# MULTIRRESOLUCIÓN Y NAVEGACIÓN INTERACTIVA EN EL MODELO VIRTUAL PARA EL FORUM 2004 DE BARCELONA

JAVIER MONEDERO ISORNA
Profesor del Departamento de Expresión Gráfica Arquitectónica I
Universidad Politécnica de Cataluña
Barcelona - España
www.upc.edu/ega1

HÉCTOR ZAPATA
Colaborador del Departamento de Expresión Gráfica Arquitectónica I
Universidad Politécnica de Cataluña
Barcelona - España
www.upc.edu/ega1

---

En abril de 2002, el Laboratorio de Modelización Virtual recibió el encargo de construir un modelo virtual para el Forum de las Culturas de Barcelona. La finalidad del encargo era diversa pero básicamente se trataba de construir un modelo que sirviera para explicar lo que en aquel momento era una compleja red de intervenciones de diverso tipo que pocos eran capaces de visualizar adecuadamente. Esta finalidad debía servir a una variedad de agentes: arquitectos, gestores, ingenieros o representantes de los ciudadanos. El modelo debía tener el suficiente detalle como para satisfacer a los arquitectos, que necesitaban comprobar como se resolvían determinados encuentros entre edificios y elementos varios, elaborados por diversos profesionales. Pero debía ser, por otra parte, lo suficientemente ligero como para permitir un recorrido interactivo por el conjunto del área de modo que pudiera ser comprendido con relativa facilidad por los no especialistas.

Aunque la multirresolución es un tema que se ha estudiado con cierta extensión en la literatura sobre construcción de modelos virtuales, los problemas técnicos que aparecen en su aplicación a un modelo complejo, como es el caso del Forum tienen, considerable interés pues en un encargo real aparecen problemas concretos que no suelen abordarse en una discusión teórica. Se probaron diferentes alternativas, desde programas comerciales relativamente sencillos, hasta programas más sofisticados, incluyendo los desarrollados por nuestros colegas del UPC/CRV (Centre de Realidad Virtual de la UPC). El trabajo final se concluyó en enero de 2003 y en la ponencia se resumen los problemas técnicos abordados y las soluciones adoptadas. Queremos resaltar la contribución de Carolina Ruiz que desarrolló un estudio específico sobre multirresolución, como trabajo final del Programa de Máster en Informatización de Proyectos Arquitectónicos y en relación con su trabajo de modelado del Parque de Auditorios. El modelo general del Forum fue elaborado por el siguiente equipo: Javier Monedero, Francisco Muñoz, Andrés Lupiáñez, Marc Pujol, Carolina Ruiz, Andreína Linares y Héctor Zapata.

La investigación sobre Multirresolución se ha convertido en los últimos años en un tópico de primordial interés. En el área de Arquitectura y Urbanismo, sin embargo, pese a la importancia de estas técnicas por razones evidentes que se resumirán en lo que sigue, hay un conocimiento mínimo de los avances conceptuales y teóricos y una utilización prácticamente nula de las aplicaciones técnicas derivadas de estos avances.

Con la presente comunicación se pretenden dos cosas. En primer lugar, esbozar, muy someramente, el estado actual de la investigación en estas técnicas. Y, en segundo lugar, comentar su disponibilidad práctica en relación con un caso particular. Como se verá, no suponen ninguna aportación substancial a la literatura sobre este tema. Sin embargo creemos que pueden ser recibidas con interés debido a la importancia que tienen para la arquitectura y el urbanismo y a las pocas experiencias similares que se han desarrollado en esta área. Los dos primeros apartados están dedicados a un resumen de las ideas básicas y el tercero al caso concreto del Forum.

## 1. Modelos reales, modelos virtuales, multirresolución

Los conceptos de “modelo”, “escala”, “precisión” o “resolución” están íntimamente relacionados entre sí. Esta dependencia es obvia en el caso de modelos tradicionales, construidos “a pequeña escala”. Pero no lo es tanto en el caso de modelos virtuales que se construyen “a la misma escala” que los reales.

Para una comprensión adecuada de los problemas implicados hay que comenzar por resaltar que nuestra percepción del mundo real también se basa en “representaciones” y “modelos”.

