Informe Técnico
DLSI 01/11/2001

Un Esquema Multirresolución General, Simple y Eficiente para Visualización Interactiva de Superficies Poligonales

José Ribelles[i], Ángeles López[ii], Óscar Belmonte[i], Inmaculada Remolar[i], Miguel Chover[i]

[i]Departamento de Lenguajes y Sistemas Informáticos
[ii]Departamento de Ingeniería y Ciencia de los Computadores

Correo electrónico: `ribelles@lsi.uji.es`

Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I
Campus de Riu Sec
E-12080 Castellón, España

A General, Simple and Efficient Multiresolution Scheme for Interactive Visualization of Polygonal Surfaces

José Ribelles[i], Ángeles López[ii], Óscar Belmonte[i], Inmaculada Remolar[i], Miguel Chover[i]

[i]Departamento de Lenguajes y Sistemas Informáticos
[ii]Departamento de Ingeniería y Ciencia de los Computadores
Universitat Jaume I
Campus de Riu Sec
E-12080 Castellón, España

E-mail:ribelles@lsi.uji.es

Abstract:

Modelling objects by means of polygonal surfaces currently predominates in the field of interactive computer graphics. Visualisation of these models is generally slow for distant objects, where the level of detail is excessive, or low in quality for close objects, where the level of detail is insufficient. Multiresolution modelling represents an object through multiple approximations and allows each object to be visualised, in running time, using the approximation with the most suitable level of detail. This means, for example, that less time is wasted on painting imperceptible details. In this work, we present a multiresolution scheme that summarises, improves and expands on our previous study entitled Multiresolution Ordered Meshes (MOM), and which offers simplicity, generality and efficiency as its main characteristics. Developed for the interactive visualisation of polygonal surfaces, it allows the extraction of approximations with both a uniform and variable level of detail. MOM is based on an ordered arrangement of data that is established by the method of simplification used in the construction process. This order will allow most of the dependencies that exist between the different stored triangulations to be codified implicitly, thus allowing lower storage costs and increased performance of access algorithms. Where necessary, we have included and extended material, some of which is not easily available, already published in the proceedings and short communications of different conferences.

Keywords:

Multiresolution modeling, level of detail, interactive visualization, polygonal simplification.

Un Esquema Multirresolución General, Simple y Eficiente para Visualización Interactiva de Superficies Poligonales

José Ribelles[i], Ángeles López[ii], Óscar Belmonte[i], Inmaculada Remolar[i], Miguel Chover[i]

[i]Departamento de Lenguajes y Sistemas Informáticos
[ii]Departamento de Ingeniería y Ciencia de los Computadores
Universitat Jaume I
Campus de Riu Sec
E-12080 Castellón, España

E-mail:ribelles@lsi.uji.es

Resumen:

El modelado de objetos mediante superficies poligonales domina actualmente en el campo de los gráficos interactivos. La visualización de estos modelos es, generalmente, lenta para los objetos distantes, donde el nivel de detalle utilizado es excesivo, o de baja calidad para los objetos cercanos, donde el nivel de detalle resulta insuficiente. El modelado multirresolución representa a un objeto mediante múltiples aproximaciones y permite que, en tiempo de ejecución, cada objeto se visualice utilizando la aproximación con el nivel de detalle más adecuado evitando, por ejemplo, malgastar el tiempo en pintar detalles imperceptibles. En este trabajo se presenta un esquema multirresolución que recopila, mejora y amplía nuestro trabajo anterior denominado Mallas Ordenadas Multirresolución (MOM) y que se caracteriza por su simplicidad, generalidad y eficiencia. Desarrollado para la visualización interactiva de superficies poligonales, permite extraer aproximaciones con nivel de detalle tanto uniforme como variable. MOM se basa en una disposición ordenada de los datos, ordenación establecida por el método de simplificación empleado en el proceso de construcción. Este orden va a permitir codificar de forma implícita la mayor parte de las dependencias existentes entre las distintas triangulaciones almacenadas redundando en un menor coste de almacenamiento y en un aumento de las prestaciones de los algoritmos de acceso. Donde sea necesario, se incluye y extiende material publicado anteriormente en actas y comunicaciones cortas (algunas de difícil acceso) de diversas conferencias.

Palabras clave:

Modelado multirresolución, nivel de detalle, visualización interactiva, simplificación poligonal.

Un Esquema Multirresolución General, Simple y Eficiente para Visualización Interactiva de Superficies Poligonales

J. Ribelles† A. López‡ O. Belmonte† I. Remolar† M. Chover†

†Departamento de Lenguajes y Sistemas Informáticos
‡Departamento de Ingeniería y Ciencia de los Computadores

Universitat Jaume I, 12080 Castellón, Spain
e – mail: {ribelles}@lsi.uji.es
Phone: +34-96-472-83-18, Fax: +34-96-472-84-35

1. INTRODUCCIÓN

Los avances tecnológicos han proporcionado extensas bases de datos de objetos definidos con alto grado de detalle. Métodos como por ejemplo la adquisición de datos vía satélite, escaners por laser o dispositivos de imagen médica, permiten crear, gracias a su alta precisión, enormes conjuntos de datos tridimensionales representados, normalmente, mediante modelos de superficie poligonal [17]. Este tipo de representación es, hoy en día, la más utilizada en el campo de los gráficos interactivos debido a diversas razones como que la mayor parte de los programas de visualización existentes la utilicen o que los sistemas gráficos aceleren la visualización de polígonos. Sin embargo, no es raro que estas superficies poligonales estén formadas por cientos de miles, o incluso millones, de polígonos. Visualizar tal enorme cantidad de datos se convierte en un serio problema cuando el usuario necesita interactuar con la aplicación. Además, este problema llega a ser crítico si hay que mantener una tasa de, por ejemplo, treinta fotogramas por segundo como ocurre en aplicaciones de realidad virtual.

El modelado multirresolución [21] ha sido presentado como una solución eficiente a este problema. Consiste en representar a un objeto mediante múltiples aproximaciones o niveles de detalle (LoDs) cada uno formado por un número diferente de polígonos. De esta forma, la aplicación puede realizar una selección del nivel de detalle (LoD) en tiempo de ejecución a fin de establecer un equilibrio entre el LoD con el que se representa un objeto y la cantidad de tiempo necesario para visualizarlo [24]. En la literatura, los esquemas multirresolución aportan dos posibles soluciones al problema de la visualización interactiva de superficies poligonales: proporcionar aproximaciones donde el nivel de detalle sea *uniforme* [26, 41, 4, 22, 1, 23], o proporcionar aproximaciones donde el nivel de detalle sea *variable* [14, 52, 27, 37, 23, 10]. Por ejemplo, la visualización de una silla en una habitación se realiza utilizando aproximaciones con nivel de detalle uniforme, mientras que el terreno en un simulador de vuelo se debe representar con aproximaciones donde el nivel de detalle varíe a lo largo de la superficie según la visibilidad y la distancia respecto al piloto. En cualquier caso, un esquema multirresolución debe definir las estructuras de datos y los algoritmos de acceso que permiten almacenar y recuperar las diferentes aproximaciones. Su diseño repercute de forma directa en las prestaciones y determina su aplicabilidad. De nada sirve un esquema en el que la sobrecarga que produce el proceso de manipulación de los datos sea tal que haga inútiles los esfuerzos por acelerar la visualización.

Se presenta un esquema multirresolución que recopila, mejora y amplía nuestro trabajo anterior denominado *Mallas Ordenadas Multirresolución (MOM)* [43, 44, 45]. Este esquema aporta una solución al problema de la visualización interactiva de superficies poligonales formadas por cientos de miles de polígonos. MOM permite, ante una petición, extraer aproximaciones de nivel de detalle tanto uniforme como variable de una forma eficiente y manteniendo un bajo coste de almacenamiento. Se basa en una disposición ordenada de los datos, ordenación establecida por el método de simplificación empleado en el proceso de construcción. Este orden va a permitir codificar de forma implícita la mayor parte de las dependencias existentes entre las distintas triangulaciones almacenadas redundando en un menor coste de almacenamiento y en un aumento de las prestaciones de los algoritmos de acceso. Las principales ventajas de este esquema son las siguientes:

- Simplicidad. La estructura de datos básica del modelo se corresponde con una estructura basada en listas, simple y fácil de implementar. Esta característica ayuda a que, tanto el proceso de construcción como los algoritmos de recorrido sean simples también.

- Generalidad. Cualquier método de simplificación puede ser utilizado para construir la representación multirresolución. Tampoco impone restricción alguna respecto al tipo de superficie poligonal a representar, pudiendo ser utilizado tanto para superficies *manifold* como *non-manifold*. Ambas características favorecen el uso de este esquema en un mayor abanico de aplicaciones.
- Eficiencia. Los experimentos realizados muestran que los costes de almacenamiento, los tiempos de extracción de aproximaciones de nivel de detalle uniforme y los tiempos de visualización de aproximaciones de nivel de detalle variable presentan mejoras importantes respecto a otros esquemas multirresolución de características similares a MOM.

Como se verá en la sección 2, son diversos los esquemas multirresolución que se han propuesto en la literatura. Sin embargo, son pocos los que se han desarrollado cumpliendo con la característica de generalidad, es decir, sin imponer condiciones acerca de la geometría del objeto (permitiendo representar superficies *non-manifold*) o sobre la técnica de simplificación válida para construir la representación multirresolución y así poder aprovechar el enorme abanico de posibilidades que existen [25]. Por ejemplo, los esquemas presentados en [51, 26, 27, 48, 22, 42] están restringidos a trabajar con superficies *manifold* y obligan a que la simplificación se realice mediante contracción de aristas. Sin embargo, *Multi-Triangulation* [16] y *Hierarchical Dynamic Simplification* [37] son esquemas que cumplen esta característica de generalidad y, por este motivo, se utilizan en posteriores secciones como esquemas de referencia a fin de compararlos con MOM. Además, también se ha utilizado el esquema *Progressive Meshes* [26] como referencia por formar parte del producto *DirectX* de *Microsoft*.

Estas son las principales contribuciones de este trabajo:

- Nuevas estructuras de datos que aumentan las prestaciones del proceso de recuperación de aproximaciones gracias a que permiten un mayor acceso localizado a memoria disminuyendo los fallos de página. Además, no se produce un aumento del coste de almacenamiento (sección 4).
- Un proceso de construcción de la representación multirresolucion genérico, es decir, independiente de la herramienta de simplificación utilizada (sección 5).
- Un algoritmo mejorado que disminuye los tiempos de extracción de aproximaciones con nivel de detalle uniforme (sección 6).
- Un algoritmo que extrae aproximaciones con nivel de detalle variable, independiente de los criterios de simplificación o refinamiento, y un conjunto de optimizaciones que permiten aumentar sus prestaciones (sección 7).

Notación. Se puede definir un modelo de superficie poligonal, M, como la tupla $\{V,T\}$, donde V es un conjunto de n puntos ordenados $\{v_0,v_1,...,v_{n-1}\}$ donde cada punto $v_i=(x_i,y_i,z_i)\in\mathbb{R}^3$ y puede ser identificado por un único entero i; y T es un conjunto de m triángulos también ordenados $\{t_0,t_1,...,t_{m-1}\}$ donde cada triángulo $t_j=(p,q,r)$ es una lista ordenada de los tres índices que identifican sus vértices (v_p,v_q,v_r), y puede también ser identificado por un único entero j.

2. TRABAJO RELACIONADO

En general, la idea del modelado multirresolución no es una idea nueva. Hace 20 años ya aparecieron los primeros trabajos principalmente aplicados a superficies de terreno en simuladores de vuelo. Sin embargo, el modelado multirresolución de superficies poligonales en general es relativamente reciente. Los modelos multirresolución que aparecieron inicialmente estaban basados en un número de aproximaciones relativamente pequeño (normalmente, entre 5 y 10 [19]) y se les ha denominado modelos multirresolución discretos. Hoy en día existen diversas aplicaciones que hacen uso de los modelos discretos como, por ejemplo, *Open Inventor* [50], *IRIS Performer* [46] o *VRML* [2]. Más tarde, aparecieron los modelos multirresolución continuos que, al almacenar un elevado número de aproximaciones, permiten que el nivel de detalle se adapte con mayor exactitud a los requisitos de la aplicación.

