Visualización interactiva 3D en dispositivos de informática móvil

Javier Lluch Rafael Gaitán Miguel Escrivá Emilio Camahort

Dept. de Sistemas Informáticos y Computación
Univ. Politécnica de Valencia
Camino de Vera s/n 46021 Valencia
{jlluch,rgaitan,mescriva,camahort}@dsic.upv.es

Resumen

En este artículo se presenta un sistema cliente-servidor que permite visualizar, de forma interactiva, escenas 3D en dispositivos móviles. Este tipo de dispositivos tienen grandes restricciones en cuanto a memoria y potencia de cálculo. Por lo tanto, se necesita limitar la cantidad de geometría que se envía desde el servidor a cada cliente. La geometría que es visible en cada cliente se extrae de la escena y se envía a través de la red. El cliente visualiza la geometría que recibe utilizando OpenGL|ES [11]. El algoritmo de extracción de geometría emplea multirresolución y simplificación dependiente de la vista. Los resultados que se presentan utilizan una implementación software de OpenGL|ES que funciona sobre PocketPC 2003.

1. Introducción

Recientemente, se ha avanzado considerablemente en el desarrollo de dispositivos móviles, cada vez disponen de una mayor potencia de cálculo, mayor memoria, e incluso algunos integran tecnología de comunicación sin cables. Los gráficos por ordenador también han avanzado rápidamente en este tipo de dispositivos, actualmente se está desarrollando una librería gráfica 3D denominada OpenGL|ES (for embedded systems). Sin embargo, los dispositivos móviles todavía tienen muchas limitaciones referidas tanto a potencia de cálculo como de tamaño de memoria, y la mayoría de los dispositivos que se encuentran en el mercado, no disponen de aceleración gráfica por hardware. Este hecho dificulta la visualización interactiva de escenas tridimensionales de gran tamaño.

Por otro lado, los recientes avances en los campos de diseño 3D, de adquisición de imágenes y de simulación han llevado a que los datos geométricos sean cada vez mayores, de modo que exceden el tamaño de la memoria principal y las capacidades de visualización del hardware gráfico actual. Se han presentado diversas soluciones algorítmicas para superar el hueco cada vez mayor que existe entre la potencia del hardware y la complejidad de los datos geométricos. Entre estas proposiciones se encuentran: la visualización con multirresolución jerárquica, el recortado contra el volumen de la vista, la selección por oclusión y la visualización basada en la imagen.

Los métodos de recortado se pueden combinar fácilmente con modelos multirresolución para seleccionar los niveles de detalle de acuerdo con criterios como la situación del punto de vista, las condiciones de iluminación y el ratio de cambio de la escena. Esta aproximación se denomina simplificación dependiente de la vista, sin embargo, el uso de modelos multirresolución incrementa el tamaño de los datos geométricos y requiere que toda la geometría se mantenga en la memoria principal. Esta es la razón por la que los modelos multirresolución sólo se pueden aplicar para conjuntos de datos que no excedan el tamaño de la memoria principal.

Para solucionar este problema, se han desarrollado nuevos modelos que permiten almacenar grandes escenas en memoria secundaria (out-of-core), de modo que se pueden visualizar, de forma interactiva, conjuntos de datos enormes utilizando técnicas de simplificación dependiente de la vista.

Todas estas técnicas pueden ser aplicadas a la visualización de escenas 3D complejas en dispositivos móviles, debido a las carencias de potencia de cálculo y de memoria que tienen. Existen, principalmente, tres técnicas para visualizar en 3D sobre dispositivos móviles: visualización basada en la imagen, que utiliza imágenes precalculadas para sustituir geometría y

obtener efectos realistas a bajo coste [1] [15] [17], visualización basada en el punto, que visualiza geometrías complejas procesando un conjunto de puntos sobre la superficie de los objetos [4] y visualización basada en polígonos.

En este artículo se hace uso de la última, ya que se van a aplicar técnicas actuales de multirresolución, de recortado contra el volumen de la vista y de out-of-core para visualizar en 3D en un dispositivo móvil. Se presenta un sistema cliente-servidor que envía geometría simplificada a la PDA a través de una conexión inalámbrica. El servidor almacena el grafo de escena y extrae los niveles de detalle necesarios para su visualización en el cliente. Una precisa selección de los niveles de detalle apropiados permite que en el dispositivo móvil se visualice a ratios interactivos.