En los ***modelos reales***, la percepción visual es dinámica y, aparentemente, continua. Pero está pautada por representaciones que hemos interiorizado y que dependen, en principio, de la estructura de nuestro campo visual. En este campo, que cubre unos 150º en sentido vertical y cerca de 208º en sentido horizontal (debido a la refracción de la córnea) la percepción no es homogénea, es mucho más nítida en el área que rodea a la fóvea que en los extremos y la transición hacia los extremos tampoco es regular. La fóvea cubre un ángulo de unos 5.2º, y recientes investigaciones sobre las áreas V1 y V2 del córtex visual han mostrado que el campo central que rodea la fóvea, de unos 10º, se proyecta sobre un área mucho más extensa que el resto (al igual que ocurre con la mano que también se proyecta sobre un área proporcionalmente mucho más grande del cerebro). Por otro lado, la capacidad de resolución del sistema visual humano es de 0º1’, que puede disminuir a 0.5’ en condiciones óptimas (equivalente a entre 380 y 780 dpi a 25 cms de distancia o a la capacidad de resolver una trama de líneas blancas y negras, de 1 mm, a una distancia de entre 7 y 4 metros). La distancia óptima de observación es de unos 25 cms, una distancia que aumenta con la edad. Estas cifras resumen apretadamente los límites físicos principales de nuestro sistema de visión.

La percepción del tamaño de los objetos se debe fundamentalmente a dos factores que dependen de la distancia de observación. Hasta una distancia de poco más de 1 metro la apreciación del tamaño está basada en la modificación del ángulo de vergencia y en la visión estereoscópica. A partir de los experimentos de Bela Julesz y los trabajos de David Marr se sabe que esta apreciación es directa, independiente del reconocimiento de objetos. Más allá de esta distancia crítica, sin embargo, el factor fundamental es la familiaridad con los objetos, el “reconocimiento”. Esto puede ser apoyado en claves complementarias no excesivamente fiables, las principales de las cuales son la oclusión parcial, el gradiente de textura y la convergencia perspectiva de líneas. A partir de 1 metro, los adultos son capaces de discriminar el tamaño de objetos, si el entorno es conocido, hasta una distancia de unos 30 metros, aunque este aprendizaje es largo: los niños de unos 8 años pueden hacerlo hasta una distancia de unos 3 metros: más allá subestiman el tamaño real, tanto más cuanto mayor es la distancia.
La percepción del mundo real se basa en una serie que nos imaginamos continua pero que no lo es. Desde la visión de una montaña a la visión de la hoja de un árbol que hay en esa montaña, hay múltiples transiciones, múltiples imágenes que son teóricamente infinitas pero que pasan, realmente, por estadios concretos, cada uno de los cuales implica una distancia de observación y una determinada “resolución adecuada” para tal distancia. ¿Qué quiere decir “resolución adecuada”? Que el número de muestras tomadas para esa determinada distancia real será el suficiente para que la imagen, contemplada a una determinada distancia virtual, “parezca la misma”. Esta resolución se mide corrientemente en pixels por centímetro. Como referencia provisional podemos dar las siguientes:

| | | |
|---|---|---|
| 1 pixel = | 0.10 cms | Rostro humano ocupando media pantalla (1) |
| 1 pixel = | 0.25 cms | Balcón ocupando toda la pantalla de un monitor de 17” |
| 1 pixel = | 0.50 cms | Detalles arquitectónicos a alta resolución (2) |
| 1 pixel = | 1.00 cms | Modelo de alta calidad (3) |
| 1 pixel = | 10.00 cms | Modelo de baja calidad (4) |
| 1 pixel = | 100.00 cms | Mapa de alta calidad |
| 1 pixel = | 250.00 cms | Mapa de calidad media (5) |

(1) De un monitor de 17" (unos 300 pixels de alto)

(2) Un criterio sencillo es considerar que una junta de ladrillo, de 0.5 cms aproximadamente, debe ocupar al menos 1 pixel para ser representable.

(3) Otra referencia que puede ser de interés es la siguiente: parte de los pavimentos de la explanada del Forum se representaron (tras varias modificaciones) a partir de una imagen, obtenida desde el 5º piso de la ETSAB, que tenía unos 1600 pixels y correspondía a unos 20 metros de extensión. Es decir que, en este caso, 1 px = 1.25 cms.

(4) En otros casos se han utilizado imágenes obtenidas de fotografías áreas tomadas a menos de 100 metros de altura con una resolución de 1 pixel = 10 cms.

(5) Los mapas comercializados por el Instituto Cartográfico de Cataluña tienen aproximadamente esta resolución.