Independientemente del tipo de modelo multirresolución, siempre que haya que construir una representación multirresolución es necesario disponer de un método de simplificación. Un método de simplificación convierte, de forma automática, una superficie poligonal con alto grado de detalle en una aproximación formada por un número menor de polígonos. Es evidente la estrecha relación entre el modelado multirresolución y la simplificación. El modelado multirresolución necesita de métodos de simplificación que, mediante su aplicación repetida, generen las distintas aproximaciones. La necesidad de utilizar métodos de simplificación también aparece en otros campos de la informática gráfica y también en otras áreas como la visión por ordenador, cartografía, análisis de elementos finitos, etc. Esto ha producido el desarrollo de una gran diversidad de métodos. Cada uno de estos métodos ha sido motivado por unas razones diferentes y, sin embargo, en algunas ocasiones llegan a aparecer coincidencias en la forma de resolver el problema [25]. Varios autores han realizado trabajos muy completos con el fin de clasificar y analizar los distintos métodos de simplificación [12, 25, 39, 21]. También se han presentado trabajos que muestran comparaciones de las prestaciones entre diferentes métodos [6, 4, 34, 35], y se han creado herramientas que miden la calidad de las mallas obtenidas como resultado de la simplificación [7].

Puppo y Scopigno [39] clasifican los modelos continuos según su estructura: *modelos basados en árboles* y *modelos históricos o evolucionarios*. Los modelos basados en árboles se basan en subdivisiones anidadas, de acuerdo a la idea básica de que una aproximación puede ser refinada mediante la subdivisión recursiva de una región en un conjunto de regiones más pequeñas y que la jerarquía de estas regiones anidadas puede describirse mediante un árbol. Los modelos históricos se basan en el proceso de refinamiento o simplificación para su construcción y almacenan la evolución que sufre la malla a través de dicho proceso.

Aparte de esta clasificación, otros autores [28, 21] han preferido utilizar como criterio el tipo de superficie al que son aplicables. Así, dividen a los modelos continuos en modelos para *campos de altura* y modelos para *superficies arbitrarias*. Debido a las características del modelo multirresolución que se presenta en esta tesis, también se ha preferido clasificar a los modelos continuos según este criterio.

Campos de altura. Se encuentran entre los tipos de superficies más simples. Se definen como el conjunto de puntos que satisfacen una ecuación de la forma $z = f(x,y)$ donde x e y varían en un subconjunto del plano Cartesiano. Se han desarrollado numerosos métodos para este caso particular de superficie [13, 25], principalmente debido a su aplicación en sistemas de información geográfica y simuladores de vuelo [39]. De hecho, los primeros métodos de refinamiento y simplificación [18, 32] se desarrollaron como herramientas de compresión, diseñadas con el propósito explícito de construir representaciones aproximadas y compactas de terrenos mediante mallas irregulares de triángulos. Los modelos multirresolución que se aplican a este tipo de superficies utilizan representaciones jerárquicas como *árboles de cuadrantes* [49], *triangulaciones cuaternarias* [36], o *triangulaciones jerárquicas más generales* [33, 5, 13, 47, 8, 28].

Superficies arbitrarias. Comparativamente, hay menos trabajos sobre modelado multirresolución aplicado a este tipo de superficies. Este tipo de modelos se clasifican en: *representaciones incrementales*, que codifican el conjunto de aproximaciones utilizando la de menor detalle y una secuencia lineal de actualizaciones que permiten refinar la superficie; y *representaciones jerárquicas*, que se basan en bosques (conjunto de árboles) donde las hojas representan la aproximación de mayor detalle, los nodos las simplificaciones y las raíces la aproximación de menor detalle.

2.1. Multi-triangulation MT

La Multitriangulación [38, 40] es una estructura que constituye un marco de trabajo general para los modelos multirresolución continuos. En realidad, todos los modelos multirresolución propuestos en la literatura pueden ser interpretados como casos especiales de MT. La idea general que subyace en MT es que cualquier representación multirresolución puede ser construída mediante operaciones locales que progresivamente modifican el objeto original, ya sean mediante refinamiento o simplificación, y que dichas modificaciones locales pueden ser dispuestas en un orden parcial de acuerdo a sus dependencias. Este marco de trabajo es independiente del tipo de modificación local realizada, pudiendo ser cualquier combinación de operaciones genéricas que modifiquen la superficie localmente, tales como eliminación de un vértice, contracción de una arista, eliminación de un triángulo, etc. En [31, 15, 16] se describen diversas implementaciones de una MT proponiendo varias estructuras de datos alternativas para su codificación.

2.2. Representaciones incrementales

El nombre de representación incremental se debe a que este tipo de modelos codifican el conjunto de aproximaciones utilizando la aproximación de menor detalle, M_{n-1}, y una secuencia lineal de operadores que aplicados en dicho orden reconstruyen todas y cada una de las distintas aproximaciones (en ocasiones, este tipo de representaciones referencian a la aproximación de menor detalle como M_0, por ser la aproximación inicial, y a M_{n-1} como la de mayor detalle). Este tipo de modelos continuos aparecen algunas veces en la literatura con el nombre de *secuencias lineales*.

Las representaciones incrementales han sido desarrolladas principalmente para las aplicaciones de transmisión progresiva, compresión y visualización interactiva con aproximaciones de nivel de detalle uniforme. La mayor parte de los modelos fueron propuestos en los últimos años. Eck et al. [9] presentaron un modelo basado en *wavelets* aplicable a superficies con subdivisión conectada y fue extendido por Certain et al. [3] para incluir, además de la geometría, otros atributos como, por ejemplo, el color. Más tarde, Hoppe presentó el esquema conocido como *Progressive Meshes* (PM) [26], extendido por Popovic y Hoppe [41] para permitir cambios en la topología, y por Hoppe [29] para codificar de forma eficiente un objeto PM. Otros modelos son el de Cignoni et al. [5], Taubin et al. [48], Gueziec et al. [22] y Abadjev et al. [1].

2.3. Representaciones jerárquicas

Las representaciones incrementales sólo son capaces de reproducir aproximaciones que hayan sido generadas por el método de simplificación en el proceso de construcción de la representación multirresolución. El principal objetivo de una representación jerárquica es representar a un objeto utilizando diferentes niveles de detalle coexistiendo al mismo tiempo en distintas zonas de la superficie. Se basan en bosques (conjunto de árboles) donde las hojas representan la aproximación de mayor detalle y los nodos representan las simplificaciones.

Este tipo de estructura jerárquica fue desarrollada independientemente por distintos autores [51, 27, 37]. La mayor parte de los esquemas presentados hasta el momento utilizan métodos de simplificación basados en contracción de aristas generando árboles binarios. Los trabajos presentados por Xia et al. [51], Hoppe [27], Guéziec et al. [23], El-Sana et al. [11], El-Sana y Varshney [10] pertenecen a este grupo. Sólo el esquema propuesto por Luebke y Erikson [37] utiliza un método basado en agrupación de vértices para su construcción y, en consecuencia, cada nodo tiene un número arbitrario de hijos.

Una vez construida la representación jerárquica, cada aproximación posible se corresponde con un corte en el árbol. Cualquier cambio en este corte, producirá siempre una nueva aproximación. Manteniendo un corte a través del árbol y ajustándolo incrementalmente a cada cuadro, es fácil adaptar el modelo a las nuevas condiciones exigidas por la aplicación. Sin embargo, expandir o contraer un nodo del árbol de forma arbitraria puede producir aproximaciones erróneas, siendo necesario almacenar las posibles dependencias que existen entre las simplificaciones representadas en la jerarquía y realizar algún tipo de comprobación con el fin de que dichas dependencias se respeten. Esta operación suele ser costosa, tanto desde un punto de vista espacial como temporal y, en general, va a condicionar las prestaciones del modelo multirresolución [10].

3. INTRODUCCIÓN A MOM

El esquema multirresolución MOM se clasifica dentro de las llamadas representaciones incrementales. Sean M y M^r la suprficie poligonal original y su representación multirresolución, respectivamente. Se define M^r como:

$$M^r = \{V^r, T^r\} \tag{1}$$

donde V^r es el conjunto de todos los vértices y T^r es el conjunto de todos los triángulos utilizados para representar cualquiera de las diferentes aproximaciones que almacena M^r.

Para construir la representación multirresolución M^r de n aproximaciones, es necesario aplicar de forma iterativa un método de simplificación $n-1$ veces que, partiendo de $M = M_0$, genere la secuencia de aproximaciones M_1, M_2,...,M_{n-1}. Cada simplificación S_i, $0 \leq i < n-1$, genera una nueva aproximación M_{i+1} y puede ser representada por la tupla $S_i = \{\mathcal{V}_i, \mathcal{T}_i, \mathcal{V}_i', \mathcal{T}_i'\}$ donde $\mathcal{V}_i$ y $\mathcal{T}_i$ son los conjuntos de vértices y triángulos que son eliminados de M_i, y $\mathcal{V}_i'$ y $\mathcal{T}_i'$ son los conjuntos de vértices y triángulos que son añadidos a M_i para generar, finalmente, M_{i+1}. De esta forma, el objeto resultante, M_{i+1}, se podría expresar de la siguiente forma:

$$M_{i+1} = (M_i - \{\mathcal{V}_i, \mathcal{T}_i\}) \cup \{\mathcal{V}_i', \mathcal{T}_i'\},\ 0 \leq i < n-1 \tag{2}$$

Dado que M^r almacena todos los vértices y triángulos que pueden ser utilizados para representar cualquier aproximación, M^r puede definirse como:

$$M^r = \bigcup_{i=0}^{n-1} M_i, n \geq 1 \tag{3}$$

De las ecuaciones 1 - 3, se obtiene que M^r puede ser expresado en función de la aproximación de mayor detalle, $M = M_0$, y de todos los vértices y triángulos generados en cada iteración del proceso de simplificación:

$$V^r = V_0 \cup \mathcal{V}_0' \cup \mathcal{V}_1' \cup \ldots \cup \mathcal{V}_{n-2}' = V_0 \cup \bigcup_{i=0}^{n-2} \mathcal{V}_i' \tag{4}$$

$$T^r = T_0 \cup \mathcal{T}_0' \cup \mathcal{T}_1' \cup \ldots \cup \mathcal{T}_{n-2}' = T_0 \cup \bigcup_{i=0}^{n-2} \mathcal{T}_i' \tag{5}$$

o también, en función de la aproximación de menor detalle, M_{n-1}, y de los vértices y triángulos eliminados en cada iteración del proceso de simplificación:

$$V^r = \mathcal{V}_0 \cup \mathcal{V}_1 \cup \ldots \cup \mathcal{V}_{n-2} \cup V_{n-1} = \bigcup_{i=0}^{n-2} \mathcal{V}_i \cup V_{n-1} \tag{6}$$

$$T^r = \mathcal{T}_0 \cup \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_{n-2} \cup T_{n-1} = \bigcup_{i=0}^{n-2} \mathcal{T}_i \cup T_{n-1} \tag{7}$$

La idea básica de una representación MOM se basa en las ecuaciones 6 y 7. Es decir, almacenar de forma ordenada las secuencias de vértices y triángulos eliminados, $\mathcal{V}_i$ y $\mathcal{T}_i$, $0 \leq i < n-1$, más la aproximación de menor detalle $M_{n-1} = \{V_{n-1}, T_{n-1}\}$. Sin embargo, esta información será necesaria pero nunca suficiente para que un algoritmo sea capaz de extraer una triangulación correcta y con el nivel de detalle requerido. Además, es preciso almacenar las relaciones de dependencia, lo que puede suponer un importante aumento del coste de almacenamiento [10]. La estructura de datos que se corresponde con este esquema sería un grafo acíclico dirigido [40] donde cada triangulación se correspondería con un nodo del grafo y cada relación de dependencia con un arco.