El sistema que se ha desarrollado usa una caché de geometría que maneja dos copias del conjunto de los objetos visibles. Una copia está en el servidor y la otra en el cliente. El servidor actualiza la caché cuando los parámetros de la vista cambian. Un proceso de sincronización mantiene la coherencia entre ambas cachés, de modo que sólo se envían actualizaciones de la geometría desde el servidor al cliente, reduciendo la latencia y los requerimientos de ancho de banda del sistema. Además, este método tiene la ventaja de poder utilizar técnicas avanzadas de recortado en el lado del servidor.

El sistema desarrollado en el servidor también soporta multirresolución y simplificación dependiente de la vista, de modo que permite la selección del nivel de detalle apropiado en función de los parámetros de la vista. El sistema permite la visualización de escenas de cualquier número de polígonos. Por lo tanto, mejora aproximaciones anteriores porque no almacena la totalidad de la escena en el dispositivo móvil, ni tampoco se realiza el proceso de visualización en el servidor y se envían las imágenes generadas al cliente. Al contrario, el sistema se asegura de que el cliente sólo recibe aquellas partes del grafo de escena que son visibles, y las visualiza en el dispositivo móvil. Para ello se utiliza una librería software, de modo que en cuanto esté disponible la aceleración gráfica para este tipo de dispositivos, se podrá visualizar una mayor cantidad de geometría sin necesidad de realizar grandes cambios en el sistema.

El artículo está estructurado de la siguiente forma: primero, se revisan los trabajos previos relacionados con la visualización 3D en dispositivos móviles, modelado multirresolución y procesado de geometría out-of-core, después se presenta el sistema y su arquitectura, en la siguiente sección se muestran algunos de los resultados obtenidos con nuestro sistema, finalmente se presentan las conclusiones y las directrices a seguir en el trabajo futuro.

2. Trabajos previos

En esta sección se va a realizar un estudio de los trabajos previos que se han realizado en visualización 3D en dispositivos móviles basados en polígonos, modelado multirresolución y métodos out-of-core. También se presenta la idea de simplificación dependiente de la vista en memoria externa y la API de OpenGL|ES.

2.1. Visualización 3D en dispositivos móviles basada en polígonos

Estos sistemas son parecidos a los que se implementaban en las tradicionales aceleradoras gráficas, suelen utilizar sus propias librerías graficas, aunque algunos utilizan PocketGL [12], que es una librería software para dispositivos móviles. Esta librería fue desarrollada para dispositivos específicos, la mayoría de ellos no eran compatibles con OpenGL|ES. La visualización de escenas tridimensionales se hace mediante la conversión al raster por software utilizando la API 2D nativa, para escribir en la memoria gráfica del dispositivo.

D'amora y Benardini presentan un visualizador 3D para dispositivos pocketPC [2]. El visualizador proporciona una solución independiente para acceder a modelos MCAD 3D durante las fases del proceso de fabricación donde la utilización de estaciones de trabajo es imposible o limitada. El visualizador acepta modelos en su propio formato comprimido. Los modelos se envían a través de la red inalámbrica y se descomprimen en la recepción. El problema de este sistema es que envía todo el modelo al dispositivo de visualización, en vez de enviar una versión simplificada. Por lo tanto, el tamaño de las escenas que se pueden visualizar viene limitado por el dispositivo móvil, y por ello no es apropiado para la visualización de grandes conjuntos de datos.

Zunino y otros [18] describen un motor de visualización con multirresolución continua basado en PocketGL para PDA's. El motor puede ajustar dinámicamente el nivel de detalle de los objetos de acuerdo con el número de imágenes por segundo que el usuario quiera alcanzar. El ratio se considera como una función objetivo que determina tanto la calidad de la escena como la capacidad de interacción. El visualizador requiere que los modelos geométricos estén almacenados en la memoria de la PDA. Esto limita el tamaño de la escena que se puede visualizar.

