an_open_standard/>. [Consulta: 26 de abril de 2010].

[76] Goesele, Michael et al. Modeling and Rendering of Translucent Materials. Max Planck Institut. Disponible en Web: <http://www.mpi-inf.mpg.de/departments/d4/areas/scenedigitization/>. [Consulta: 25 de febrero de 2010].

[77] González, Antonio. ETRS89. Wikipedia. Disponible en Web: <http://es.wikipedia.org/wiki/ETRS89>. [Consulta: 30 de marzo de 2010].

[78] Google. Google Earth. Google. Disponible en Web: <http://earth.google.com>. [Consulta: 10 de septiembre 2008].

[79] Google Code. Registro en el API de Google Maps. Google. Disponible en Web: <http://code.google.com/intl/es-ES/apis/maps/signup.html>. [Consulta: 25 de marzo de 2010].

[80] Google Code. API de Google Maps. Google. Disponible en Web: <http://code.google.com/intl/es/apis/maps/index.html>. [Consulta: 8 de julio de 2009].

[81] Google Code. Google Earth Plug-in. Google. Disponible en Web: <http://dl.google.com/earth/plugin/GoogleEarthPluginSetup_en.exe>. [Consulta: 11 de noviembre de 2009].

[82] Google Crisis Response. Gulf of Mexico Oil Spill. Google. Disponible en Web: <http://www.google.com/crisisresponse/oilspill/>. [Consulta: 5 de abril de 2010].

[83] Google Crisis Response. Apoya los esfuerzos de ayuda en Haití. Google. Disponible en Web: <http://www.google.com/intl/es/relief/haitiearthquake/>. [Consulta: 25 de enero de 2010].

[84] Google Labs. Google Labs. Google. Disponible en Web: <http://labs.google.com>. [Consulta: 8 de julio de 2009].

[85] Google Maps. Google Maps. Google. Disponible en Web: <http://maps.google.com>. [Consulta: 25 de marzo de 2010].

[86] Google Press Center. Bringing the Power of Google Earth to the Browser. Google Press Center. Disponible en Web: <http://www.google.com/intl/en/press/annc/earthapi_20080528.html>. [Consulta: 5 de marzo de 2009].

[87] Google SketchUp. Google 3d Warehouse. Google. Disponible en Web: <http://sketchup.google.com/3dwarehouse/>. [Consulta: 24 de febrero de 2010].

[88] Google. KML Reference. Google Code. Disponible en Web: <http://code.google.com/intl/en-EN/apis/kml/documentation/kmlreference.html>. [Consulta: 29 de julio de 2008].

[89] Google. Google AJAX APIs Developer's Guide. Google Code. Disponible en Web: <http://code.google.com/intl/en-EN/apis/ajax/documentation/>. [Consulta: 29 de julio de 2008].

[90] Google. Google Earth API Reference. Google Code. Disponible en Web: <http://code.google.com/intl/es/apis/earth/documentation/reference/index.html>. [Consulta: 29 de julio de 2008].

[91] Google. Google Earth Plug-in / Interactive Samples. Google Code. Disponible en Web: <http://earth-api-samples.googlecode.com/svn/trunk/demos/interactive/index.html>. [Consulta: 10 de septiembre de 2008].

[92] Google. Google Earth API. Google Code. Disponible en Web: <http://code.google.com/intl/en-EN/apis/earth/>. [Consulta: 10 de septiembre de 2008].

[93] Google. Parsing a KML file with Google Earth Plug-in. Google Groups. Disponible en Web: <http://groups.google.es/group/google-earth-browser-plugin/browse_thread/thread/4f82989a23e584c2#>. [Consulta: 2 de marzo de 2009].

[94] Google. Google Maps API - How to handle two instances. Google Groups. Disponible en Web: <http://groups.google.com/group/Google-Maps-API/browse_thread/thread/f4a73ffc0843e756#>. [Consulta: 15 de marzo de 2009].