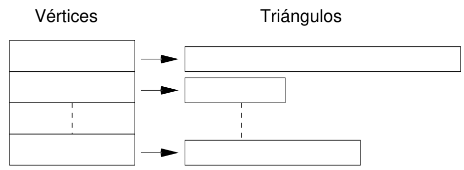


Figura 1: *Esquema de la estructura de datos basada en listas utilizada por una representación multirresolución MOM. En ella, cada vértice tiene asociada una lista de triángulos.*

El esquema MOM se caracteriza por ser capaz de resolver este problema con apenas un incremento del coste de almacenamiento gracias a la disposición ordenada de los vértices y triángulos. Esta ordenación permite que no haya que almacenar de forma explícita las dependencias. Sin embargo, sí es necesario:

- Establecer una relación entre los vértices y triángulos eliminados en la misma simplificación, S_i. Esta relación se realiza dividiendo el conjunto de triángulos, $\mathcal{T}_i$, en tantos subconjuntos como vértices hay en $\mathcal{V}_i$, y asociando cada subconjunto de $\mathcal{T}_i$ a cada vértice de $\mathcal{V}_i$. La única norma a seguir es que cada subconjunto esté formado por triángulos que utilicen el vértice al cual es asociado.
- Cada triángulo almacenado estará identificado por un valor que representa su posición en la secuencia ordenada dada por la ecuación 5. Este valor se utilizará en el proce-

so de extracción de una aproximación para conocer si un triángulo pertenece o no al LoD requerido.

Consecuentemente, la estructura de datos pasa de ser un grafo acíclico dirigido a una estructura basada en listas. En la figura 1 se muestra un esquema de dicha estructura en la que se puede observar una lista de vértices donde cada uno de ellos tiene asociada una lista de triángulos. Evidentemente, cada triángulo también apunta a elementos de la lista de vértices pero estos punteros han sido omitidos para mayor claridad.

4. ESTRUCTURA DE DATOS BÁSICA

La estructura de datos de un objeto MOM sigue el esquema que se muestra en la figura 2, donde todas las listas de triángulos que se pueden observar en la figura 1 se almacenan de forma consecutiva en una única lista. En la figura 3 se muestra la estructura de datos que se propone. Tres son las listas necesarias: lista de vértices (campo *listaVertices*), encargada de almacenar los vértices del objeto MOM; lista de triángulos (campo *listaTriangulos*), encargada de almacenar los triángulos; y la lista de control (campo *listaControlLod*), que almacena información adicional necesaria para recuperar un LoD dado.

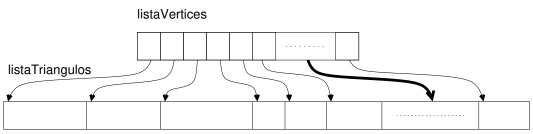


Figura 2: *Esquema de la estructura de datos básica de una representación MOM (sólo se han dibujado las listas de vértices y triángulos).*

Cada vértice en una representación MOM consta de sus coordenadas (campo *coordenadas*) y de un índice (campo *primerTriangulo*) de la lista *listaTriangulos* correspondiente al primer elemento de su lista asociada de triángulos (en la sección 5 se explica el proceso de construcción y, exactamente, qué conjunto de triángulos se asocia con cada vértice y en qué orden se almacenan). Ya que las listas de triángulos se almacenan consecutivamente en la lista *listaTriangulos*, no es necesario que cada vértice almacene un puntero al último elemento de su lista ni la longitud de ésta. De esta forma, es posible conocer el último triángulo asociado a un vértice preguntando por el primer triángulo asociado al vértice siguiente.

Asimismo, el hecho de que cada vértice apunte a un conjunto de triángulos permite que cada triángulo sea representado justo por sólo los otros dos vértices (campos *v1* y *v2* codificados como punteros a elementos de tipo *Vertice*). Además, es necesario que cada triángulo almacene un tercer dato, campo *id*, que se utilizará como identificador para

```
estructura Vertice {
  real coordenadas[3];
  entero primerTriangulo;
};
estructura Triangulo {
  entero id;
  estructura Vertice *v1, *v2;
};
estructura ControlLod {
  entero verticesEliminados;
  entero triangulosGenerados;
};
estructura MOM {
  entero nVertices, nTriangulos, nLods;
  estructura Vertice listaVertices[nVertices];
  estructura Triangulo listaTriangulos[nTriangulos];
  estructura ControlLod listaControlLod[nLods];
};
```

Figura 3: *Estructura de datos básica de una representación MOM.*

conocer si el triángulo pertenece o no al nivel de detalle requerido, y representa su posición en la secuencia ordenada que se mostró en la ecuación 5.

Por último, cada elemento de la lista de control contiene información específica que permite recuperar una determinada aproximación. Estos datos están muy ligados al proceso de construcción de un objeto MOM (explicado en la siguiente sección). En pocas palabras, para cada iteración del proceso de simplificación un número concreto de vértices se eliminan y, en consecuencia, el agujero que se produce se triangula utilizando un número concreto de triángulos. Ambas cantidades se almacenan en los campos *verticesEliminados* y *triangulosGenerados* respectivamente.

4.1. Coste de almacenamiento

Para realizar el estudio del coste de almacenamiento de una representación multirresolución MOM, M^r, se va a suponer que el coste de almacenar un valor entero, real o puntero es de una palabra para todos ellos. De esta forma, un elemento de tipo *Vertice* tendrá un coste de 4 palabras, un elemento de tipo *Triangulo* tendrá un coste de 3 palabras, y, por último, un elemento de tipo *ControlLod* tendrá un coste de 2 palabras. Ya que un objeto MOM almacena tres listas, una por cada uno de estos tres tipos, se obtiene un coste total de $4|V^r|+3|T^r|+2n$, donde $|V^r|, |T^r|$ y n son, respectivamente, el número de vértices, triángulos y LoDs almacenados en M^r.

M^r almacena de forma explícita cada uno de los vértices y triángulos que pueden utilizarse para representar a M en cualquiera de sus n LoDs. De esta forma, se puede definir

V^r y T^r como:

$$V^r = \bigcup_{i=0}^{n-1} V_i \qquad T^r = \bigcup_{i=0}^{n-1} T_i \tag{8}$$

donde V_i y T_i son los conjuntos de vértices y triángulos, respectivamente, de M_i, $0 \le i \le n-1$. En el peor caso, los V_i serán conjuntos disjuntos, y lo mismo los T_i, obteniendo que $|V^r|$ es $O(|V|^2)$ y $|T^r|$ es $O(|T|^2)$.

Para estudiar el caso medio, supóngase que por término medio existen 6 triángulos alrededor de cada vértice de la superficie [30], y se cumple que

$$|V_i| \simeq \frac{|T_i|}{2}, \quad 0 \le i < n-1 \tag{9}$$

Si cada simplificación realizada consiste en la eliminación de un vértice de la superficie entonces $|V^r| = |V|$ (no se añaden vértices) y $|T^r| = |T| + 4n$ (se añaden 4 triángulos por simplificación realizada). Como n está acotado por el número de vértices de M, se obtiene que $|T^r| \le |T| + 4|V|$, y por la ecuación 9 se obtiene que $|T^r| \le 3|T|$. Análogamente, si la simplificación se basa en la contracción de una arista, $|V| \le |V^r| \le 2|V|$ y $3|T| \le |T^r| \le 5|T|$ dependiendo de si el punto de contracción coincide con uno de los vértices de la arista o no.

En la tabla 1 se resumen los costes de almacenamiento teóricos obtenidos para el caso medio. El coste de almacenamiento del modelo original es $3|V| + 3|T|$ utilizando una representación de lista de vértices [17]. Mediante la ecuación 9, se obtiene que la representación multirresolución es 2'67 veces mayor si se genera mediante un método de simplificación basado en eliminación de vértices, y entre 2'67 y 4'44 veces mayor si se utiliza un método de simplificación basado en contracción de aristas.

4.2. Resultados

Para crear las representaciones multirresolución se ha utilizado el método de simplificación propuesto por Garland y Heckbert [20] basado en contracción de aristas obligando a que el punto de contracción coincida con uno de los dos vértices de la arista. En la tabla 2 se resumen las características y los costes de almacenamiento de los objetos utilizados en los experimentos. Para cada objeto se indica su número de vértices y triángulos, y su coste de almacenamiento utilizando una representación de lista de vértices y suponiendo que una palabra (entero, real o puntero) tiene un coste de 4 bytes. Respecto a la representación MOM se indica el número de aproximaciones, n, el número de triángulos (el de vértices es el mismo que el de la representación original por obligar a que el punto de contracción coincida con uno de los dos vértices de la arista) y el coste de almacenamiento. Además, se incluye una columna con la relación entre los costes de almacenamiento de ambas representaciones. Los resultados muestran que el coste de una representación MOM es entre 2'3 y 2'7 veces mayor que el del modelo original, que aproximadamente coincide con el coste teórico obtenido para el caso medio, que es 2'67 (ver tabla 1).

En la tabla 3 se muestra una comparación de los costes de almacenamiento para las representaciones multirresolución MOM, PM [29] y MT [16]. Para hacer esta comparación se han utilizado las representaciones propuestas por los autores diseñadas a propósito para visualización interactiva. Es interesante destacar que el esquema PM, a pesar de codificar de forma implícita parte de los datos, restringir su uso a superficies *manifold* y no poder extraer aproximaciones con nivel de detalle variable, requiere un coste de almacenamiento equivalente al utilizado en las representaciones MOM. Sin embargo, el modelo VDPM [27], versión de PM que es capaz de extraer aproximaciones con nivel de detalle variable, necesita de, por ejemplo, 7'8MB para el objeto *Bunny*, coste bastante superior al de una representación MOM o PM.

Table 3: *Costes de almacenamiento para las representaciones multirresolución MOM, PM y MT (en megabytes).*

	MOM	PM	MT
Cow	0'237	0'250	0'321
Sphere	1'306	1'318	1'694
Bunny	2'876	2'993	3'854
Phone	6'940	7'144	9'187
Isis	15'667	16'770	20'784
Buda	43'957	46'753	60'142

Los datos para el modelo MT se han obtenido utilizando el coste teórico publicado por los autores. Dicho coste es de aproximadamente $29v$ palabras donde v es el número de vértices del modelo original. Para obtener este coste los autores han realizado un empaquetado de los datos y así, por ejemplo, sólo han contado 21 bits por puntero, en lugar de los 32 bits que se han calculado para las representaciones MOM y PM. Sin embargo, los autores afirman un coste real de almacenamiento de 6'5MB para el modelo *Marcy* (de similares características al modelo *Bunny*) formado por 35.162 vértices y 69.718 triángulos. Este coste se aleja bastante de los 3'9MB que se obtienen de forma teórica. Aun así, los resultados obtenidos muestran unos costes de almacenamiento claramente superiores tanto a MOM como a PM.

Hierarchical Dynamic Simplification [37] no ha sido incluido en la tabla porque los autores no han presentado un análisis del coste de almacenamiento de su modelo. Sin embargo, sí que han publicado algunos datos concretos como, por ejemplo, que el modelo *Bunny* necesita alrededor de cinco megabytes comprimidos utilizando el compresor de ficheros *gzip*. Este coste es muy alto comparado con cualquiera de los otros tres modelos.