Sanna et al. proponen una arquitectura general para búsqueda, recuperación y visualización de modelos tridimensionales complejos sobre una PDA [14]. La arquitectura incluye servidores de geometría, servidores de visualización y aplicaciones cliente que visualizan las imágenes obtenidas. El principal problema que presenta este sistema es que sólo es capaz de representar un objeto que es transmitido completamente al cliente, y por lo tanto su tamaño no puede exceder la memoria del mismo.

2.2. Representación de mallas multirresolución

Las mallas multirresolución se utilizan a menudo para construir una representación de superficies geométricas con distintos niveles de detalle. El uso del término multirresolución indica que la precisión (o nivel de detalle) de la malla que aproxima una superficie, está relacionada con su resolución, es decir, con su densidad (tamaño y número) de sus celdas. Una malla de este tipo proporciona diferentes mallas que aproximan al objeto (por ejemplo, una superficie que representa la frontera de un objeto sólido).

Una malla multirresolución es una colección de fragmentos de malla que generalmente describen pequeñas porciones de un objeto con diferentes precisiones. También incluye las relaciones adecuadas para permitir seleccionar un subconjunto de fragmentos (de acuerdo con el criterio de precisión definido por el usuario), y combinándolos en una malla que cubre parte o la totalidad del objeto. Los modelos multirresolución existentes difieren en el tipo de los fragmentos de malla que se utilizan y la manera en la que se definen la relaciones entre dichos fragmentos. En [7] se puede obtener una información más detallada sobre este tipo de modelos.

2.3. Simplificación dependiente de la vista en memoria externa

Recientemente, se ha presentado la idea de simplificación dependiente de la vista [6]. Dicha técnica, permite cambios en una jerarquía multirresolución que depende de parámetros tales como la locación del observador, las condiciones de iluminación o la velocidad del movimiento. Estas simplificaciones adaptan la estructura de la malla para obtener el nivel de detalle correcto en cada imagen. El problema principal es que este tipo de estructuras incrementan el tamaño de los datos y requieren que estén almacenados en la memoria principal.

Las técnicas de memoria externa (out-of-core) solucionan este problema para conjuntos de datos cuyo tamaño sea mayor al de la memoria principal [5]. La idea es almacenar las estructuras de datos en el disco duro y posteriormente los datos son preprocesados para obtener un conjunto de árboles, dependientes de la vista, optimizados para realizar operaciones de entrada y salida. Estos árboles están formados por jerarquías con multirresolución. Durante la navegación, se mantienen todos los árboles en el disco, de modo que la memoria principal sólo almacena aquellos árboles que son necesarios para obtener el nivel de detalle actual. Además, también se almacenan en memoria principal algunos otros árboles que se prevé que serán necesarios en un futuro próximo.

2.4. OpenGL|ES (Embedded Systems)

OpenGL|ES es una librería de bajo nivel y muy ligera para generar gráficos 3D en dispositivos móviles. Utiliza un subconjunto de las órdenes que contiene OpenGL y además proporciona una API de bajo nivel entre las aplicaciones software y los motores de visualización ya sean hardware o software.

Esta API estándar para la generación de gráficos 3D para dispositivos móviles facilita la obtención de gráficos 3D avanzados para una amplia gama de dispositivos móviles. Como la librería está basada en OpenGL no se necesitan nuevas tecnologías, lo que asegura la sinergia y la posibilidad de migración a OpenGL, la API gráfica más extendida en la actualidad.

Desde la aparición de OpenGL|ES, los gráficos 3D en dispositivos móviles han avanzado

bastante. Actualmente, sólo unos pocos dispositivos incorporan esta tecnología por hardware (véase [3] como ejemplo). Los fabricantes están comenzando a incorporar aceleración gráfica por hardware en la nueva generación de dispositivos.

La falta de aceleración por hardware ha motivado el desarrollo de librerías de código abierto que implementan una interfaz para OpenGL|ES, como Klimt [9] o Vincent3D [16]. Nuestra implementación se basa en la primera.

3. Nuestro sistema

En este trabajo se resuelve el problema de visualizar escenas 3D cuyo tamaño exceda el de la memoria principal sobre dispositivos móviles. Para ello, se utiliza almacenamiento out-of-core y simplificación dependiente de la vista para almacenar en el dispositivo el menor número posible de polígonos. Para alcanzar este objetivo se emplean mallas multirresolución y un sistema cliente-servidor para distribuir la geometría.