[95] Google. Multiple Google Earth. Google Code. Disponible en Web: <http://earth-api-samples.googlecode.com/svn/trunk/demos/multiple/index.html>. [Consulta: 25 de junio de 2009].

[96] Google. Google Earth Plug-in Driving Simulator. Google Code. Disponible en Web: <http://earth-api-samples.googlecode.com/svn/trunk/demos/drive-simulator/index.html>. [Consulta: 30 de julio de 2009].

[97] Google. google.earth Namespace Reference. Google Code. Disponible en Web: <http://code.google.com/intl/es/apis/earth/documentation/reference/google_earth_namespace.html>. [Consulta: 11 de octubre de 2009].

[98] Google. Google Earth COM API Documentation. Google Earth. Disponible en Web: <http://earth.google.com/comapi/>. [Consulta: 11 de octubre de 2009].

[99] Google. Google Earth Fusion. Google. Disponible en Web: <http://www.google.com/intl/es_ALL/enterprise/earthmaps/earth_fusion.html>. [Consulta: 5 de abril de 2010].

[100] GPS Explorer Ltda. Interfaz de Google Earth. GPS Explorer Ltda. Disponible en Web: <http://www.gpsexplorer.cl/capacitaciones.html>. [Consulta: 5 de abril de 2010].

[101] Guinot, Jerome. Javascript Depth of Field Effect. Geeks3D. Disponible en Web: <http://www.geeks3d.com/20090710/javascript-depth-of-field-effect/>. [Consulta: 25 de febrero de 2010].

[102] Hanke, Hynek et al. Global illumination. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/Global_illumination>. [Consulta: 25 de febrero de 2010].

[103] Hanna, Ramy. Transparent Alpha Channel. 3DS Max Rendering. Disponible en Web: <http://3dsmaxrendering.blogspot.com/2010/01/transparent-alpha-channel.html>. [Consulta: 25 de febrero de 2010].

[104] Hardy, Michael et al. Constructive solid geometry. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/Constructive_solid_geometry>. [Consulta: 23 de febrero de 2010].

[105] Hare, James et al. Wikipedia:WikiProject Pharmacology. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Pharmacology>. [Consulta: 23 de febrero de 2010].

[106] Hill, Gary. 30 Graphics 3D: Introduction to Java 3D. School of Science and Technology, University of Northampton. Disponible en Web: <http://www.computing.northampton.ac.uk/~gary/csy3019/CSY3019SectionD.html>. [Consulta: 23 de febrero de 2010].

[107] Histourist. Histourist. Histourist. Disponible en Web: <http://www.histourist.com/home>. [Consulta: 6 de mayo de 2010].

[108] Human-Computer Interaction Laboratory. MobiX3D Player. University of Udine. Disponible en Web: <http://hcilab.uniud.it/MobiX3D/>. [Consulta: 16 de febrero de 2010].

[109] Humusoft. Orbisnap. Humusoft. Disponible en Web: <http://www.orbisnap.com>. [Consulta: 16 de febrero de 2010].

[110] Huskey, W. Phillip. Polarización LCD. Rutgers-Newark: The State University of New Jersey. Disponible en Web: <http://andromeda.rutgers.edu/~huskey/335f09_lec.html>. [Consulta: 2 de marzo de 2010].

[111] Imageshack. Google Earth Client (Enterprise). Imageshack. Disponible en Web: <http://img73.imageshack.us/i/dell2bearth2benterprisezl6.jpg/>. [Consulta: 5 de abril de 2010].

[112] Institute for Information Technology. Demotride. National Research Council Canada. Disponible en Web: <http://www.demotride.com/>. [Consulta: 16 de febrero de 2010].

[113] Instituto Cartográfico Valenciano. Insituto Cartográfico Valenciano. Generalitat Valenciana. Disponible en Web: <http://www.icv.gva.es/ICV/>. [Consulta: 26 de febrero de 2009].

[114] Instituto Geográfico Nacional. ETRS89. Instituto Geográfico Nacional. Disponible en Web: <http://www.fomento.es/MFOM/LANG_CASTELLANO/DIRECCIONES_GENERALES/INSTITUTO_GEOGRAFICO/Geodesia/red_geodesicas/etrs89.htm>. [Consulta: 30 de julio de 2009].