En los ***modelos virtuales*** la percepción es también, aparentemente, dinámica y continua. Pero, aparte del hecho de que los límites son mucho más estrictos que en la percepción real (el campo visual es bastante más simple), la representación básica debe adecuarse a un punto de vista principal. Esto resulta evidente en el caso de las texturas. Si proyectamos sobre una fachada una textura de ladrillo que se ha obtenido mediante una fotografía tomada a 15 metros de distancia es obvio que la simulación será más convincente a distancia cercanas a ésta. Más lejos, aparecerán patrones repetitivos más o menos notorios (según el trabajo que nos hayamos tomado en disimularlos y lo grande que sea el muro). Más cerca, se verán pixels cuadrados, en lugar del detalle de la textura de ladrillo. Pero otro tanto cabe decir de la geometría. Una fachada con ventanas puede simularse con una foto pegada sobre un plano, con "geometría falsa". Pero a corta distancia se verá la trampa. Si el modelo no se va a contemplar a estas distancias críticas podemos aceptar la simplificación. En caso contrario será necesario aumentar la resolución, lo que resulta, y seguirá resultando durante muchos años, prohibitivo.

Un modelo con ***multirresolución*** es un modelo que incluye diferentes niveles de detalle (*LOD, Levels of detail*) que pueden ser activados automática (por ejemplo, cuando se encuentren a una determinada distancia de la cámara) o manualmente (seleccionando una u otra alternativa). Esto implica que debe haber diferentes versiones de un mismo objeto, con geometría más o menos densa. Las diferentes versiones pueden generarse también manual o automáticamente, a partir de un procedimiento conocido como "optimización" o "simplificación de superficies" (*surface simplification*) que puede aplicarse a su vez mediante LOD estático o dinámico. Todo esto implica una serie de problemas abiertos.

En primer lugar, los relativos al tipo de modelo utilizado. Algunas técnicas de multirresolución utilizan sistemas de modelado (*voxels*, *octrees*) que facilitan el proceso de substitución y se adaptan muy bien a determinados algoritmos de multirresolución pero que no se utilizan en arquitectura y urbanismo, excepto en ciertas aplicaciones relacionadas con el cálculo de estructuras. Un tema de investigación abierto es el de si los métodos más corrientes en la actualidad, objetos paramétricos que se resuelven en última instancia en mallas poligonales, son los más adecuados para la creación de modelos virtuales de grandes dimensiones. El propio concepto técnico de multirresolución se adapta naturalmente a una estructura de nodos subdivididos de modo regular, como resulta obvio si pensamos en un mapa que se subdivide en cuatro partes iguales cada una de las cuales puede ser substituida por un mapa con mayor detalle y este a su vez en otros cuatro, etc., hasta formar un árbol recursivo (*quadtree*) que se puede recorrer con facilidad.

Por otro lado, muchas de las técnicas que se están desarrollando recientemente, tales como la reconversión de mallas poligonales por medio de parámetros multidireccionales o la reestructuración jerárquica de mallas, son relevantes para simular superficies orgánicas, de curvaturas complejas. Pero este es un problema que, en el caso de la arquitectura aparece tan sólo de modo excepcional, principalmente en el caso de terrenos para los que ya se cuenta por otro lado con técnicas específicas.

Y, por añadidura, muchas técnicas de modelado no son eficientes en la práctica. Por ejemplo, el modelado de superficies libres con curvas suaves, tales como los parches de subdivisión (*subdivision patches*) es, en principio, mucho más eficiente debido a que estos parches sólo requieren un pequeño número de puntos de control, a diferencia de lo que ocurre con las mallas poligonales corrientes. Sin embargo, esta eficacia se pierde en el momento en que se envían a un motor de rendering que, como ocurre en todas las tarjetas gráficas actuales, requiere que la superficie se tesele, se recubra de una malla virtual adecuada para ser procesada por los chips especializados de la tarjeta gráfica. El ancho de banda ganado con un procedimiento más elegante y más adecuado se anula debido a que los requisitos mecánicos están opitimizados para fuerza bruta.