La elección de esta arquitectura es la adecuada dado que permite evitar la excesiva transmisión por la red de los datos de la escena, además de que permite visualizar escenas de grandes dimensiones progresivamente. En esta sección se presenta la arquitectura del sistema y se describen sus diferentes componentes.

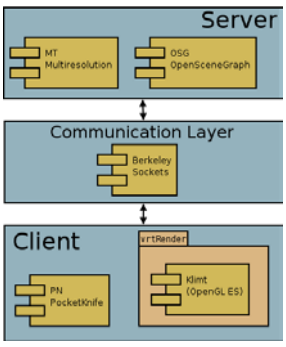

Figura 1. Arquitectura del sistema

3.1. Arquitectura

La arquitectura cliente-servidor de nuestro sistema se muestra en la figura 1. El sistema está dividido en tres subsistemas principales: (i) el sistema servidor capaz de cargar la escena 3D y extraer los datos geométricos que se deben visualizar en el cliente, (ii) la aplicación cliente que se conecta al servidor, recibe los datos geométricos y los visualiza en el dispositivo móvil, y (iii) el nivel de comunicación que implementa los protocolos de comunicación y la caché de objetos/triángulos para mejorar los ratios de transferencia. En la figura 2 se muestra la arquitectura interna del sistema servidor de la aplicación cliente.

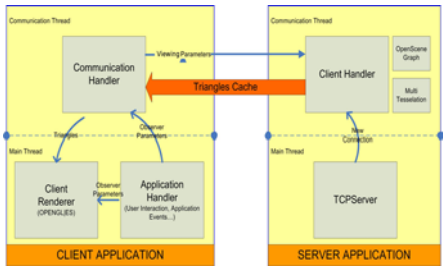

Figura 2. Estructura interna

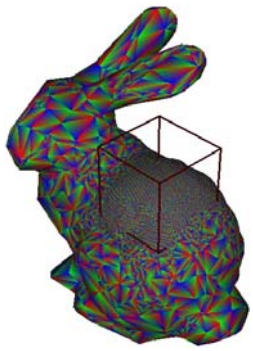

Figura 3. Objeto bunny MT con volumen tipo cubo

3.2. Sistema servidor

El servidor carga el grafo de escena y comienza a escuchar las conexiones. Para almacenar y manejar la escena se usa la librería OpenSceneGraph (OSG) [13], que es una potente herramienta de código abierto utilizada por desarrolladores de aplicaciones gráficas en campos tales como simulaciones visuales, juegos, modelado, realidad virtual y visualización científica. Una vez que el cliente establece una conexión, el servidor crea un nuevo hilo para manejarla, entonces el cliente envía los

parámetros de la vista al servidor y el servidor configura la cámara dentro de OSG, después de esto, el servidor realiza el proceso de recortado.

OSG implementa métodos de recortado que reducen substancialmente la geometría que se envía al visualizador, sin embargo, no soporta técnicas de multirresolución. Para ello el sistema combina OSG con la librería *multitesselation library* (MT) [8]. De esta forma, se pueden manejar escenas complejas con multirresolución.

La librería MT permite utilizar un volumen de inclusión con el que realizar una extracción detallada de la geometría, es decir sólo se extraen los polígonos que están situados dentro de dicho volumen. En función del tipo de escena se puede utilizar como volumen: una caja, una pirámide o cualquier otro objeto. La figura 3 muestra un ejemplo de un volumen de tipo caja. Nótese que el foco de atención de la librería MT, una caja, contiene una resolución mayor que el resto de la malla.

Para realizar el recortado se ha implementado una nueva clase *ExtractVisitor* que extiende la clase estándar de OSG *CullVisitor*. El *visitor* llama al proceso de recortado y extrae la geometría que se debe visualizar en el cliente. Para esto, el *visitor* utiliza los parámetros de la vista y la geometría almacenada en el grafo de escena. Se pueden utilizar todas las primitivas geométricas de OSG. En las mallas MT se hace un trabajo suplementario para extraer el nivel de detalle correcto, ya que se debe crear un volumen que incluya toda la vista para extraer la geometría mediante la librería MT. La geometría obtenida se envía al cliente utilizando el nivel de comunicaciones.