Table 1: *Costes teóricos de almacenamiento en el caso medio.*

Cost: $4\|V^r\|+3\|T^r\|+2n$	$\|V^r\|$	$\|T^r\|$	n	Total	Ratio
Eliminación de vértices	$\|V\|$	$3\|T\|$	$\|V\|$	$6\|V\|+9\|T\|$	2'67
Contracción de aristas (coincide)	$\|V\|$	$3\|T\|$	$\|V\|$	$6\|V\|+9\|T\|$	2'67
Contracción de aristas (no coincide)	$2\|V\|$	$5\|T\|$	$\|V\|$	$10\|V\|+15\|T\|$	4'44

Table 2: *Lista de objetos con sus características y costes de almacenamiento.*

	Original			MOM			Ratio
	Vértices	Triángulos	MB.	n	Triángulos	MB.	
Cow	2.905	5.804	0'100	2.803	14.982	0'237	2'37
Sphere	15.315	30.624	0'526	15.264	83.486	1'306	2'48
Bunny	34.835	69.451	1'193	33.990	182.192	2'876	2'41
Phone	83.045	165.963	2'850	81.668	441.181	6'940	2'44
Isis	187.871	375.736	6'450	187.370	993.559	15'667	2'43
Steve	372.275	739.628	12'725	358.392	2.195.898	33'545	2'64
Buda	543.653	1.085.636	18'646	543.106	2.754.083	43'957	2'36

5. PROCESO DE CONSTRUCCIÓN

Dos son las etapas que conforman el proceso de construcción. La primera involucra las tareas que se han de repetir para cada iteración del proceso de simplificación. La segunda agrupa un conjunto de operaciones que, a modo de postproceso, terminan de construir M^r.

Inicialización. Previamente al inicio de este proceso es necesario inicializar M^r con el primer nivel de detalle, $M_0 = M$. Esto consiste únicamente en actualizar los campos $verticesEliminados = 0$ y $triangulosGenerados = |T_0|$ del primer elemento de la lista $listaControlLod$. La lista de vértices y la lista de triángulos permanecen vacías (figura 4).

Primera etapa. Para cada iteración i, $0 \leq i < n-2$, del proceso de simplificación, se realizan los siguientes cuatro pasos (figura 4):

1. Añadir los vértices eliminados $\mathcal{V}_i$ a M^r. De esta forma, la lista de vértices contendrá la secuencia ordenada de conjuntos de vértices eliminados $\mathcal{V}_0, \mathcal{V}_1, ..., \mathcal{V}_{n-2}$.
2. Almacenar los triángulos eliminados $\mathcal{T}_i$ en M^r. De igual forma, la lista de triángulos contendrá la secuencia ordenada de conjuntos de triángulos eliminados $\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_{n-2}$. Para realizar esto, han de seguirse las siguientes indicaciones:

   a. Dividir $\mathcal{T}_i$ en j subconjuntos disjuntos, donde $j = |\mathcal{V}_i|$, de tal forma que los triángulos que forman cada subconjunto tengan en común un vértice de $\mathcal{V}_i$.
   b. Ordenar los triángulos de cada subconjunto utilizando el campo id.
   c. Almacenar cada subconjunto de triángulos en el mismo orden en que han sido almacenados los vértices $\mathcal{V}_i$.
3. Actualizar el campo $primerTriangulo$ para cada vértice de $\mathcal{V}_i$ de forma que cada uno apunte al primer triángulo de su subconjunto de caras asociado.
4. Actualizar la lista de control almacenando el número total de vértices eliminados y el total de triángulos generados hasta el momento.

Segunda etapa. En este momento, M^r está almacenando todas las aproximaciones desde M_0 hasta M_{n-2} ambas inclusive. Las tareas a realizar serán (figura 5):

1. Añadir M_{n-1} a M^r. Esta tarea se realiza de la misma forma que se indica en la primera etapa repitiendo los pasos 1, 2 y 3 y teniendo en cuenta que los vértices sin un subconjunto de triángulos asociado se deben almacenar siempre al final de la lista.
2. Actualizar los dos punteros a vértices que cada triángulo en *listaTriangulos* contiene. Esto es necesario ya que los vértices cambian de posición cuando son almacenados en V^r respecto a la posición que ocupaban en V.

Tres hechos del proceso de construcción van a influir de forma considerable en la eficiencia de los algoritmos que han de recorrer la estructura de datos para extraer aproximaciones con el nivel de detalle requerido.

1. Los vértices en *listaVertices* quedan ordenados respecto a la aproximación en la que son eliminados.
2. Los triángulos asociados a cada vértice se encuentran ordenados respecto a la aproximación en la que son generados.
3. Cada triángulo sólo almacena en *listaTriangulos* dos de sus vértices.

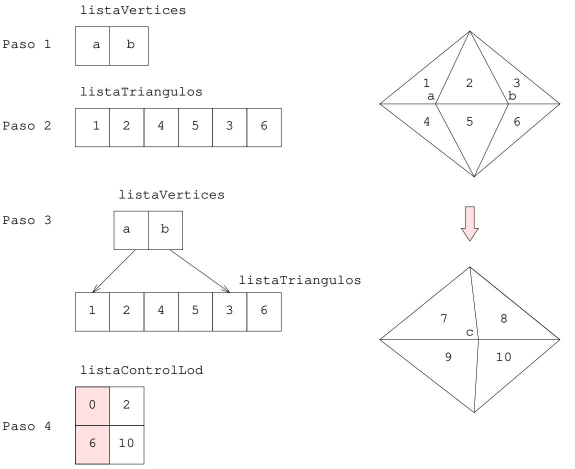


Figura 4: *Ejemplo de construcción. Inicialización (en gris claro) y pasos 1 a 4 de la primera etapa.*

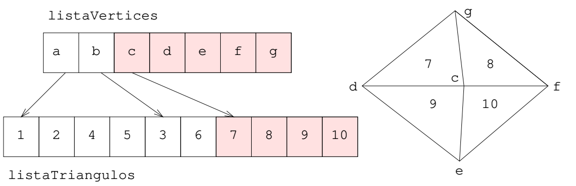


Figura 5: *Ejemplo de construcción. Segunda etapa en gris claro.*

Además, dos pequeños detalles pero a su vez también muy importantes son que los vértices eliminados siempre tienen algún triángulo asociado, y que los vértices de la aproximación de menor detalle están ordenados de forma que aquellos que no tienen triángulos asociados figuran al final de la lista.

6. VISUALIZACIÓN DE LODS UNIFORMES

Los algoritmos de extracción de aproximaciones de nivel de detalle uniforme recorren la estructura de datos hasta alcanzar la aproximación requerida bien partiendo de la aproximación de menor nivel de detalle y refinando [26, 37, 52, 40], o bien partiendo de la aproximación de mayor nivel de detalle y simplificando [33]. El algoritmo que se presenta, gracias a la ordenación y a un almacenamiento explícito de los datos, es capaz de recuperar los datos que forman la aproximación requerida sin necesidad de partir de una aproximación determinada ni realizar una secuencia de simplificaciones o refinamientos.

Sea M_L la aproximación requerida. El algoritmo que se propone calcula, en primer lugar, el número de triángulos, NT, que pertenecen a M_L. Después, recorre la lista de vértices y, para cada uno, su lista de triángulos asociados pintando aquellos que pertenecen a M_L. El algoritmo termina cuando se han visualizado los NT triángulos. A continuación se describen cada uno de estos pasos.

6.1. Cálculo del número de triángulos

De la lista de control, *listaControlLod*, se obtienen los dos valores establecidos para M_L, es decir, el número de vértices eliminados, VE, y el número de triángulos generados, TG. Si VE es el número de vértices eliminados se sabe que estos estarán en las VE primeras posiciones de *listaVertices*. Además, para cada uno de los vértices que no pertenecen a M_L tampoco pertenecerán sus triángulos asociados. Por lo tanto, restando a TG la posición del primer triángulo en *listaTriangulos* que puede pertenecer a M_L (*listaVertices[VE].primerTriangulo*) se obtiene NT. En la figura 6 se muestran las operaciones correspondientes al cálculo del número de triángulos.

6.2. Visualización de los triángulos

El algoritmo recorre la lista de vértices a partir del vértice VE ya que este es el número de vértices eliminados. Respecto a qué triángulos pintar, si para una aproximación dada, M_L, se sabe que el número de triángulos generados es TG entonces nunca un triángulo con un identificador igual o superior a TG pertenecerá a M_L. Además, si recorriendo los triángulos asociados a un vértice se encuentra un triángulo que no pertenece a M_L entonces no es necesario continuar recorriendo la lista ya que los triángulos asociados a un vértice están almacenados en orden respecto su identificador.

En la figura 6 se muestra el algoritmo que realiza las tareas descritas y en la figura 7 se muestran ocho aproximaciones distintas de un objeto MOM.

Es interesante hacer varias observaciones sobre el algoritmo presentado y que van a influir decisivamente en sus prestaciones. En primer lugar su sencillez, debida principalmente a la propia estructura de datos de un objeto MOM: listas de vértices y triángulos ordenadas, y no existen vértices sin triángulos asociados excepto los que se encuentran al final de la lista. En segundo lugar, las siguientes características:

- La lista de vértices se recorre siempre, a partir de un vértice dado, de forma secuencial.
- La lista de triángulos se recorre siempre de forma lineal y hacia adelante.
- El vértice común a un conjunto de triángulos sólo es leído una vez, quedando por leer justo los otros dos vértices de cada triángulo.
- Todas las operaciones aritméticas se reducen a incrementos o decrementos en uno.

6.3. Coste temporal

A grandes rasgos, el funcionamiento del algoritmo es el siguiente: dada una aproximación, M_L, se calcula el número

```
// PRIMERO: cáculo del número de triángulos NT
TG= listaControlLod[L].triangulosGenerados;
VE= listaControlLod[L].verticesEliminados;
NT= TG - listaVertices[VE].primerTriangulo;

// SEGUNDO: visualización de los triángulos
i= VE;
primero= listaVertices[i].primerTriangulo;
mientras NT > 0 hacer
  // Recorrer la lista de triángulos asociados a vi
  ultimo= listaVertices[i+1].primerTriangulo;
  pvi= direccion (listaVertices[i]);
  mientras primero < ultimo y
  listaTriangulos[primero].id < TG hacer
    pintaTriangulo (pvi,
                    listaTriangulos[primero].v1,
                    listaTriangulos[primero].v2);
    NT= NT - 1;
    primero= primero + 1; // siguiente triángulo
  fin mientras
  i= i + 1; // siguiente vértice
  primero= ultimo;
fin mientras
```

Figura 6: *Algoritmo de visualización de una aproximación, M_L, con nivel de detalle uniforme.*

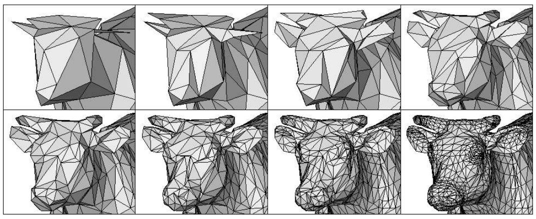


Figura 7: *De izquierda a derecha y de arriba a abajo se muestran ocho aproximaciones del objeto* Cow *correspondientes a los siguientes niveles de detalle: 0 (el más bajo), 0'03, 0'07, 0'14, 0'21, 0'38, 0'64 y 1'0 (el más alto).*

de triángulos, $NT = |T_L|$, que pertenecen a M_L, y se recorre la lista de vértices (bucle externo) y a su vez, para cada uno de ellos, su lista de triángulos asociados (bucle interno). El algoritmo termina cuando los $|T_L|$ triángulos son visualizados.

Respecto al bucle externo, en el peor caso, el algoritmo visita lo que queda de lista de vértices, es decir, $|\mathcal{V}^r| - i$ donde el vértice i es a partir del cual comienza a recorrer de forma secuencial la lista de vértices. Respecto al bucle interno, el algoritmo siempre visualiza las caras correspondientes a M_L, es decir, $|T_L|$. Sumando ambos valores se obtiene $|\mathcal{V}^r| - i + |T_L|$. En el peor caso $|\mathcal{V}^r| - i$ es de orden cuadrático respecto a $|V_L|$, y en consecuencia, el algoritmo tiene un coste $O(|V_L|^2)$. Sin embargo, si el método de simplificación actúa eliminando vértices o contrayendo aristas, $|\mathcal{V}^r| - i$ es de orden lineal respecto a $|V_L|$, y el coste es $O(|V_L| + |T_L|)$ que puede simplificarse en $O(|T_L|)$ ya que en la práctica los objetos cumplen que $|T_L| \approx 2|V_L|$.