[115] Ivan Romero et al. Teorema de muestreo de Nyquist-Shannon. Wikipedia. Disponible en Web: <http://es.wikipedia.org/wiki/Teorema_de_muestreo_de_Nyquist-Shannon>. [Consulta: 26 de febrero de 2010].

[116] Java.net. Java3D VRML Loader. Oracle. Disponible en Web: <https://j3d-vrml97.dev.java.net/>. [Consulta: 16 de febrero de 2010].

[117] Johnson, Colin. Computer Visualisation of Dudley Castle c1550. Exrenda. Disponible en Web: <http://www.exrenda.net/dudley/index.htm>. [Consulta: 16 de febrero de 2010].

[118] Kamburelis, Michalis. View3dscene. Sourceforge. Disponible en Web: <http://vrmlengine.sourceforge.net/view3dscene.php>. [Consulta: 16 de febrero de 2010].

[119] Kerk, Justin et al. 3D modeling. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/3D_modeling>. [Consulta: 15 de mayo de 2009].

[120] Kirkpatrik, Marshall. Google Gives Up Control of Keyhole Markup Language (KML). Read Write Web. Disponible en Web: <http://www.readwriteweb.com/archives/google_gives_up_control_over_k.php>. [Consulta: 26 de abril de 2010].

[121] Konno, Satoshi. CyberX3D. Sourceforge. Disponible en Web: <http://sourceforge.net/projects/cx3dcc/>. [Consulta: 16 de febrero de 2010].

[122] Kubíček, Lukáš. Virtuální realita. Provozně Ekonomická Fakulta Mendelova Univerzita. Disponible en Web: <https://akela.mendelu.cz/~xkubice6/Etechnologie/projekt/virtualni_realita_historie.xhtml>. [Consulta: 7 de junio de 2010].

[123] Kulla, Christopher. SunFlow. Sourceforge. Disponible en Web: <http://sunflow.sourceforge.net/index.php?pg=gall>. [Consulta: 23 de febrero de 2010].

[124] KxCAD. Motion Blur with the mental ray Renderer. Beijing Innovative Linkage Technology. Disponible en Web: <http://www.kxcad.net/autodesk/3ds_max/Autodesk_3ds_Max_9_Reference/motion_blur_with_the_mental_ray_renderer.html>. [Consulta: 25 de febrero de 2010].

[125] Le Hégaret, Philippe et al. What is the Document Object Model? World Wide Web Consortium. Disponible en Web: <http://www.w3.org/TR/DOM-Level-3-Core/introduction.html>. [Consulta: 9 de abril de 2010].

[126] Lighttek Software. Alteros 3D. Lighttek Software. Disponible en Web: <http://www.lighttek.com/alteros/>. [Consulta: 16 de febrero de 2010].

[127] Loisel, Sébastien. Bump mapping. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/Bump_mapping>. [Consulta: 25 de febrero de 2010].

[128] López, Germán. Playa Puebla de Farnals. Universitat de València. Disponible en Web: <http://mural.uv.es/gerlodo/index.html>. [Consulta: 26 de mayo de 2010].

[129] Lukka, Tuomas J. Stewart, John et al. FreeWRL. Sourceforge. Disponible en Web: <http://freewrl.sourceforge.net/>. [Consulta: 16 de febrero de 2010].

[130] Mapquest. Mapquest. Map Quest Inc. Disponible en Web: <http://www.mapquest.com>. [Consulta: 25 de marzo de 2010].

[131] Mark Pesce. Mark Pesce Carreer. Mark Pesce. Disponible en Web: <http://markpesce.com/career.html>. [Consulta: 5 de junio de 2009].

[132] Marks, Mano et al. Google Earth API Samples - HTML Div Balloons. Google Project Hosting. Disponible en Web: <http://earth-api-samples.googlecode.com/svn/trunk/examples/balloon-div.html>. [Consulta: 29 de julio de 2008].