En segundo lugar, hay que mencionar los problemas relativos al motor de render o *3d engine* utilizado. Hay mallas multiresolución (*progressive meshes*) que permiten LOD dinámico. Esto está disponible en algunos motores de rendering para juegos. Pero no en el software comercial más corriente. Por otro lado, hay programas que trabajan con "estructura jerárquica de la escena", algo sobre lo que volveremos en el siguiente apartado, pero sin que queden claros muchos aspectos técnicos involucrados (como ocurre, por ejemplo en 3D Studio), por insuficiente documentación técnica de los manuales.

Un último problema, que nos toca más de cerca, es que no hay investigadores en el área de arquitectura y urbanismo que estén al tanto de unas técnicas que se han desarrollado con gran intensidad en los últimos años (aunque los antecedentes teóricos pueden remontarse a finales de los 1970). Sería bueno que esto se subsanase con más trabajos de doctorado en este campo que, sin pretender descubrir nuevos algoritmos, buscasen adaptar nuestros conceptos a las nuevas tecnicas y, a la inversa, plantear nuevos problemas necesitados de soluciones técnicas originales. Los investigadores interesados pueden encontrar referencias e información buscando en Internet o en Bibliotecas especializadas por palabras clave tales como CAD, CAAD, Surface modeling, Data structures, Computer vision, Visualization, Interactive display, Virtual reality o Terrain modeling, entre otras. Por lo que respecta, más concretamente, a la multirresolución, hay un gran número de técnicas o algoritmos específicos que se han elaborado en los últimos años: diezmación (decimation) de mallas triangularizadas, uno de los primeros algoritmos de simplificación poligonal (Shroeder, Zarge y Lorenson); agrupación de vertices (Rossignac y Borrel); agrupación de celdas (Low-Tan); análisis arbitrario de mallas de múltiple resolución (Eck et al.); envolturas de simplificación (Cohen et al., Cohen, Olano y Manocha); técnicas diversas desarrolladas por Garland y Heckbert o Erickson y Manocha; simplificación mediante el manejo de imagen (Lindstrom y Turk), mallas progresivas (Hoppe); simplificación dinámica jerárquica (Luebke y Erikson). En el Departamento de Lenguajes y Sistemas Informáticos de la UPC pueden encontrarse también referencias notables en trabajos desarrollados por Pere Brunet y Carlos Ándujar entre otros.

## 2. Navegación interactiva

Al igual que en el apartado anterior, en éste se incluirán algunos comentarios sobre el desarrollo reciente y lo que ***no*** es posible hacer en nuestro campo aunque es previsible que lo sea en breve plazo. No está de más recordar que en 2000 había más de 600 motores de rendering en el mercado, incluyendo los más conocidos como Open GL o DirectX.

La aparición de VRML (Virtual Reality Modeling Language) en 1994, con la intención manifiesta de ser el equivalente 3D de HTML en Internet, generó unas expectativas que se han cumplido sólo parcialmente, tras la aparición de la segunda versión en 1997 y el anuncio por parte del Web Consortium de lanzar X3D como alternativa superior. Ciertamente, VRML presenta ventajas bien conocidas: es de libre uso, independiente de la plataforma en la que se visualiza el modelo, y totalmente programable. Pero los inconvenientes también son bien conocidos. En primer lugar, en VRML no se puede contar con aceleración por hardware pues está diseñado para poder emplearse en cualquier plataforma. Algunos navegadores, como Cortona o Cosmo, posibilitan, hasta cierto punto, el uso de aceleración 3D pero su rendimiento dista mucho de ser óptimo. Otro problema es la multitud de navegadores existentes, algunos incompatibles entre si, y cada uno con un sistema distinto de navegación y de representación. En fin, VRML es un lenguaje que dificilmente se podrá mantenerse al día de los último avances en representación 3D, como, para no mencionar sino los anteúltimos, multitextura o LOD dinámico.

Quienquiera que haya tenido ocasión de manejar en una “consola” (por ejemplo, la de una Play Station) uno de esos juegos en los que un coche se desplaza a gran velocidad por una gran ciudad, perfectamente modelada, y que incluye colisiones y efectos de todo tipo, se habrá preguntado como es posible que en un ordenador más o menos potente, sea imposible mantener en escena un número muy inferior de edificios sin que todo se comience a ralentizar. La respuesta básica es que, por un lado, las consolas cuentan con procesadores optimizados para rendering. Y, por otro lado, que los programas incorporan algoritmos especializados adaptados a las necesidades de cada juego. De estos, como veremos, la utilización de algoritmos eficientes para determinar que parte de una escena 3D será visible es un aspecto clave para poder visualizar entornos complejos con suficiente velocidad. Hay un dato que puede ser más revelador que cualquier otra cosa: en 2001, la industria del video juego facturó 9.400 millones de dólares en Estados Unidos mientras que las ventas de entradas de cine sólo alcanzaron los 8.400. En 2000, en España, la industria del vídeojuego, aún despegando, facturó poco menos que la del cine: 510 frente a 538 millones de euros. Tampoco está de más recordar que el desarrollo de un vídeojuego cuenta con presupuestos de varios miles de euros y puede dar trabajo a unas cuantas docenas de personas con diferentes especialidades.