3.3. Aplicación cliente

Se ha creado una aplicación cliente capaz de establecer conexiones inalámbricas con el servidor. La aplicación se puede ejecutar tanto en portátiles, equipos de sobremesa como en PDA's con conectividad inalámbrica. Cuando el cliente establece la conexión se envían al servidor los parámetros de la vista y se recibe la geometría desde el servidor. El hilo principal del cliente visualiza la escena y maneja la interacción con el usuario.

La aplicación cliente se puede ejecutar tanto en plataformas Win32 como en PocketPC's. La ejecución de la aplicación sobre Win32 permite visualizar escenas de forma remota en portátiles y en equipos de sobremesa. La librería PocketKnife hace posible esta dualidad, ya que permite el desarrollo multiplataforma para x86 Windows y Windows CE. Esto quiere decir que se puede desarrollar sobre un equipo de sobremesa, y sólo compilar sobre la PDA para comprobar los resultados y realizar la depuración dependiente del dispositivo. Esta librería provee una clase *Aplication* que sirve de base en el desarrollo de aplicaciones y en el manejo los mensajes de Windows.

La visualización de gráficos 3D en dispositivos móviles se realiza más fácilmente utilizando OpenGL|ES. Por ello, se ha utilizado la implementación software ofrecida por la librería Klimt, que también es multiplataforma, y por lo tanto se ejecuta en PC's y PocketPC's. Por encima de Klimt se ha implementado una librería, *vrtRender*, que proporciona una abstracción para las mallas, materiales, fuentes de luz y texturas. El cliente utiliza esta librería que ofrece una interfaz de alto nivel con OpenGL|ES.

3.4. Nivel de comunicación – La caché de la escena

La comunicación entre el cliente y el servidor se maneja por una librería que se ha desarrollado por encima de la librería estándar *Berkeley sockets*. La librería implementa dos estructuras de datos necesarias para la comunicación entre el cliente y el servidor. La primera encapsula los parámetros de la vista que envía el cliente al servidor y la segunda mantiene la caché de la escena.

Hay dos copias de la caché de la escena, una está en el cliente y la otra en el servidor. El nivel de comunicación mantiene ambas copias sincronizadas constantemente. Durante la navegación la caché es actualizada por el servidor para cada nuevo conjunto de parámetros de la vista. En ese momento el cliente accede a la caché y obtiene la nueva geometría que se debe visualizar. El objetivo de la caché es explotar la coherencia temporal entre dos imágenes consecutivas. Cuando los cambios de los parámetros de la vista son pequeños sólo será necesario realizar pocos cambios en la geometría (local) que se debe visualizar.

La caché de la escena está organizada en dos niveles: la caché de la escena misma y la caché de geometría. La caché de la escena mantiene toda la información necesaria para visualizar la escena, incluyendo los parámetros de la vista y la geometría de la escena. La caché de la escena se invalida cuando los parámetros de la vista cambian, entonces, el servidor extrae la nueva geometría visible e informa al nivel de comunicación, el cual sincroniza las caché del servidor y el cliente.

La caché de la geometría almacena los datos poligonales, de color y de normales de cada objeto geométrico. La caché de la geometría también la actualiza el servidor cuando la caché de la escena se invalida. Para mejorar la eficiencia del sistema, el servidor determina que objetos todavía se encuentran en la caché y tan sólo se añade la geometría de aquellos objetos que han pasado a ser visibles. El proceso de sincronización testea, para todos los objetos almacenados en la caché, una marca de visibilidad generada por el servidor.

Si la geometría era visible solo se envía al cliente el identificador de caché de la geometría, si no lo era, se envía toda la información al cliente.

El método se extiende para soportar mallas multirresolución. En el proceso de extracción se determina si la geometría multirresolución ha cambiado. La idea consiste en comparar la caja de inclusión de la geometría MT visible con la última caja utilizada. Si hay cambios, es necesario recalcular el nivel de detalle correcto que se debe extraer y rellenar la caché con la nueva geometría MT. Finalmente, el proceso de sincronización realiza el resto del trabajo. El sistema también utiliza un factor de error que hace posible, por ejemplo, extraer un mayor nivel de detalle para los objetos más cercanos al observador.