[133] Martín, David et al. Huerta Norte. Wikipedia. Disponible en Web: <http://es.wikipedia.org/wiki/Huerta_Norte>. [Consulta: 26 de mayo de 2010].

[134] Maury Markowitz et al. Non-uniform rational B-spline. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/Nurbs>. [Consulta: 25 de marzo de 2010].

[135] Mayer, Daniel et al. Texture mapping. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/Texture_mapping>. [Consulta: 25 de febrero de 2010].

[136] McDaniel, Braden. OpenVRML. Sourceforge. Disponible en Web: <http://openvrml.sourceforge.net/>. [Consulta: 16 de febrero de 2010].

[137] McPherson, Shaun et al. 3D computer graphics software. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/3D_computer_graphics_software>. [Consulta: 15 de mayo de 2009].

[138] Minnus. Minnus. Minnus. Disponible en Web: <http://www.minnus.com.ar/index.php?id=>. [Consulta: 6 de mayo de 2010].

[139] Moody, Niall. Heilan X3D Browser. Centre for Music Technology, University of Glasgow. Disponible en Web: <http://www.niallmoody.com/heilan/>. [Consulta: 16 de febrero de 2010].

[140] Mostert, Rogier. Rendering en ray-tracing. Mostert. Disponible en Web: <http://www.mostert.org/3dindepraktijk/rendering_en_raytracing.php>. [Consulta: 25 de febrero de 2010].

[141] Myriad. Myriad 3D Reader. Informative Graphics Corporation. Disponible en Web: <http://www.myriadviewer.com/myriadreader.htm>. [Consulta: 16 de febrero de 2010].

[142] National Institute of Standards and Technology. VRML Plugin and Browser Detector. U.S.Department of Commerce. Disponible en Web: <http://cic.nist.gov/vrml/vbdetect.html>. [Consulta: 25 de septiembre de 2008].

[143] National Institute of Standards and Technology. Cosmo Player VRML Plugin. U.S. Department of Commerce. Disponible en Web: <http://cic.nist.gov/vrml/cosmoplayer.html>. [Consulta: 16 de febrero de 2010].

[144] National September 11 Memorial & Museum. National September 11 Memorial. National September 11 Memorial & Museum. Disponible en Web: <http://www.national911memorial.org>. [Consulta: 5 de junio de 2010].

[145] National Weather Service Internet Services Team. National Weather Service. (NOAA). National Oceanic and Atmospheric Administration. Disponible en Web: <http://forecast.weather.gov>. [Consulta: 25 de marzo de 2010].

[146] New Dimension. Reverse Engineering. New Dimension Systems Co. Ltd. Disponible en Web: <http://www.newdimchina.com/expertise/reverse_engineering.html>. [Consulta: 23 de febrero de 2010].

[147] Octaga. Octaga Player. Octaga. Disponible en Web: <http://www.octaga.com>. [Consulta: 16 de febrero de 2010].

[148] OGP Surveying and Positioning Committee. EPSG Geodetic Parameter Registry. OGP Surveying and Positioning Committee. Disponible en Web: <http://www.epsg-registry.org/>. [Consulta: 30 de marzo de 2010].

[149] Open Geospatial Consortium Inc. KML Specification. Open Geospatial Consortium Inc. Disponible en Web: <http://www.opengeospatial.org/standards/kml/>. [Consulta: 21 de septiembre de 2009].

[150] Open Wonderland Foundation. Project Wonderland. Oracle. Disponible en Web: <http://www.projectwonderland.com/>. [Consulta: 16 de febrero de 2010].

[151] Pablo d'Angelo et al. Hugin. Sourceforge. Disponible en Web: <http://hugin.sourceforge.net/>. [Consulta: 25 de septiembre de 2008].

[152] Patrick, James et al. Shading. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/Shading>. [Consulta: 25 de febrero de 2010].

[153] Pelayo, Vicente et al. Puebla de Farnals. Wikipedia. Disponible en Web: <http://es.wikipedia.org/wiki/Puebla_de_Farnals>. [Consulta: 26 de mayo de 2010].