Los programas corrientes de rendering utilizan desde hace algunos años algunos recursos básicos para reducir cálculos innecesarios. Dos de estos recursos básicos son la exclusión de los elementos que caen fuera de la pirámide visual y la exclusión de las caras posteriores de objetos. El primero de estos recursos (*view frustum culling*) excluye del proceso a todos los elementos cuyas coordenadas y dimensiones envolventes queden fuera del volumen determinado por una pirámide con el vértice en el observador y los cuatro planos laterales fijados por el ángulo de visión. El segundo (*back face culling*) presupone que todos los objetos son cerrados y que, por consiguiente, las caras cuyas normales formen un ángulo superior a 90º con el vector de visión no pueden ser visibles. Ambos son bien conocidos y están documentados en cualquier manual sobre rendering.

Sin embargo ésto ha resultado insuficiente para las exigencias del mercado de juegos y han surgido otras técnicas que se conocen, a veces, con el término general de “exclusión mediante oclusión basada en células” (*cell-based occlusion culling*). Aunque la literatura sobre el tema es relativamente amplia nos referiremos sumariamente a las dos principales, BSP y Portal Engines. En los dos casos (como en otros similares) se parte de una subdivisión de la escena en “celdas” (*cells*) conectadas entre sí por “portales” (*portals*). Si un portal no es visto desde un determinado punto de vista ninguna de las celdas tras el portal será visible y puede ser excluida.

El más utilizado de estos dos sistemas, los árboles BSP (*Binary Space Partitioning trees*), se ha utilizado desde principios de los 1990 en la industria de juegos (a partir de una serie de desarrollos teóricos de principios de los 1980 algunos de cuyos principios pueden remontarse hasta 1969). Básicamente un árbol BSP es una subdivisión jerárquica del espacio en subespacios convexos. Esto puede utilizarse con varias finalidades. Originalmente se hacía para optimizar los cálculos de rendering, básicamente la ocultación de caras y el cálculo de sombras. Posteriormente se utilizó principalmente para reestructurar la escena de modo que sólo una parte de los datos necesitara ser cargada. El área que se preveía que iba a cambiar se rodeaba con un polígono. Como el algoritmo BSP no interactúa con normales invertidas (el interior del polígono) éste queda aislado. También se utiliza para ordenar los polígonos en función de su distancia al observador (el jugador). Los motores 3D actuales que utilizan BSP se basan en un sistema denominado PVS (*Potentially Visible Sets*) que efectúa un cálculo previo para determinar, para cada celda de la escena, todas las otras celdas que son “potencialmente visibles”. De este modo, al hacer un render, se localiza la celda en que está el observador así como las celdas que quedan dentro de la pirámide de visión y sólo es necesario representar las que cumplen estas dos condiciones, quedar dentro del PVS y de la pirámide de visión.

En el segundo y más reciente método, los *Portal Engines*, la escena se descompone de un modo similar. Al hacer un render, se localiza la celda en la que está el observador y todos los portales de esta celda se comprueban en relación con los planos de la pirámide de visión. Los que quedan dentro, total o parcialmente, se vuelven a procesar, de modo que el resultado es equivalente a ir añadiendo sucesivos planos de corte. La ventaja de este método es que no requiere un preproceso y de que es mucho más eficiente que el anterior. Sin embargo el cálculo analítico implicado para hallar los planos de corte es costoso por lo que se han desarrollado técnicas alternativas que reducen la efectividad a coste de un procesamiento menos estricto.

Es decir, que la navegación interactiva eficiente implica varias cosas: multirresolución, definición adecuada de niveles (LOD), métodos estáticos o dinámicos de simplificación de superficies, métodos jerárquicos de organización de la escena y, en determinados casos, de los propios objetos por medio de superficies de subdivisión jerárquica, capacidad de procesamiento de canales alfa y de mapas procedurales, capacidad de procesamiento por métodos avanzados de iluminación y cálculo de sombras.