Esta configuración impide el envío de información innecesaria a través de la red. También soporta la navegación a través de la geometría almacenada en el cliente, incluso si la comunicación entre el cliente y el servidor falla.

Caché de doble buffer

El principal problema de esta configuración es el acceso concurrente a la caché de los dos hilos que se ejecutan en el cliente: el de visualización y el de sincronización. Si se bloquea la caché durante el proceso de sincronización, el ratio de visualización cae súbitamente. Por ello, se utiliza una caché con doble buffer, uno para el proceso de visualización y el otro para el de sincronización. Cuando el proceso de sincronización ha finalizado, se realiza un intercambio entre los dos buffers. De este modo, se puede dibujar el contenido de un buffer mientras se sincroniza el otro.

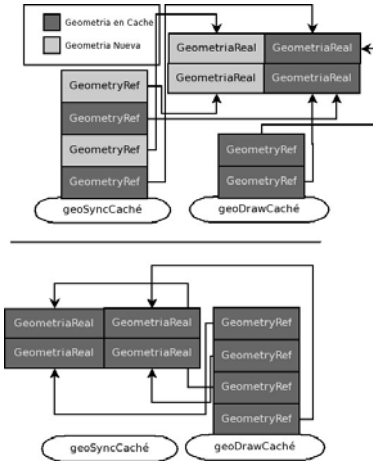

Figura 4. Operación de caché de doble buffer

Figura 5. El cliente ejecutándose sobre el emulador

Para evitar el uso de demasiada memoria, tan sólo se mantienen referencias a la geometría real, que se almacena en un manejador de geometría. Esto lleva a un considerable ahorro en el uso de memoria y además a que el proceso de intercambio sea muy rápido, y por lo tanto a que el periodo de bloqueo sea mínimo. La figura 4 muestra como se realiza todo este proceso. Este sistema funciona con geometrías estáticas, para geometrías dinámicas, se necesita mantener información replicada para evitar problemas durante la navegación. Para este tipo de geometrías el cliente recibe la nueva información de geometría durante el proceso de sincronización, esta información se almacena en buffers

auxiliares. Cuando se hace el intercambio entre los buffers el sistema tiene disponible el nuevo nivel de detalle que se tiene que visualizar.

4. Resultados

Para probar el sistema se ha ejecutado la aplicación servidor sobre un equipo de sobremesa y el cliente sobre diferentes dispositivos: un portátil con acceso inalámbrico, un emulador de PocketPC 2003 sobre el mismo equipo donde se ejecuta el servidor y sobre un HP ip 4150 sobre PocketPC 2003. La figura 5 muestra dos imágenes del sistema funcionando sobre el emulador.

Se han utilizado un par de escenas para testear el sistema. En la figura 6 se muestran algunas imágenes obtenidas al ejecutar el sistema con una malla multirresolución formada por 2400 polígonos. Se consiguen visualizar hasta 10 mallas de este tipo simultáneamente. La figura 7 muestra la otra escena formada por múltiples mallas estáticas que representan una terminal de contenedores [10].

Para analizar el sistema se han hecho las siguientes gráficas con la evolución del ratio de visualización (fig. 8 y 9) y del número de fallos de caché durante la navegación (fig. 10 y 11). Los resultados se han obtenido moviendo la cámara a lo largo de rutas precalculadas La geometría del MT bunny tiene 2400 polígonos como máximo y 1500 como mínimo. La geometría cambia progresivamente durante la navegación. Se puede observar que el número de imágenes por segundo se mantiene al máximo posible. La escena de los contenedores es mucho más compleja, contiene muchos objetos y transformaciones. El camino precalculado para la escena mueve la cámara adelante y atrás por una calle con varios bloques de contenedores a ambos lados.

Se puede observar que el número de imágenes por segundo varía mucho en esta escena, esto es debido a la cantidad de geometría visible que hay en cada momento, y por lo tanto se realizan muchos envíos de geometría, de ahí los dientes de sierra que aparecen en la gráfica. Además, este número no es muy alto debido a que no se dispone de aceleración gráfica y por lo tanto, toda la visualización se hace por software con la librería Klimt. En la figura 12 se puede observar el número de imágenes por segundo que se obtienen si se ejecuta el mismo test sobre un cliente Win32 con una tarjeta aceleradora gráfica. Se puede observar como la evolución es similar, pero a una escala diferente. Por ello, se puede concluir que el problema reside en la utilización o no de aceleración gráfica.