[154] Pelayo, Vicente et al. Información turismo de Platja Pobla de Farnals. CasaSpain. Disponible en Web: <http://www.casaspain.com/wesp/infotur.asp?loc=platja%20pobla%20de%20farnals>. [Consulta: 26 de mayo de 2010].

[155] Pinecoast Software. SwirlX3D editor. Pinecoast Software, Inc. Disponible en Web: <http://www.pinecoast.com/swirl3d.htm>. [Consulta: 16 de febrero de 2010].

[156] Pixar. Pixar's Renderman. Pixar. Disponible en Web: <https://renderman.pixar.com/>. [Consulta: 30 de marzo de 2010].

[157] Pobla Marina. Pobla Marina. Pobla Marina. Disponible en Web: <http://www.poblamarina.es/>. [Consulta: 26 de mayo de 2010].

[158] Publietur S.L. Playa Puebla de Farnals. Asociación de comerciantes Virgen del Carmen. Disponible en Web: <http://www.playapuebladefarnals.com/>. [Consulta: 26 de mayo de 2010].

[159] R. Holloway, Dennis. Virtual Reality Archaeology. Dennis R. Holloway Architect. Disponible en Web: <http://www.dennisrhollowayarchitect.com/PreContact.html>. [Consulta: 16 de febrero de 2010].

[160] R. Vance, Joseph. Particle Art. josephrvance.com. Disponible en Web: <http://josephrvance.com/particles.html>. [Consulta: 24 de febrero de 2010].

[161] Raktarok. FFXIII: VERSUS NEW INFO! Gametrailers.com. Disponible en Web: <http://forums.gametrailers.com/thread/ffxiii--versus-new-info-------/986558?page=4>. [Consulta: 24 de febrero de 2010].

[162] Real Skills. Real Skills. Excel Driving School LLC. Disponible en Web: <http://www.real-skills.net/simulators.html>. [Consulta: 16 de febrero de 2010].

[163] Render Plus. Reflection. Render Plus Systems Inc. Disponible en Web: <http://www.renderplus.com/wk/Reflection_w.htm>. [Consulta: 25 de febrero de 2010].

[164] Right Hemisphere Inc. Deep View. Right Hemisphere Inc. Disponible en Web: <http://www.righthemisphere.com/products/client-products/deep-view>. [Consulta: 16 de febrero de 2010].

[165] Rigmarole, Ross. Recap; Side Effects Houdini Apprentice HD. Blogspot. Disponible en Web: <http://ross-rigmarole.blogspot.com/2008/10/recap-side-effects-houdini-apprentice.html>. [Consulta: 25 de febrero de 2010].

[166] Rob Hooft et al. X3D. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/X3D>. [Consulta: 16 de febrero de 2010].

[167] Roskes, Bonnie. SketchUp Geometry Project of the month. Math Forum Drexel University. Disponible en Web: <http://mathforum.org/sketchup/projects_of_the_month.html>. [Consulta: 23 de febrero de 2010].

[168] Ross, Brian. COSC 3P98 Animations - 2002. Brock Computer Science. Disponible en Web: <http://www.cosc.brocku.ca/Offerings/3P98/2002/>. [Consulta: 16 de febrero de 2010].

[169] Schutzberg, Adena. KML Now an OGC Standard: World Changing? All Points Blog. Disponible en Web: <http://apb.directionsmag.com/archives/4185-KML-Now-an-OGC-Standard-World-Changing.html>. [Consulta: 26 de abril de 2010].

[170] Seer. World Geodetic System. Google Earth Community Forums. Disponible en Web: <http://bbs.keyhole.com/ubb/ubbthreads.php?ubb=showthreaded&Number=30500&site_id=1#import>. [Consulta: 30 de julio de 2009].

[171] Silicon Graphics International. SGI - Developer Central Open Source. Silicon Graphics International. Disponible en Web: <http://oss.sgi.com/projects/inventor/>. [Consulta: 15 de septiembre de 2009].