6.4. Resultados

Los experimentos han sido realizados en una estación de trabajo Silicon Graphics RealityEngine 2 con un MIPS R10000 a 194 MHz y 256 Mb RAM. Estos experimentos han consistido en medir tiempos de extración. El tiempo de extracción es el tiempo que invierte el algortimo de visualización en obtener los datos que forman la aproximación requerida. No se incluye el tiempo que que el sistema gráfico utiliza en pintar estos datos. El nivel de detalle se representa mediante un rango $[0..1]$, donde 0 es el nivel de menor detalle y 1 el de mayor. En la tabla 4 se muestran los tiempos de extracción y el número de triángulos que forman cada una de las aproximaciones obtenidas para cuatro objetos diferentes. En la figura 8 se muestra el comportamiento del algoritmo de visualización para estos cuatro objetos. Se puede observar que el algoritmo tiene un comportamiento lineal respecto al nivel de detalle de la aproximación requerida coincidiendo con el resultado obtenido en el análisis teórico.

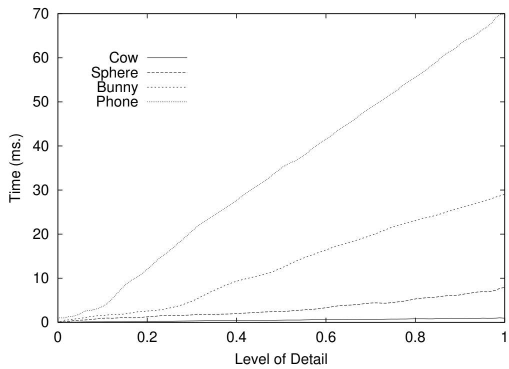


Figura 8: *Relación entre el nivel de detalle y el tiempo de extracción (ms.) para cuatro objetos MOM.*

En la tabla 5 se muestran los tiempos obtenidos para una representación MOM del modelo *Bunny* y una representación MT del modelo *Marcy* (modelo de similares características al modelo *Bunny* formado por 35.162 vértices y 69.718 triángulos). Este es el único modelo para el que los autores de la implementación de MT han publicado tiempos de extracción. Los resultados muestran unos tiempos mucho menores para MOM a la hora de extraer representaciones de similar complejidad. No se ha incluido ninguna representación HDS porque sus autores no han publicado tiempos de extracción para aproximaciones de nivel de detalle uniforme.

Se ha realizado una implementación del modelo PM a fin de compararlo con MOM. La implementación de PM ha sido optimizada hasta conseguir las prestaciones publicadas

Table 4: *Tiempos de extracción (ms.) para diferentes aproximaciones de cuatro objetos MOM.*

	lod 1		lod 0'5		lod 0'25		lod 0'1	
	Tri.	Tiempo	Tri.	Tiempo	Tri.	Tiempo	Tri.	Tiempo
Cow	5.804	1'0	3.002	0'5	1.600	0'2	760	0'1
Sphere	30.624	7'1	15.362	2'9	7.730	1'7	3.152	0'5
Bunny	69.451	29'2	35.482	12'0	18.497	4'0	8.307	1'6
Phone	165.963	69'0	84.297	34'0	43.468	16'0	18.972	5'0

Table 5: *Tiempos de extracción (ms.) de la representación MOM del modelo* Bunny *y de la representación MT del modelo* Marcy.

	Tri.	Tiempo	Tri.	Tiempo	Tri.	Tiempo	Tri.	Tiempo
MT Marcy	69.718	462'32	37.073	252'05	8.438	39'05	1.281	5'41
MOM Bunny	69.451	29'20	37.046	13'40	8.444	1'40	1.557	0'40

por el autor [29]. La figura 9 muestra los resultados obtenidos para los objetos *Bunny* y *Phone*. Para cada uno se muestra una gráfica que representa el tiempo de extracción cuando la aproximación requerida es la $M_{n/2}$ y la última solicitada fue cualquier otra. Se puede observar cómo PM mejora su comportamiento al disminuir la distancia entre dos aproximaciones solicitadas consecutivamente, y cómo MOM es independiente de dicha distancia. PM consigue mejorar apenas el comportamiento de MOM sólo cuando dicha distancia es muy pequeña, mientras que muestra un comportamiento mucho peor en cuanto esa distancia aumenta.

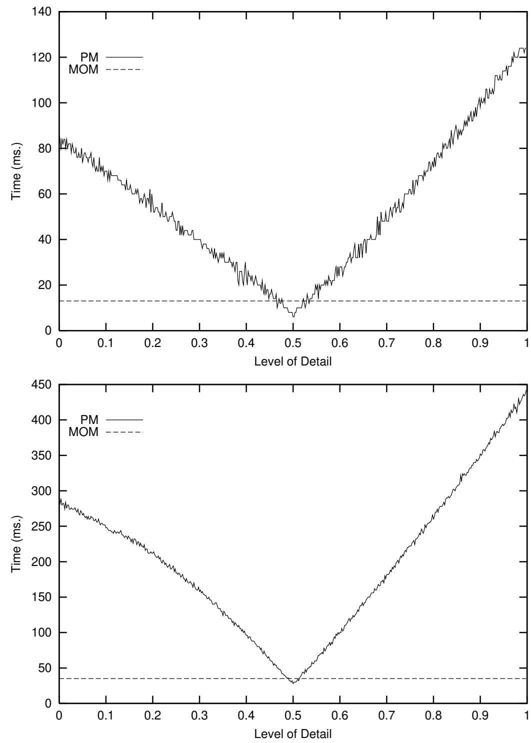


Figura 9: *Comparación de MOM y PM con los objetos Bunny (arriba) y Phone (abajo).*

7. VISUALIZACIÓN DE LODS VARIABLES

Se va a describir con un ejemplo el significado de nivel de detalle variable. En la figura 10, una esfera colocada interactivamente por un usuario señala la zona a representar con alto detalle. Esa zona se dibuja utilizando triángulos que pertenecen a la aproximación de mayor detalle (gris claro). El resto del objeto se pintará con el menor detalle posible manteniendo siempre la conectividad en la superficie. Así, se pueden observar dos conjuntos de triángulos: un conjunto formado por triángulos que pertenecen a la aproximación de menor detalle (gris oscuro) y otro conjunto formado por triángulos (gris medio) que realizan la transición entre los que pertenecen a la aproximación de mayor detalle y los de la de menor detalle, asegurando siempre la conectividad.

En general, los esquemas multirresolución que permiten representar a un objeto mediante aproximaciones con nivel de detalle variable se basan en estructuras de datos jerárquicas de tipo árbol [51, 27, 37] o grafo [16], las cuales típicamente presentan un mayor coste de almacenamiento que las representaciones incrementales. Por ejemplo, Hoppe cambió la representación incremental [26] de Progressive Meshes a una representacion jerárquica [27] para poder extraer aproximacio-

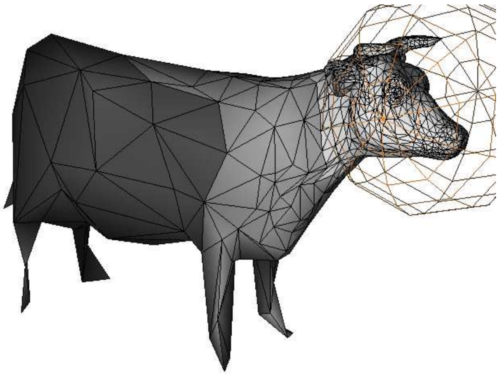


Figura 10: *Un ejemplo de nivel de detalle variable.*

nes de nivel de detalle variable en tiempo real. Una importante ventaja del algoritmo que se propone es que utiliza la misma representación incremental de MOM, sin tener que reorganizar los datos en una nueva estructura de datos ni almacenar nueva información y así mantener un coste de almacenamiento reducido. Consecuentemente, el mismo proceso de construcción crea una representación de la cual se pueden extraer aproximaciones de nivel de detalle tanto uniforme como variable, sin necesidad de un proceso adicional que incrementaría el coste y la complejidad del proceso de construcción.

Un problema típico a la hora de extraer niveles de detalle variables es el de los plegamientos en la superficie sino se tienen en cuenta las posibles dependencias entre las distintas triangulaciones. Las listas de dependencias fueron introducidas por Xia et al. [52] y mejoradas posteriormente por Hoppe [27] para evitar este tipo de problemas. Sin embargo, almacenar estas listas de forma explícita resulta caro y supone un incremento en el coste computacional al chequearlas en tiempo de ejecución [10]. Otra ventaja del algoritmo que se propone es que no necesita almacenar listas de dependencias ya que el propio objeto MOM las codifica de forma implícita.

Extraer una representacion con nivel de detalle variable exige de un criterio o conjunto de criterios que decidan que parte del objeto simplificar y que parte mantener con detalle. El usario puede indicar de forma interactiva las regiones que deben ser visualizadas con mayor o menor detalle, o la aplicación también puede tomar esa decisión. Diversos autores [27, 52, 37, 10] han definido criterios para automatizar dicha selección y son los siguientes:

- **Iluminación local.** Representar con más detalle aquellas zonas donde la iluminación cambia de forma brusca con lo que tanto brillos como límites de sombras se pueden representar mejor.
- **Volumen de la vista.** Representar con mayor detalle sólo las zonas de los objetos que quedan dentro del volumen de la vista. Con esto se pretende reducir el coste de las etapas de transformación y recortado del proceso de obtención de imágenes.
- **Silueta.** Aumentar el detalle en aquellas zonas que en la proyección van a formar parte de la silueta del objeto.
- **Orientación.** Aumentar el detalle en aquellas superficies que, estando situadas dentro del volumen de la vista, además están de cara al observador. El objetivo es reducir el coste de la etapa de eliminación de caras traseras en el proceso de obtención de imágenes.
- **Distancia al observador.** Dentro del volumen de la vista, es posible que hayan zonas más cercanas al observador que otras. Habría que representar con mayor detalle las zonas cercanas y disminuirlo a medida que la misma superficie se aleja del observador.

Las soluciones propuestas en la literatura pueden ser fácilmente incorporadas en nuestro esquema dado que el algoritmo es independiente de los criterios utilizados.

7.1. Funcionamiento básico

El algoritmo recorre la secuencia de simplificaciones $\{S_0,\ldots,S_{n-1}\}$ y decide si realizarlas o no utilizando para ello una funcion de test la cual incluye la evaluación de los criterios definidos por la aplicación.

Como resultado de la evaluación, si la respuesta es no llevar a cabo la simplificación S_i, cada triángulo $t' \in \mathcal{T}_i'$ y cada vértice $v' \in \mathcal{V}_i'$ no podrán pertenecer a la solución, generando una dependencia a la hora de realizar posteriores simplificaciones. Por ejemplo, en la figura 11, se representa una simplificación que consiste en la eliminación de un vértice *v* y la triangulación del agujero con los triángulos *tm, tn* y *tp*. Ninguno de estos triángulos, *tm, tn* y *tp*, podrán eliminarse en posteriores simplificaciones si el vértice *v* no se elimina previamente. Para controlar estas dependencias se define R como el conjunto de triángulos resultado de realizar la operación $R \leftarrow R \cup \mathcal{T}_i'$ para cada simplificación S_i no realizada (donde inicialmente $R = \emptyset$). Por lo tanto, independientemente del resultado de la evaluación del criterio, la simplificación S_i no será realizada si existe un triángulo t tal que $t \in \mathcal{T}_i$ y $t \in R$.

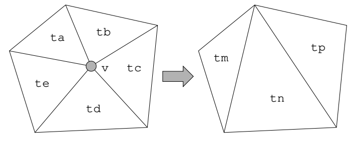


Figura 11: *Los triángulos* tm, tn y tp *no pueden aparecer en la solución si el vértice* v *no es eliminado previamente* ($\mathcal{V}_i = \{v\}, \mathcal{T}_i = \{ta,tb,tc,td,te\}, \mathcal{V}_i' = \{\}, \mathcal{T}_i' = \{tm,tn,tp\}$).