5. Conclusiones y trabajo futuro

En este artículo se ha presentado un sistema para visualizar gráficos tridimensionales de forma interactiva sobre dispositivos móviles. Se ha implementado un sistema cliente-servidor que envía geometría al dispositivo móvil y la visualiza. Para la visualización se ha usado la librería Klimt, una implementación software de OpenGL|ES que funciona sobre PocketPC 2003. Por lo tanto, el sistema se ejecutará sin problemas en los futuros dispositivos que incorporen una implementación hardware de OpenGL|ES.

El servidor usa OpenSceneGraph y mallas multirresolución MT para el manejo de las escenas. Dados los parámetros de la vista enviados por el cliente, el servidor recorta el grafo de escena y envía al cliente sólo la geometría que se tiene que visualizar. La geometría se almacena en una caché de objeto/triángulo manejada por el nivel de comunicación.

El sistema mejora anteriores propuestas porque no almacena toda la escena en el dispositivo móvil, sino que sólo aquellos polígonos que son visibles. El sistema se puede aplicar, por ejemplo, en juegos por ordenador, sistemas de realidad aumentada para monitorizar procesos, visitas guiadas, mantenimiento de almacenes, etc.

Se han realizado pruebas con diferentes escenas y se puede concluir que el sistema funciona bien aunque el tamaño de las escenas sea muy grande. También funciona bien aunque hayan constantes cambios en la geometría visible. Las pruebas se han realizado tanto con geometrías estáticas como dinámicas y ambas con movimientos de cámara. Otra ventaja que cabe destacar es que cuando no se producen grandes cambios en la geometría se produce una notable reducción del ancho de banda utilizado.

En el futuro sería interesante mejorar el funcionamiento de la caché de geometría, de modo que se puedan almacenar en ella nodos multirresolución en vez de triángulos, utilizando técnicas de simplificación dependiente de la vista.

Figura 6. Imágenes del sistema funcionando: arriba el servidor y un cliente, en el centro dos clientes con la misma escena, y abajo con una escena formada por varias mallas multirresolución

Figura 7. Algunas imágenes del sistema ejecutándose con una escena compleja: Una terminal de contenedores.

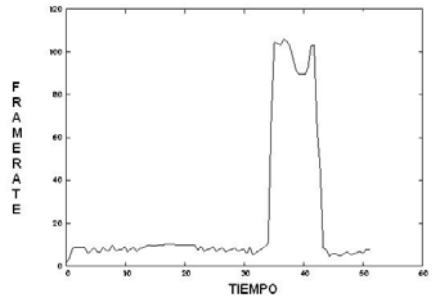

Figura 8. Número de imágenes por segundo obtenido en la visualización de MT bunny sobre la PDA. El pico se debe al momento que el objeto sale del volumen de la vista.

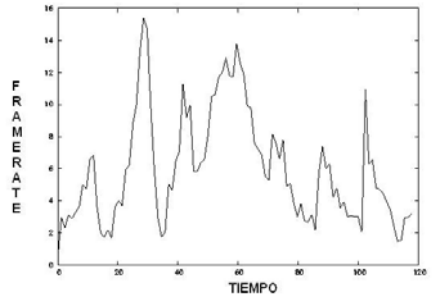

Figura 9. Número de imágenes por segundo obtenido durante la visualización de la terminal de contenedores.

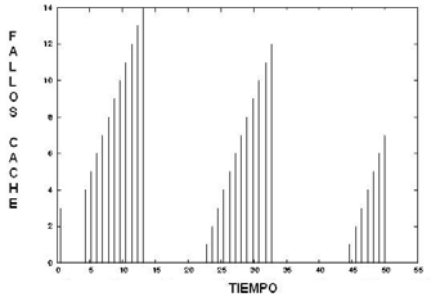

Figura 10. Fallos de caché durante la visualización de MT bunny

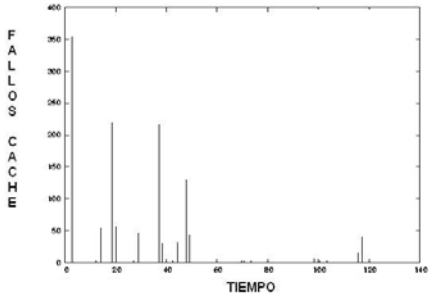

Figura 11. Fallos de caché durante la visualización de la terminal de contenedores