[172] Silva, David. 3D Total Maya Tutorial. 3DTotal.com. Disponible en Web: <http://www.3dtotal.com/ffa/tutorials/maya/tut/mazertut1_main.htm>. [Consulta: 23 de febrero de 2010].

[173] Singh, Raj. KML now an OGC Open Standard. rajsingh.org blog. Disponible en Web: <http://www.rajsingh.org/blog/2008/04/15/kml-now-an-ogc-open-standard/>. [Consulta: 26 de abril de 2010].

[174] Smith, Geoffrey. ED50. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/ED50>. [Consulta: 30 de marzo de 2010].

[175] Smith, Jamie. Digital Representation & Programming. NewCastle Digital Media. Disponible en Web: <http://dm.ncl.ac.uk/blog/basic-techniques-week-1-digital-representation-programming>. [Consulta: 25 de febrero de 2010].

[176] Sons, Kristian. E. Ramey, Larry et al. X3D Tool Kit. Sourceforge. Disponible en Web: <http://sourceforge.net/projects/x3dtoolkit/>. [Consulta: 16 de febrero de 2010].

[177] Stannard, Aaron. CERN'S Large Hadron Collider Portrayed via Google Maps Overlay in SmartDraw 2009. SmartDraw. Disponible en Web: <http://blog.smartdraw.com/archive/2008/09/09/cern-s-large-hadron-collider-portrayed-via-google-maps-overlay-in-smartdraw-2009.aspx>. [Consulta: 25 de marzo de 2010].

[178] Strickland, Jonathan. How Virtual Reality Works. HowStuffWorks. Disponible en Web: <http://electronics.howstuffworks.com/gadgets/other-gadgets/virtual-reality6.htm>. [Consulta: 15 de febrero de 2010].

[179] The Anome. World Geodetic System. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/World_Geodetic_System>. [Consulta: 30 de marzo de 2010].

[180] The JavaView Project. JavaView. Textures. The JavaView Project. Disponible en Web: <http://www.javaview.de/doc/demo/textures.html>. [Consulta: 24 de febrero de 2010].

[181] Timonen, Ville. 3D engine. Wili.cc. Disponible en Web: <http://wili.cc/softography/3d-engine/>. [Consulta: 23 de febrero de 2010].

[182] Trice, Andrew. Explorign the Google Earth API. O'Reilly InsideRIA. Disponible en Web: <http://www.insideria.com/2008/08/exploring-the-google-earth-api.html>. [Consulta: 30 de agosto de 2008].

[183] Uralsky, Yury. Efficient Soft-Edged Shadows Using Pixel Shader Branching. Nvidia. Disponible en Web: <http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter17.html>. [Consulta: 25 de febrero de 2010].

[184] Varios. La pobla de Farnals. Valencians.com. Disponible en Web: <http://www.valencians.com/valencia/hn/pobladefarnals/>. [Consulta: 26 de mayo de 2010].

[185] View Life. Webseed 3D studio. View Life. Disponible en Web: <http://www.viewlife.com/Web3D/Webseed-Studio-real-time-3D.htm>. [Consulta: 25 de septiembre de 2008].

[186] W. Peters, Arno. Mercator projection. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/Mercator_projection>. [Consulta: 30 de marzo de 2010].

[187] W3C HTML Working Group. XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition). World Wide Web Consortium. Disponible en Web: <http://www.w3.org/TR/xhtml1/>. [Consulta: 26 de noviembre de 2009].

[188] W3Schools. HTML 4.01 / XHTML 1.0 Reference. Refsnes Data. Disponible en Web: <http://www.w3schools.com/TAGS/>. [Consulta: 19 de febrero de 2010].

[189] W3Schools. XML Applications. Refsnes Data. Disponible en Web: <http://www.w3schools.com/xml/xml_applications.asp>. [Consulta: 2 de marzo de 2009].

[190] W3Schools. JavaScript Tutorial. Refsnes Data. Disponible en Web: <http://www.w3schools.com/JS/default.asp>. [Consulta: 5 de marzo de 2009].