7.2. Algoritmo

El algoritmo de visualización de una aproximación con nivel de detalle variable se muestra en la figura 12. Para codificar el conjunto R se va a añadir una lista de marcas, el vector $flag$ en el algoritmo, que permita señalar aquellos triángulos que pertenecen a R. Inicialmente, todos los triángulos son susceptibles de ser visualizados, lo cual se codifica marcando cada triángulo como *VISUALIZABLE*. A partir de aquí, dos etapas son claramente visibles en el algoritmo propuesto.

```
// Inicializar la lista de marcas
flag[0..nTriangulos]= VISUALIZABLE;

// PRIMERA ETAPA
// para cada simplificación almacenada en la lista de control
para cada simplificación Si ∈ [S0..Sn−2] hacer
  si test(Si)= CIERTO entonces
    // marcar las caras que no se generan
    ∀t ∈ Ti' flag[t]= NO_VISUALIZABLE
    // para cada vertice a ser eliminado
    para cada vértice v ∈ Vi hacer
      // recorrer los triángulos asociados a v
      para cada triángulo t asociado a v hacer
        si flag[t]= VISUALIZABLE entonces pintaTriangulo(t);
        fin si
      fin para
    fin para
  fin si
fin para

// SEGUNDA ETAPA
// para la aprox. de menor detalle
para cada vértice v ∈ Mn−1 hacer
  // recorrer los triangulos asociados a v
  para cada triángulo t asociado a v hacer
    si flag[t]= VISUALIZABLE entonces pintaTriangulo(t);
    fin si
  fin para
fin para
```

Figura 12: *Algoritmo de visualización de una aproximación con nivel de detalle variable.*

Primera etapa

Esta primera parte recorre la secuencia de $n-1$ simplificaciones que, partiendo de M_0, generan las $n-1$ restantes aproximaciones almacenadas en la representación multirresolución. Para decidir si realizar una simplificación S_i, se define la función de $test()$ que será $Cierto$ (preserva el detalle) si ocurre alguna de las dos siguientes situaciones y $Falso$ en cualquier otro caso:

1. El criterio de la aplicación decide preservar detalle y existe un triángulo t tal que $t \in \mathcal{T}_i$ y $t \in T_0$.
2. Si existe un triángulo $t \in \mathcal{T}_i'$ tal que flag[t]= *NO_VISUALIZABLE*.

Dependiendo del resultado de la función de *test()*, la acción a tomar será una de entre las dos siguientes:

1. Si $test(S_i) = Cierto$, $\forall t \in \mathcal{T}_i'$ flag[t]= *NO_VISUALIZABLE* y cada triángulo $t \in \mathcal{T}_i$ se visualiza si flag[t]= *VISUALIZABLE*.
2. Si $test(S_i) = Falso$, el algoritmo efectúa la simplificación, lo que significa que no se realiza ningún tipo de acción y se pasa a examinar la información referente a la siguiente simplificación.

Segunda etapa

Esta segunda parte concierne a los triángulos que pertenecen a la aproximación de menor detalle, M_{n-1}. Simplemente se recorre la parte de la estructura de datos correspondiente a M_{n-1} visualizando aquellos triángulos que estén marcados como *VISUALIZABLE*. Estos triángulos terminarán de completar la aproximación requerida.

Ejemplo

En la figura 10 se muestra el objeto *Cow* en el que se pueden distinguir tres conjuntos de triángulos definidos por colores diferentes. Cada color representa la etapa y el motivo por el que han sido dibujados:

- Gris claro: triángulos dibujados en la primera etapa porque el criterio de la aplicación decide preservar con detalle esa zona del objeto.
- Gris medio: triángulos dibujados en la primera etapa correspondientes a una zona del objeto que el criterio de la aplicación decide no preservar con detalle, pero que entre los triángulos asociados a los vértices de esta zona existen triángulos marcados como *NO_VISUALIZABLE*.
- Gris oscuro: triángulos dibujados en la segunda etapa del algoritmo.

7.3. Coste temporal

El análisis del coste temporal muestra que en el peor caso el algoritmo es $O(nK + T^r + V^r)$, donde n es el número de simplificaciones, K es el coste de evaluación del criterio definido por la aplicación, y T^r, V^r son el número total de triángulos y vértices de la representación multirresolución respectivamente. Si K es constante, el primer término es de menor orden y puede ser eliminado. T^r y V^r son, en el peor caso, de orden cuadrático respecto al tamaño del modelo original (T y V respectivamente). Sin embargo, teniendo en cuenta que cada vértice de la superficie está compartido por seis triángulos adyacentes por término medio, y utilizando un método de simplificación basado en contracción de aristas o eliminación de vértices en el proceso de construcción de la representación multirresolución, T^r y V^r son de orden lineal respecto a T y V respectivamente.

7.4. Mejora de las prestaciones

El algoritmo propuesto en la sección 7.2, y que de aquí en adelante llamaremos algoritmo de *Fuerza bruta*, puede presentar un comportamiento poco eficiente cuando la zona a mostrar con alto nivel de detalle es pequeña comparada con la superficie total del objeto. Esto es debido a dos causas principalmente:

1. Evaluación del test para cada simplificación almacenada en M^r. La evaluación puede ser una operación cara. Resulta absurdo repetirla para toda la secuencia de simplificaciones cuando ésta sólo se evaluará a cierto en un número reducido de ocasiones.
2. Recorrido de toda la secuencia de simplificaciones. Esta tarea es realmente costosa y, sin embargo, la mayor parte de las simplificaciones se realizan.

Así, para aumentar las prestaciones del algoritmo de fuerza bruta, se proponen las dos siguientes actuaciones:

- Aplicar un método de agrupamiento sobre el modelo creando una jerarquía de vértices de dos niveles.
- Mantener una lista dinámica de simplificaciones. Esta lista la formarán aquellas simplificaciones, S_i, que no se realizen, es decir, que cumplan que al menos exista un triángulo $t \in \mathcal{T}_i$ tal que t pertenezca a la aproximación requerida.

A continuación se detallan ambas actuaciones.

7.5. Agrupamiento

Se propone aplicar un método de agrupamiento para generar una jerarquía de sólo dos niveles de profundidad. Cada agrupación reúne las simplificaciones que actúan sobre la misma zona de la superficie del modelo (figura 13). El propio método de simplificación utilizado en la construcción del modelo multirresolución puede servir para generar las agrupaciones. Cada agrupación contendrá, además, la información necesaria de forma que el test de refinamiento pueda aplicarse directamente sobre él.

El test se aplica ahora a todas y cada una de las agrupaciones pero las simplificaciones reunidas en una agrupación serán chequeadas sólo cuando el resultado del test indique que haya que hacerlo. Si, como es previsible, esto ocurre en contadas ocasiones entonces el número de veces que se aplicará el test se reducirá en un alto porcentaje.

Para implementar esta mejora es necesario crear una nueva lista que almacene la secuencia de clusters. Cada cluster almacenará una lista de simplificaciones así como la información necesaria para poder aplicar el test de refinamiento.

7.6. Lista dinámica

El objetivo es reducir el número de simplificaciones a recorrer manteniendo una lista dinámica formada por sólo aquellas simplificaciones S_i que cumplan que al menos exista un triángulo $t \in \mathcal{T}_i$ tal que t pertenezca a la aproximación requerida. Esto ocurre o bien cuando el *test* indica que una simplificación no debe llevarse a término o bien cuando haya una dependencia con otras simplificaciones. Para este último caso, hay que amplíar la estructura de datos de forma que cada triángulo almacenado en M^r conozca el vértice al que está asociado. Esta ampliación se realiza en tiempo de ejecución una vez la representación MOM resida en memoria por lo que no es necesario modificar el proceso de construcción explicado en la sección 5. Sin embargo, esta ampliación supone un incremento de aproximadamente el 24% del coste de almacenamiento de la representación multirresolución pero, ya que se realiza una vez cargado el objeto MOM en memoria, la aplicación puede decidir entre llevarla acabo o no según la disponibilidad de los recursos del sistema.

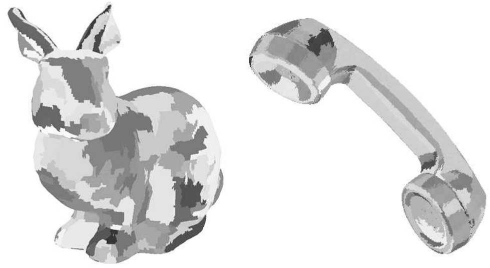


Figura 13: *Agrupaciones de los modelos* Bunny *(izquierda) y* Phone *(derecha). El modelo* Bunny *consta de 264 agrupaciones mientras que el modelo* Phone *consta de 408. Cada agrupación ha sido dibujada de un color diferente.*

La construcción de la lista dinámica es la siguiente. Inicialmente la lista está vacía. Cada simplificación que el test de refinamiento indique que no haya que llevar a cabo (es decir, que haya que preservar el detalle), se añade a la lista. Cada simplificación añadida a la lista implica una serie de dependencias, es decir, otras simplificaciones que tampoco pueden llevarse a cabo y, por lo tanto, éstas también han de añadirse a la lista. Al final de este proceso se obtiene una lista formada por las simplificaciones que no pueden realizarse y que son las que contienen los triángulos que forman la solución.

La lista dinámica generada puede ser creada para cada fotograma, o actualizarse aplicando coherencia. En este caso, a la hora de actualizarse, ésta se recorre en primer lugar para eliminar aquellas simplificaciones que ya no formen parte del siguiente fotograma. Después se añadirán las nuevas simplificaciones de la misma forma que se ha explicado en el párrafo anterior.

7.7. Resultados

Los experimentos fueron realizados utilizando una estación de trabajo Silicon Graphics RealityEngine 2, con un MIPS

R10000 a 194 MHz y 256 Mb RAM. La codificación de MOM se realizó en *C++* y utilizando la librería gráfica *OpenGL*. El método de simplifcación utilizado para construir las representaciones multirresolución es el propuesto por Garland y Heckbert [20] basado en contracción de aristas obligando a que el nuevo punto coincida con uno de los vértices de la arista.

Son tres los algoritmos que finalmente se han implementado. El primero es el algoritmo de *Fuerza bruta* tal y como está descrito en la sección 7.2. El segundo es el resultado de añadir las agrupaciones y la lista dinámica, y que se denomina algoritmo de *Agrupamiento*. El tercero es el resultado de explotar la coherencia cuadro a cuadro en el algoritmo de *Agrupamiento* y que se denomina algoritmo de *Coherencia*.

El objetivo de los experimentos es observar y comparar los tiempos de visualización de los distintos algoritmos cuando la zona a preservar con detalle varía en tamaño. Dos son los modelos utilizados: *Bunny* y *Phone*. Para cada uno de ellos se ha seleccionado una zona de la superficie a refinar. Esta zona se determina utilizando una esfera cuyo centro se designa interactivamente y su radio varía haciendo que la zona del modelo que queda dentro quede refinada.

En las figuras 14 y 15 se representa como varían las prestaciones de los tres algoritmos al ir aumentando la zona de refinamiento. El eje X representa el número de triángulos que forman la aproximación visualizada. El eje Y representa el número de fotogramas por segundo que se consiguen. También se ha incluido una curva, denominada *Uniforme*, con el número de fotogramas que se obtienen cuando se visualiza el modelo original. Las tablas 6 (*Bunny*)y 7 (*Phone*) recogen las prestaciones obtenidas por los tres algoritmos para tres aproximaciones concretas. En la figura 18 se muestran imágenes de dichas aproximaciones.