La realidad queda bastante lejos de la anterior descripción. Muchos navegadores no cuentan con métodos como los citados para optimizar la navegación ni con métodos avanzados de iluminación ni tan siquiera con canales alfa o mapas procedurales. Por esta razón hay que recurrir a trucos más o menos eficaces, como simular la iluminación con texturas o renunciar a los canales alfa o, si es posible traducirlas a formatos más unversales, como PNG o convertir las texturas procedurales en mapas de bits, grabando una representación y reproyectando el resultado como mapa de bits.

## 3. El modelo elaborado para el Forum 2004

La generación de un sistema simple de Realidad Virtual para el Forum 2004 era prácticamente la primera de estas características que desarrollábamos en el LMVC. Contábamos con la experiencia de proyectos anteriores, como el proyecto Casas Recuperadas (visita virtual a proyectos no construidos, desarrollado también por los autores de esta ponencia), pero estos modelos eran muy inferiores en dimensiones y complejidad al proyecto completo del Forum. En el resultado, aunque plenamente satisfactorio, se pueden señalar algunos fallos propios de la inexperiencia y de la complejidad inherente al proyecto.

El objetivo inicial no era éste sino obtener un modelo 3D de alta resolución con la máxima calidad posible para obtener imágenes estáticas. La idea de realizar una visualización interactiva del modelo apareció posteriormente. Debido a esto el modelo de partida era completamente inviable para una visualización en tiempo real. Éste modelo contaba aproximadamente con 770.000 polígonos y las texturas ocupaban en total 345 Mb.

Estas dimensiones sobrepasaban las capacidades del hardware más potente basado en PC por estas fechas. A título orientativo, pues este factor puede variar enormemente, una tarjeta aceleradora 3D actual puede manejar del orden de 5 millones de polígonos por segundo, que a una tasa de 50 fps, cuadros por segundo equivale a representar con relativa soltura un modelo de 100.000 polígonos. El modelo original del Forum sólo se hubiera podido mostrar a unos 6 fps, algo totalmente insuficiente. El problema de las texturas era aún más insalvable: una tarjeta gráfica normal tiene de 32 a 128 Mb de memoria, tanto para la geometría como para las texturas. Incluso las más potentes no suelen sobrepasar los 256 Mb, con lo cual las texturas del modelo no habrían cabido en memoria. A pesar de que las tarjetas gráficas AGP pueden emplear memoria del sistema para las texturas, esto hubiera supuesto una ralentización aún mayor, reduciendo la tasa de refresco, en el mejor de los casos, a 1 ó 2 cuadros por segundo. Era obvio que el modelo debía ser simplificado.

Otro problema al cual debíamos enfrentarnos era la elección del software a emplear para la creación del modelo virtual. Dada la gran complejidad del modelo, incluso una vez simplificado, un requisito imprescindible era el uso de aceleración por hardware para poder representar el modelo a una velocidad suficiente.

Sin duda alguna, la solución óptima desde el punto de vista de la velocidad de representación es el método empleado en la creación de videojuegos: la programación directa del motor de representación en un lenguaje de programación como C++, empleando alguna de las APIs más usuales de representación de modelos 3D como OpenGL o DirectX. Este método permite una getión óptima de los recursos y el sistema más rápido posible de creación de sistemas de LOD y programación de comportamientos. Sin embargo, esta opción quedaba claramente fuera de nuestro alcance debido a los grandes recursos humanos, técnicos y de tiempo que requiere.

Descartada esta opción, la primera alternativa considerada, por el otro extremo, fue VRML pero los inconvenientes mencionados en el apartado anterior eran claramente de más peso que las ventajas. Tras algunas pruebas, el software que finalmente escogimos para realizar la aplicación fue Macromedia Director, un software de aplicaciones multimedia que en la versión 8.5 incorpora la representación de modelos 3D. Sus ventajas, frente a VRML son que, en primer lugar, cuenta con aceleración por hardware, tanto mediante OpenGL como DirectX. En segundo lugar, permite emplear una amplia serie de técnicas actuales de representación tal como múltiples texturas, MIP mapping, canales alpha, mapas de reflejo y LOD, tanto dinámico como estático. Además, permite crear un archivo ejecutable autónomo que se puede ejecutar en cualquier PC sin necesidad de ningún software adicional. Y, en fin, ofrece pleno control sobre la representación, la navegación y la programación de todos los aspectos de la aplicación mediante su lenguaje de script, Lingo.