[191] Wald, Ingo. Benthin, Carsten. Slusallek, Philipp. Interactive Ray Tracing of Dynamic Scenes. Computer Graphics Group. Saarland University. Disponible en Web: <http://www.openrt.de/Gallery/2002_DynamicRayTracing/>. [Consulta: 25 de febrero de 2010].

[192] Web 3D Consortium. Web3D Consortium. Web 3D Consortium. Disponible en Web: <http://www.web3d.org/>. [Consulta: 25 de marzo de 2010].

[193] Web 3D Consortium. VRML97 and Related Specifications. Web 3D Consortium. Disponible en Web: <http://www.web3d.org/x3d/specifications/vrml/>. [Consulta: 25 de marzo de 2010].

[194] Web 3D Consortium. What is X3D. Web 3D Consortium. Disponible en Web: <http://www.web3d.org/about/overview/>. [Consulta: 25 de septiembre de 2008].

[195] Wieser, Manuel. Leila – Facial Motion Capture. ManuelWieser.com. Disponible en Web: <http://www.manuelwieser.com/>. [Consulta: 23 de febrero de 2010].

[196] Winkelholz, Carsten. Weiss, Malte et al. Open ActiveWrl. Sourceforge. Disponible en Web: <http://open-activewrl.sourceforge.net/>. [Consulta: 16 de febrero de 2010].

[197] Wolinsky, Cary. Virtual Surgery, NASA Ames Research Center, California, 2001. National Geographic Society. Disponible en Web: <http://photography.nationalgeographic.com/photography/enlarge/virtual-surgery-wolinsky_pod_image.html>. [Consulta: 16 de febrero de 2010].

[198] Woodford, Chris. Explain that stuff! Virtual Reality. Explain that stuff! Disponible en Web: <http://www.explainthatstuff.com/virtualreality.html>. [Consulta: 16 de febrero de 2010].

[199] Woratek. Vista de Manhattan en Google Earth. Woratek-Revista de Tecnología y Adelantos. Disponible en Web: <http://www.woratek.com/2009/04/14/descargar-google-earth-50/>. [Consulta: 5 de abril de 2010].

[200] World Wide Web Consortium (W3C). W3C Markup Validation Service. World Wide Web Consortium (W3C). Disponible en Web: <http://validator.w3.org/>. [Consulta: 26 de febrero de 2010].

[201] Wycisk, Peter. Pr. Dr. Mehr als nur ein Bild. Martin-Luther-Universität Halle-Wittenberg. Department Hydro- and Environmental Geology. Disponible en Web: <http://www.3d-geology.de/how_this_way/3d-geologie/>. [Consulta: 23 de febrero de 2010].

[202] X. Chen, Jim. Liu, Yanling. Yang, Lin. Virtual human anatomy. Department of computer science. George Mason University. Disponible en Web: <http://cs.gmu.edu/~jchen/exhibit.html>. [Consulta: 16 de febrero de 2010].

[203] Yang, Baoguang et al. Packet-based Hierarchal Soft Shadow Mapping. Iparla Project. Disponible en Web: <http://iparla.labri.fr/publications/2009/YFGL09/>. [Consulta: 25 de febrero de 2010].

[204] Zimbabwe Civil Action Support Group. Mapa elecciones Zimbabwe. Sokwanele. Disponible en Web: <http://www.sokwanele.com/map/all_breaches>. [Consulta: 25 de marzo de 2010].

[205] Ramin Nakisa et al. Monte Carlo method. Wikipedia. Disponible en Web: <http://en.wikipedia.org/wiki/Monte_Carlo_method>. [Consulta: 5 de mayo de 2010].

Capítulo 8

Anexo

8.1. Vistas 3D de los modelos digitales.

En este apartado se muestran las vistas 3D de todos los modelos digitales que aparecen en la página Web de este Proyecto Final de Carrera.

Junto con el modelo digital se muestra a su vez un plano de la zona en el que aparece resaltada la parcela que ocupa el modelo correspondiente.