Los tres algoritmos mejoran las prestaciones cuando la zona a refinar es pequeña comparada con el tamaño global del modelo. Cuanto menor sea dicha zona mejores son las prestaciones. En concreto, los algoritmos de Fuerza bruta y Agrupamiento presentan mejora si las aproximaciones visualizadas están formadas por menos de aproximadamente el 15% del número de triángulos del modelo, mientras que el de Coherencia amplía este margen hasta aproximadamente el 28%. El algoritmo de Fuerza bruta es el que menos mejora produce aunque tiene la ventaja de que no necesita ampliar la estructura de datos. Los otros dos algoritmos propuestos (Agrupamiento y Coherencia) consiguen una mejora muy notable cuando refinan zonas muy pequeñas y caen rápidamente en cuanto dicha zona se amplía.

A continuación se va a comparar el modelo MOM con MT. No se ha incluido una comparación con HDS debido a que los autores no describen los experimentos realizados resultando imposible su reproducción.

Table 6: *Fotogramas por segundo obtenidos para tres aproximaciones diferentes del objeto* Bunny.

#Triángulos	2.107	4.625	14.216
Fuerza bruta	16’4	15’4	12’1
Agrupamiento	143	40’0	10’3
Coherencia	240	66’6	18’9

Table 7: *Fotogramas por segundo obtenidos para tres aproximaciones diferentes del objeto* Phone.

#Triángulos	4.794	17.464	62.132
Fuerza bruta	6’6	5’5	3’7
Agrupamiento	32’3	6’6	1’9
Coherencia	50’0	10’6	3’5

7.8. Comparación con *Multi-Triangulation*

Para comparar el comportamiento de MOM con el modelo multirresolución Multi-Triangulation (MT) se ha intentado repetir el experimento realizado por los autores de la implementación de MT. Este experimento consiste en desplazar a lo largo de una trayectoria un cubo paralelo a los ejes de coordenadas de forma que se refine la zona del objeto interior a dicho cubo. La falta de detalles de la realización del experimento complican su reproducción. Aun así, se ha intentado realizar cuidando que las mallas extraídas consten de aproximadamente el mismo número de triángulos.

En la figura 16 se muestra el número de triángulos obtenidos en cada fotograma para MOM y MT utilizando el modelo *Bunny*. El eje X representa el fotograma, y el eje Y el número de triángulos visualizado. En la figura 17 se muestran los tiempos de extracción de los tres algoritmos propuestos para MOM (fuerza bruta, agrupamiento y coherencia) y el que muestra un mejor comportamiento de los algoritmos propuestos para MT. El eje X representa el fotograma, y el eje Y el tiempo en milisegundos.

Estos resultados muestran que sólo el algoritmo de fuerza bruta tiene un comportamiento peor que el de MT. Tanto el de agrupamiento como el de coherencia mejoran claramente las prestaciones obtenidas por el algoritmo de MT.

8. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo, el esfuerzo realizado se ha dirigido hacia resolver el problema de la visualización interactiva de modelos de superficie poligonal. El objetivo ha consistido en desarrollar un esquema multirresolución que, sin imponer restricciones acerca de la geometría del objeto y de su simplificación, sea capaz de realizar una manipulación eficiente del nivel de detalle, tanto uniforme como variable, a fin de mejorar las prestaciones de los modelos existentes.

El esquema MOM permite, ante una petición, recuperar

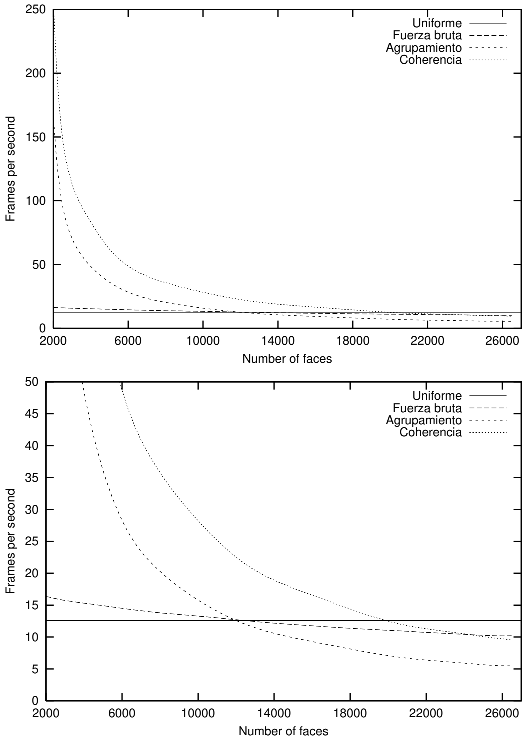


Figura 14: *Comportamiento de los algoritmos de visualización para el objeto* Bunny. *Arriba se muestra el resultado, abajo se muestra una ampliación.*

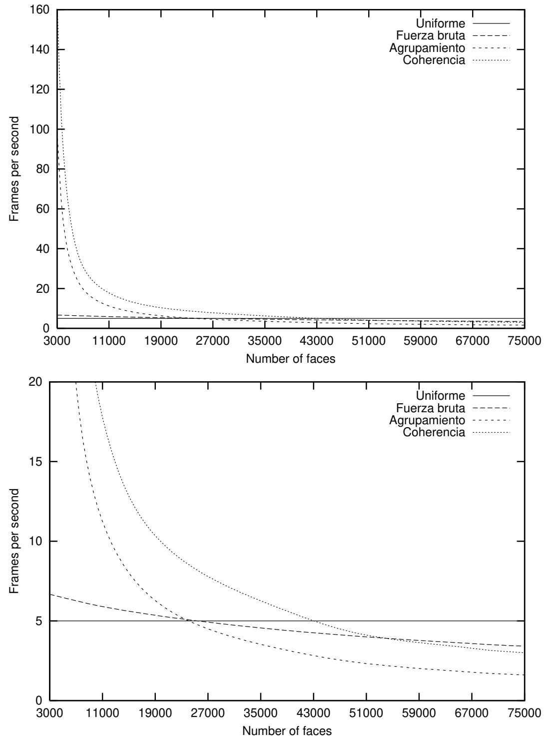


Figura 15: *Comportamiento de los algoritmos de visualización para el objeto* Phone. *Arriba se muestra el resultado, abajo se muestra una ampliación.*

aproximaciones eficientemente manteniendo un bajo coste de almacenamiento. Sus principales ventajas son: la simplicidad, la estructura de datos básica se basa en listas; la generalidad, puede utilizarse cualquier método de simplificación y cualquier tipo de superficie poligonal; y la eficiencia, tanto en el coste de almacenamiento como en el coste temporal.

La estructura de datos permite un coste de almacenamiento menor que el de los esquemas multirresolución de características similares (MT y HDS) y es comparable a modelos que carecen de la característica de generalidad (PM).

El algoritmo de extracción de aproximaciones con nivel de detalle uniforme es más eficiente que el de los modelos multirresolución de características similares. La comparación realizada con el modelo PM muestra la influencia que produce en las prestaciones de los modelos la diferencia de nivel de detalle entre dos aproximaciones requeridas consecutivamente. Esta influencia es nula para MOM pero muy importante en PM que sólo presenta apenas mejores prestaciones cuando la diferencia en el nivel de detalle de las dos aproximaciones es muy pequeña.

El algoritmo que permite extraer aproximaciones con nivel de detalle variable no necesita reorganizar los datos en una nueva estructura ni almacenar explícitamente ningún ti-

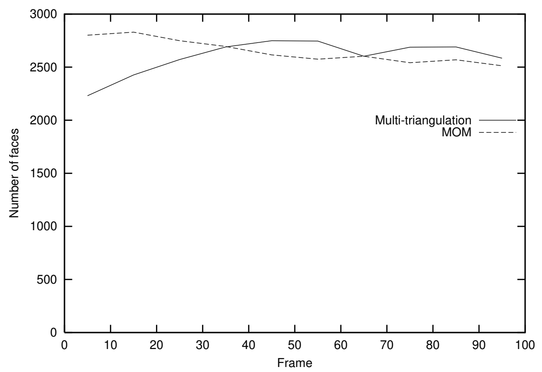


Figura 16: *Triángulos extraídos para el modelo* Bunny.

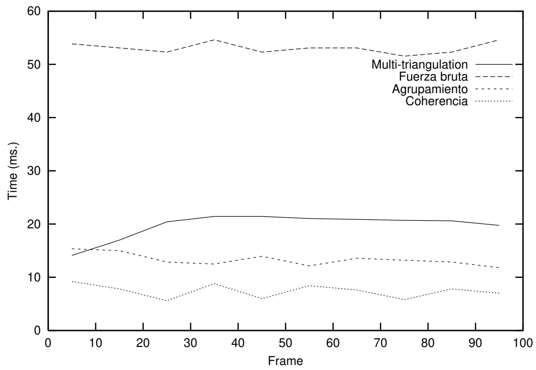


Figura 17: *Tiempos de extracción (ms.) para el modelo* Bunny.

po de dependencia entre las distintas actualizaciones del objeto representado. Junto a este algoritmo se han presentado una serie de mejoras que permiten aumentar la eficiencia y obtener mejores prestaciones que las conseguidas por el modelo multirresolución MT.

Son varias las posibilidades de extender este trabajo. Todas ellas se centran en aumentar el rendimiento del modelo disminuyendo los tiempos de extracción y de visualización sin que ello suponga un aumento desmedido del coste de almacenamiento.

Acceso a memoria. Hoppe [30], resolviendo el problema de la búsqueda de tiras en una malla, demuestra que puede ser más importante aumentar la tasa de acierto de la memoria caché que generar tiras largas. El-Sana y Varshney [10] tienen en cuenta el problema del acceso localizado a memoria para evitar fallos de página y aumentar así las prestaciones de su modelo. Algunas pruebas realizadas muestran que la tasa de fallo de caché al visualizar el nivel de mayor detalle del objeto *Buda* es del 17%. Es decir, el 17% de las veces que se ha accedido a memoria ha habido que llegar hasta la memoria principal para obtener el dato (suponiendo una memoria caché de 1024Kb). Esta tasa es claramente mayor que la del 2'5% de fallo que se obtiene con un modelo basado en lista de vértices y lista de triángulos. Por otra parte, el algoritmo de fuerza bruta para la extracción de aproximaciones de nivel de detalle variable presenta una tasa de fallo de caché aún mayor, sobre el 25%. Estos primeros resultados muestran que el uso de la caché no es tan alto en el modelo multirresolución como en el modelo de representación única. Si se organizaran los datos teniendo en cuenta su localidad, podría aumentarse la tasa de acierto a la hora de buscar un dato en la memoria caché, lo que supondría una disminución de los tiempos de extracción.

Paralelismo. Cada vez son más accesibles las estaciones de trabajo con varias unidades de proceso. Estas estaciones suelen contar con un único sistema gráfico y, generalmente, sólo uno de los procesadores puede alimentarlo. Esto hace que paralelizar el modelo no sea una tarea obvia. Sin embargo, una primera aproximación consistiría en que un procesador sea el que determine los triángulos que pertenecen a la aproximación requerida y que un segundo procesador sea el que mande a pintar cada uno de los triángulos. En cualquier caso, estudiar una posible paralelización de un modelo multirresolución es una futura línea de investigación muy interesante que podría aumentar de forma importante la tasa de cuadros por segundo y sobre la que sólo El-Sana et al. [11] han presentado un pequeño apunte.

Primitivas gráficas. Las librerías gráficas poseen primitivas de dibujo que permiten acelerar el proceso de visualización como, por ejemplo, la tira de triángulos. Hacer uso de estas extensiones es particularmente difícil en modelos multirresolución debido al continuo cambio de la conectividad de la malla. El-Sana et al. [11] han dado una primera solución a este problema permitiendo utilizar tiras de triángulos, actualizando las tiras a cada cuadro al mismo tiempo que cambia la aproximación requerida.

Agradecimientos

Este trabajo ha sido financiado por los proyectos TIC1999-0510-C02-02 y TIC2000-1131 (CICYT, Ministerio de Educación y Ciencia). Nuestro agradecimiento a Michael Garland por hacer disponible su código, y a Cyberware, Inc. y al *Standford Computer Graphics Laboratory* por proveer los conjuntos de datos tridimensionales.