Naturalmente, también presenta algunos inconvenientes. En primer lugar, permite el uso de modelos 3D pero no cuenta con un editor interno, por lo que todos los cambios sobre los modelos 3D deben programarse o realizarse con un programa de modelado externo. Además, al utilizar múltiples APIs de representación no está optimizado, lo que unido al hecho de que el lenguaje Lingo es interpretado y por tanto mucho más lento que C++, hace que el rendimiento general de una aplicación de Director, aunque superior al de VRML, sea mucho menor que el de una aplicación programada directamente. Por estas razones, y como se comentará en las conclusiones, ahora utilizamos otro software que combina una gran facilidad de uso con un rendimiento casi equivalente a la programación directa.

Con esta elección, el primer paso, como ya hemos explicado, fue la simplificación de la geometría y las texturas. El problema era complejo porque suponía renunciar a gran cantidad de detalles que se habían considerado importantes a la hora de realizar el modelo. Además el hecho de partir de un modelo más complejo y simplificarlo presentaba gran multitud de problemas: el modelo simplificado nunca contenía tan pocos polígonos como si se hubiera realizado desde cero con ese nivel de detalle. Es posible que la solución óptima hubiera sido realizar primero el modelo simplificado y posteriormente añadirle complejidad.

Un recurso de simplificación que empleamos ampliamente fue el uso de texturas para sustituir geometría. En el caso de muchos de los edificios, una fachada compleja con gran multitud de huecos se sustituyó por una fachada plana con una imagen de la original proyectada como textura. Este método nos permitió aligerar enormemente el Edificio Forum y el terreno, entre muchos otros. Tiene un problema sin embargo: aunque reduce el número de polígonos, aumenta la memoria de texturas.

En cuanto a las texturas, el único método viable era reducir su resolución, lo cual se hizo de forma automática mediante macros. Al reducir el tamaño de las texturas pasamos de 345 a 55 Mb, lo cual ya era aceptable para una representación en tiempo real.

Finalmente el modelo pasó de 770.000 a 307.000 polígonos. Este modelo simplificado, aunque ya era apto para visualizarlo en tiempo real, aún era demasiado complejo para la mayoría de ordenadores y tarjetas gráficas. Por tanto se tomó la decisión de realizar un segundo modelo aún más simple para ordenadores y tarjetas gráficas medias. En este modelo, ante la imposibilidad de simplificar aún más algunos objetos, se optó por eliminar los edificios del entorno cercano. El modelo súper simplificado constaba de 178.000 polígonos, suficientes para una visualización a 30 FPS.

Una vez cumplido el complejo objetivo de obtener un modelo 3D de tamaño suficiente, el siguiente paso fue introducirlo en Macromedia Director y programar el aspecto y comportamiento de la aplicación de Realidad Virtual. El primer y más sencillo paso fue la creación de los menús de incio, navegación y ayuda, para lo cual ni tan sólo fue necesario el uso de programación en Lingo.

La parte más compleja fue crear el sistema de navegación. Al objeto de crear una presentación lo más versátil posible, se plantearon tres modos diferentes de visualización. El primer sistema, que se activa automáticamente al iniciar la aplicación, consiste en un vuelo prediseñado por todo el proyecto del Forum. El segundo es un vuelo en el que el usuario puede moverse libremente. Por último, el tercero es un paseo, restringido a la altura de una persona. En estos dos casos el usuario puede cambiar entre diferentes posiciones iniciales predefinidas.

El control de la vista fue un aspecto que se estudió muy detenidamente. Se probaron sistemas de movimiento empleados en los videojuegos en primera persona, pero aunque muy útiles para alguien habituado resultaban complejos y confusos de utilizar por primera vez. Finalmente se optó por un sistema de navegación lo más sencillo posible: la posición del ratón en pantalla define hacia dónde se mueve la vista. Con el cursor centrado la cámara no se mueve. Con el cursor en el lado derecho la cámara se desplaza hacia la derecha, y así sucesivamente. Mediante el botón izquierdo del ratón se avanza y mediante el derecho se retrocede. Este método tan simple supuso uno de los aspectos más complejos de programar en Lingo.