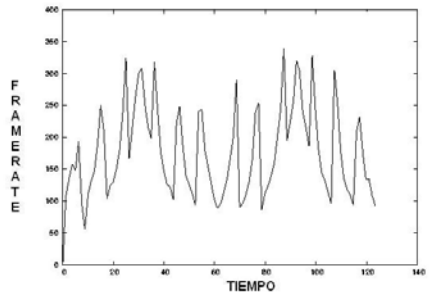

Figura 12. Número de imágenes por segundo visualizando en un PC con aceleradora gráfica

Agradecimientos

Este trabajo ha sido subvencionado por los siguientes proyectos: TIC2002-0416-C03-01 del ministerio de ciencia y tecnología, GV04B-497 2004-2005 de la Generalitat Valenciana e IST-004363 del 6th Framework Program of the European Union.

Referencias

[1] C.-F. Chang and S.-H. Ger. Enhancing 3d graphics on mobile devices by image-based rendering. In *IEEE Third Pacific-Rim Conference on Multimedia (PCM 2002)*, 2002.

[2] B. D'amora and F. Bernardini. Pervarsive 3d viewing for product data management. *IEEE*

Computer Graphics and Applications, pages 14–19, March/April 2003.

[3] Dell Computer Corporation, http://www1.us.dell.com/ *Axim X50v Details*, 2004.

[4] F. Duguet and G. Drettakis. Flexible point-based rendering on mobile devices. *IEEE Computer Graphics and Applications*, pages 57–63, July/August 2004.

[5] J. El-Sana and Y.-J. Chiang. External memory view-dependent simplification. In *EuroGraphics 2000*, volume 19, 2000.

[6] J. El-Sana and A. Varshney. Generalized view-dependent simplification. In *Proceedings EuroGraphics*, 1999.

[7] L. D. Floriani, P. Magillo, and E. Puppo. Multiresolution representation of shapes based on cell complexes. In *Discrete Geometry for Computer Imagery*, number 1568, pages 3–18.Lecture Notes in Computer Science, 1999.

[8] L. D. Floriani, P. Magillo, and E. Puppo. *The MT(Multi-Tesselation) Package*. Dipartimento di Informatica e Scienze dell'Informazione, Universita' di Genova, 2004.

[9] IMS Group, http://studierstube.org/klimt/. limt - the Open Source 3D Graphics Library for Mobile Devices, 2004.

[10] P. Jorquera, J. Lluch, and R. Vivó. Virtainer, "walkthrough" en apilamientos. In *Actas del XIII Congreso Español de Informática Gráfica, CEIG'2003*, pages 373–376, 2003.

[11] Khronos Group, http://www.khronos.org/. OpenGLES - The Standard for Embedded Accelerated 3D Graphics,2004.

[12] P. Leroy. *Pocket GL, 3D library for Pocket PC*.http://pierrel5.free.fr/, 2004.

[13] OSG Community, www.openscenegraph.org. OpenSceneGraph - Open Source high performance 3D *graphics toolkit*, 2004.

[14] A. Sanna, C. Zunino, and F. Lamberti. A distributed architecture for searching, retrieving and visualizing complex 3d models on personal digital assistants. *Internet Technology*, 3(4):235–244, 2002.

[15] P.-P. Vázquez and M. Sbert. Bandwidth reduction techniques for remote navigation systems. In *International Conference on Computational Science*, pages 249–257, 2002.

[16] Vincent 3D - A 3-D Rendering Library for Mobile Devices, 2004. http://ogl-es.sourceforge.net

[17] C. Woodward, S. Valli, P. Honkamaa, and M. Hakkarainen. Wireless 3d cad viewing on a pda device. In *Proceedings of the 2nd Asian International Mobile Computing Conference (AMOC 2002)*, 2002.

[18] C. Zunino, F. Lamberti, and A. Sanna. A 3d multiresolution rendering engine for pda devices. In *SCI 2003*, volume 5, pages 538–542, 2003.