Referencias

1. V. Abdajev, M. del Rosario, A. Lebedev, A. Migdal and V. Paskhaver, Metastream, *Proc. of VRML'99* (1999).
2. G. Bell, A. Parisi and M. Pesce, The Virtual Reality Modeling Language: Version 1.0 Specification, (http://www.vrml.org/VRML1.0/vrml10c.html, 1995).
3. A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin and W. Stuetzle, Interactive Multiresolution Surface Viewing, *Proc. of SIGGRAPH '96* (1996) 91–98.
4. A. Ciampalini, P. Cignoni, C. Montani and R. Scopigno, Multiresolution Decimation Based on Global Error, *The Visual Computer* **13(5)** (1997) 228–246.
5. P. Cignoni, E. Puppo, and R. Scopigno, Representation and visualization of terrain surfaces at variable resolution, *The Visual Computer* **13** (1997) 199–217.
6. P. Cignoni, C. Montani, and R. Scopigno, A comparison of mesh simplification algorithms, *Computers & Graphics* **22(1)** (1998) 37–54.
7. P. Cignoni, C. Rocchini, and R. Scopigno, Metro: measuring error on simplified surfaces, *Computer Graphics Forum* **17(2)** (1998) 167–174.
8. M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, C. Aldrich and Mineev-Weinstein, ROAMing terrain: real-time optimally adapting meshes, *Proc. of Visualization '97* (1997) 81–88.
9. M. Eck , T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle, Multiresolution Analysis of Arbitrary Meshes, *Proc. of SIGGRAPH '95* (1995) 173–182.
10. J. El-Sana and A. Varshney, Generalized View-Dependent Simplification, *Proc. of EUROGRAPHICS '99* (1999) 83–94.

11. J. El-Sana, E. Azanli and A. Varshney, Skip Strips: Maintaining triangle strips for view-dependent rendering, *Proc. of Visualization '99* (1999) 131–137.

12. C. Erikson, Polygonal Simplification: An Overview, *Technical Report TR96-016* (Department of Computer Science, UNC-Chapel Hill, 1996).

13. L. De Floriani, L. Marzanoo and E. Puppo, Multiresolution models for topographic surface description, *The Visual Computer* **12** (1996) 317–345.

14. L. De Floriani, E. Puppo and P. Magillo, A Formal Approach to Multiresolution Hypersurface Modeling, *Geometric Modeling: Theory and Practice* (W. Straber, R. Kein, R. Rau (Editors), Springer-Verlag, 1997).

15. L. De Floriani, P. Magillo and E. Puppo, Building and Traversing a Surface at Variable Resolution, *Proceedings IEEE Visualization '97* (1997) 103–110.

16. L. De Floriani, P. Magillo and E. Puppo, Efficient Implementation of Multi-Triangulations, *Proceedings IEEE Visualization '98* (1998) 43–50.

17. J.D. Foley, A. van Dam, S. Feiner and J. Hughes, Computer Graphics. Principles and Practice, *Addison-Wesley Publishing Company* (1990).

18. R.J. Fowler and J.J. Little, Automatic extraction of irregular network digital terrain models, *Proc. of SIGGRAPH '79* **13(3)**, (1979) 199–207.

19. T. Funkhouser, C. Séquin and S. Teller, Management of large amounts of data in interactive building walkthroughs, *Symposium on Interactive 3D Graphics* (1992) 11–20.

20. M. Garland and P. Heckbert, Surface Simplification Using Quadric Error Metrics, *Proc. of SIGGRAPH '97* (1997) 209–216.

21. M. Garland, Multiresolution Modeling: Survey & Future Opportunities, *State of the Art Reports of EUROGRAPHICS '99* (1999) 111–131.

22. A. Guéziec, G. Taubin, F. Lazarus and W. Horn, Simplicial maps for progressive transmission of polygonal surfaces, *Proc. of VRML'98* (1998).

23. A. Guéziec, G. Taubin, B. Horn and F. Lazarus, A Framework for Streaming Geometry in VRML, *IEEE Computer Graphics and Applications* **19(2)** (1999) 68–78.

24. P.S. Heckbert and M. Garland, Multiresolution modeling for fast rendering, *Proc. of Graphics Interface '94* (1994) 43–50.

25. P.S. Heckbert and M. Garland, Survey of polygonal surface simplification algorithms, *Multiresolution Surface Modeling Course Notes of SIGGRAPH'97* (1997).

26. H. Hoppe, Progressive Meshes, *Proc. of SIGGRAPH '96* (1996) 99–108.

27. H. Hoppe, View-Dependent Refinement of Progresive Meshes, *Proc. of SIGGRAPH '97* (1997) 189–198.

28. H. Hoppe, Smooth view-dependent level-of-detail control and its application to terrain rendering, *Proc. of Visualization '98* (1998) 35–42.

29. H. Hoppe, Efficient Implementation of Progressive Meshes, *Computer & Graphics* **22(1)** (1998) 27–36.

30. H. Hoppe, Optimization of mesh locality for transparent vertex caching, *Proc. of SIGGRAPH '99* (1999) 369–276.

31. R. Klein and W. Strasser, Generation of multiresolution models from CAD data for real time rendering, *Theory and Practice of Geometric Modeling* (Springer-Verlag, 1997).

32. J. Lee, Analisys of visibility sites on topographic surfaces, *Geographic Information Systems* **5(4)** (1991) 413–425.

33. P. Lindstrom, D. Koller, W. Ribarsky, L.F. Hodges, N. Faust and G.A. Turner, Real-time, continuous level of detail rendering of height fields, *Proc. of SIGGRAPH '96* (1996) 109–118.

34. P. Lindstrom and G. Turk, Fast and Memory Efficient Polygonal Simplification, *Proc. of IEEE Visualization 98* (1998) 279–286.

35. P. Lindstrom and G. Turk, Evaluation of Memoryless Simplification, *IEEE Transactions on Visualization and Computer Graphics* **5(2)** (1999).

36. M. Lounsbery, T. DeRose and J. Warren, Multiresolution analysis for surfaces of arbitrary topological type, *ACM Transactions on Graphics* **16(1)** (1997) 34–73.

37. D. Luebke and C. Erikson, View-Dependent Simplification of Arbitrary Polygonal Environments, *Proc. of SIGGRAPH '97* (1997) 199–208.

38. E. Puppo, Variable resolution terrain surfaces, *Proc. of Eight Canadian Conference on Computational Geometry* (1996) 202–210.

39. E. Puppo and R. Scopigno, Simplification, LOD and Multiresolution - Principles and Applications, *Tutorial Notes of EUROGRAPHICS'99* **16(3)** (1997).

40. E, Puppo, Variable resolution triangulations, *Computational Geometry* **11(3-4)** (1998) 219–238.

41. J. Popovic and H. Hoppe, Progressive simplicial complexes, *Proc. of SIGGRAPH'97* (1997) 217–224.

42. C. Prince, Progressive meshes for large models of arbitrary topology, *Masters Thesis* (Department of Computer Science and Engineering, University of Washington, Seattle, 1997).

43. J. Ribelles, M. Chover, J. Huerta and R. Quirós, Multiresolution Ordered Meshes, *Proc. of 1998 IEEE Conference on Information Visualization* (1998) 198–204.

44. J. Ribelles, M. Chover, A. López and J. Huerta, A First Step to Evaluate and Compare Multiresolution Models, *Short Papers and Demos of EUROGRAPHICS '99* (1999) 230–232.

45. J. Ribelles, A. López, O. Belmonte, I. Remolar and M. Chover, Variable Resolution Level-of-detail of Multiresolution Ordered Meshes, *Proc. of WSCG'01* **(2)** (2001) 299–306.

46. J. Rohlf and J. Helman, IRIS Performer: A high performance multiprocessing toolkit for real-time 3D graphics, *Proc. of SIGGRAPH '94* (1994) 381–394.

47. L.L. Scarlatos, A refined triangulation hierarchy for multiples level of terrain detail, *Proc. of Image V Conference* (1990) 115–122.

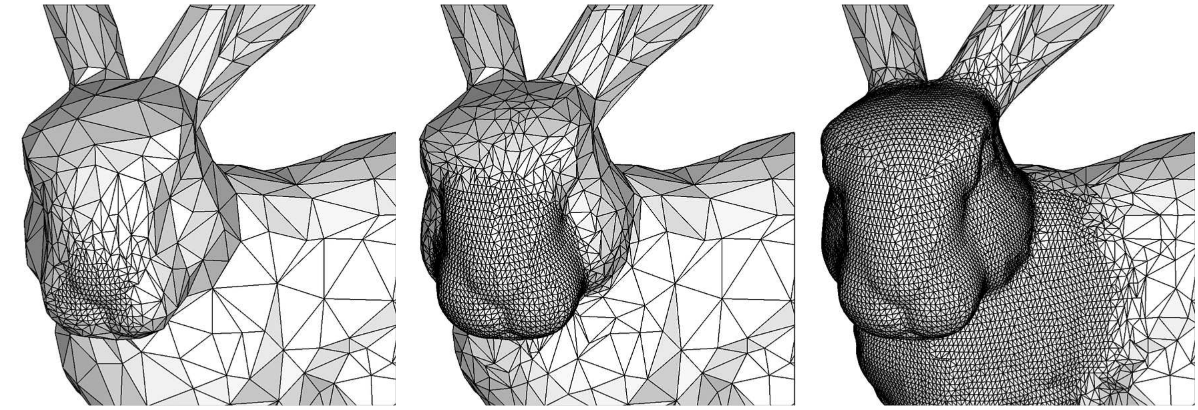
48. G. Taubin, A. Guéziec, W. Horn and F. Lazarus, Progressive forest split compression, *Proc. of SIGGRAPH '98* (1998) 123–132.

49. D.D. Taylor and W.A. Barret, An algorithm for continuous resolution polygonalizations of a discrete surface, *Proc. of Graphics Interface '94* (1994) 33–42.

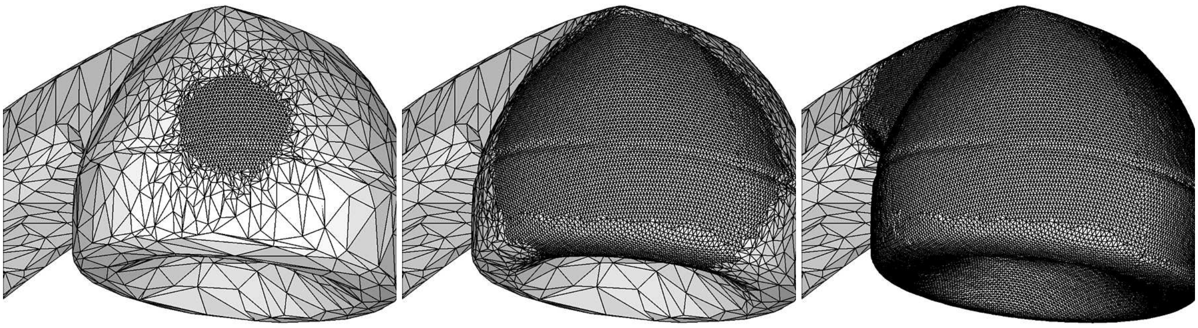
50. J. Wernecke, The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2, *Addison-Wesley* (1993).

51. J. Xia and A. Varshney, Dynamic view-dependent simplification for polygonal models, *Proc. of IEEE Visualization '96* (1996) 327–334.

52. J. Xia, J. El-Sana and A. Varshney, Adaptive Real-Time Level-of-Detail-Based Rendering for Polygonal Models, *IEEE Transactions on Visualizations and Computer Graphics* **3(2)** (1997) 171–183.



(a) 2.107 triángulos (b) 4.625 triángulos (c) 14.216 triángulos



(d) 4.794 triángulos (e) 17.464 triángulos (f) 62.132 triángulos

Figura 18: *Aproximaciones de nivel de detalle variable obtenidas de los modelos* Bunny *y* Phone*.*