Mucho más sencillo de programar fue el cambio de vistas en el que bastaba con emplear scripts tan sencillos como el que sigue (en lenguaje corriente: “cuando se pulsa sobre el botón “Cambiar Cámara”, desplazar el objeto “Cámara” a la posición deseada, y definirla como la cámara actual”):

```
on mouseup me
  global Camara
  if Camara=1 then
    Camara=2
    camera.transform = camera[2].transform
end
```

Algo más complejo fue crear el sistema de detección de colisiones. Para impedir que el usuario pudiera accidentalmente atravesar los objetos, y para restringir el movimiento a la altura de una persona en modo paseo, se programó un sistema mediante el cual cuando se detecta que se va a atravesar un objeto, el movimiento se detiene. Este sistema, aunque funciona correctamente, presenta un inconveniente: al estar programado en Lingo, el cálculo de las colisiones y la simulación de la gravedad son bastante lentos, con lo cual, al tener que calcularlos en cada fotograma, supone una reducción en la tasa de cuadros por segundo. Lo ideal hubiera sido emplear una geometría simplificada para las colisiones, un modelo mucho más simple no representable.

## 4. Conclusiones.

El modelo virtual cumplió con las exigencias requeridas. Era la mejor solución posible dadas las circunstancias. Permite una navegación interactiva fluida por un modelo 3D de grandes dimensiones. Por supuesto, el resultado es mejorable en muchos aspectos, tal y como ha probado nuestra experiencia posterior. Principalmente hay dos áreas en las que se podrían haber introducido mejoras.

En el aspecto visual, el uso de iluminación global precalculada hubiera proporcionado una calidad de imagen muy superior. Actualmente el hardware gráfico apenas puede calcular el cálculo de iluminación directa con sombras en tiempo real con soltura, de modo que la solución para obtener un resultado más realista pasa por calcular la iluminación en un programa de renderizado con iluminación global, y aplicar el resultado de esta iluminación como una textura sobre la geometría.

En el aspecto técnico, el uso de diferentes técnicas, principalmente el uso de LOD, podría mejorar el rendimiento y permitir visualizar un modelo más complejo. El LOD dinámico se descartó por dos motivos: en primer lugar, el ahorro en el número de polígonos tiene la contrapartida del tiempo de cálculo necesario para reducirlos en tiempo de ejecución. En segundo lugar, la geometría se simplifica de un modo automático pero no siempre, por no decir raras veces, satisfactorio para el diseñador. Lo ideal hubiera sido emplear LOD estático, con modelos 3D muy simplificados, de unos 10-100 polígonos por edificio.

Ambas mejoras son, aunque posibles, muy difíciles de implementar en Lingo, dado que Director es un software de uso general que no está optimizado para aplicaciones de Realidad Virtual de gran complejidad. Actualmente empleamos un software de autoría 3D llamado Quest3D, que presenta tanto una gran facilidad de uso como un total control sobre la API DirectX y todas sus características. Aunque la programación básica es en C++ las funciones están empaquetadas en un editor visual relativamente sencillo de manejar y permite implementar con facilidad tanto la iluminación precalculada como el LOD estático. Además, su rendimiento es muy superior al de Director, por lo que el movimiento es mucho más suave y fluido. En cualquier caso, es evidente que habrá que estar bastante atentos a los avances en un área que está evolucionando con una gran rapidez.

**Referencias**


Barth, T.; Chan, T.; Haimes, R. (2002) *Multiscale and multiresolution methods theory and applications*. Berlin, Springer, 2002.

Eberly, D. H. (2000). *3D Game Engine Design*. Addison Wesley.

Hoppe, H. (1996): "Progressive Meshes", *SIGGRAPH '96 Proc.*, 1996, aug., pp 99—108. Ver también: http://research.microsoft.com/~hoppe/.

Rossignac, J.; Borrel, P. (1993): "Multi-resolution {3D} approximations for rendering complex scenes", en: B. Falcidieno and T. Kunii (eds): *Modeling in Computer Graphics: Methods and Applications*, Berlín, Springer, 1993, pp 455—465 (Proc. of Conf., Genoa, Italy, June 1993)

Rosenfeld, A. *Multiresolution Image Processing and Analysis*, Berlín, Springer, Berlin.

Schröder, P. (2001): *Subdivision, Multiresolution and the Construction of Scalable Algorithms in Computer Graphics*,. Cambridge University Press, 2001.