UNIVERSITAT JAUME I

Escuela Superior de Tecnología y Ciencias Experimentales

Departamento de Informática

MALLAS ORDENADAS MULTIRRESOLUCIÓN

TESIS DOCTORAL

presentada por

José RIBELLES MIGUEL

dirigida por

Miguel CHOVER SELLÉS

Castellón, Noviembre de 2000

A Mª Angeles

Resumen

El modelado de superficies mediante mallas de polígonos domina actualmente en el campo de los gráficos interactivos. Típicamente, cada objeto se modela mediante una única aproximación poligonal. 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.

Esta tesis presenta un nuevo modelo multirresolución denominado *Mallas Ordenadas Multirresolución (MOM)*. Este modelo realiza una manipulación eficiente del nivel de detalle pudiendo extraer aproximaciones con nivel de detalle tanto uniforme como variable, haciéndolo idóneo para la visualización interactiva de modelos poligonales complejos. 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. Además, se presentan una serie de extensiones que permiten explotar la propiedad de la coherencia cuadro a cuadro y el uso de abanicos de triángulos con el fin de aumentar aún más las prestaciones y mejorar el comportamiento general del modelo.

Palabras clave: modelado multirresolución, nivel de detalle, simplificación poligonal, visualización interactiva.

Agradecimientos

Comenzando por el principio, a mis padres, por su apoyo y porque donde hoy estoy es en gran parte gracias a ellos. A mi familia, por su cariño y su enorme paciencia. A mis amigos, por las tertulias, por los paseos, por sus consejos, por sus ánimos, por ser quienes son, en definitiva, por su amistad.

A Miguel Chover, no sólo por ser mi director de tesis sino también un amigo, por su paciencia y su continuo ánimo, por sus valiosos consejos, y por guiarme en el arduo aprendizaje de la investigación.

A Oscar e Inma por su amistad y ánimo, por el trabajo en grupo y por hacer más fácil el día a día. A todos mis compañeros del Grupo de Informática Gráfica y a los miembros de la sección de Informática Gráfica de la Universidad Politécnica de Valencia por crear un magnífico entorno de trabajo.

Y finalmente, a Mª Angeles, a quien dedico este trabajo, por absolutamente todo.

Índice General

Índice de Figuras

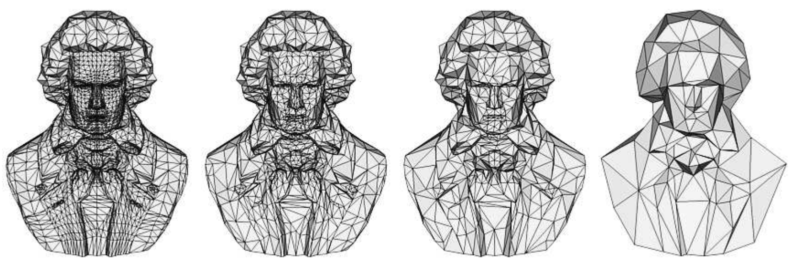
Índice de Tablas

Capítulo 1

Introducción

Los modelos geométricos con un alto grado de detalle están disponibles, cada vez con más frecuencia, para ser utilizados por una gran diversidad de aplicaciones. Estos modelos se representan, normalmente, mediante complejas mallas poligonales formadas por cientos de miles, o incluso millones, de polígonos. La visualización interactiva de tal cantidad de datos excede ampliamente la capacidad de los sistemas gráficos actuales. Este problema puede resolverse si se dispone de un conjunto de aproximaciones del modelo, diferentes en complejidad, de forma que en base a un criterio se seleccione la más apropiada.

La presente tesis se centra en el modelado de superficies mediante mallas de polígonos con múltiples aproximaciones (figura 1.1), conocido en la bibliografía como *Modelado Multirresolución de Superficies Poligonales*. Se describe un nuevo modelo multirresolución, detallando la estructura de datos que define cómo almacenar las distintas aproximaciones, los algoritmos de acceso que permiten extraerlas de forma eficiente, y se realiza un estudio del comportamiento del nuevo modelo tanto teórica como experimentalmente.



(a) Original **(b)** 50% **(c)** 25% **(d)** 5%

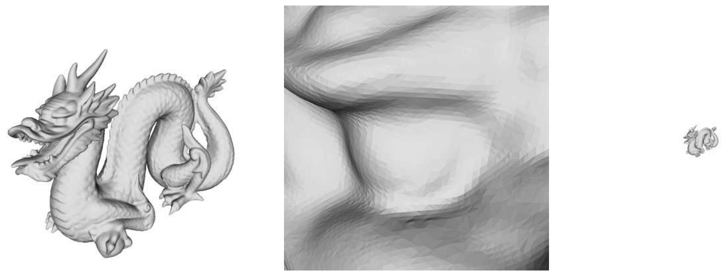
Figura 1.1: Cuatro aproximaciones distintas del modelo *Ludwig*. El porcentaje representa el número de caras utilizadas en comparación con el original.

1.1 Motivación

Los avances tecnológicos han proporcionado extensas bases de datos de objetos modelados con superficies poligonales. A su vez, estos modelos son muy complejos y no es raro que estén formados por millones de polígonos. Por ejemplo, la adquisición de datos vía satélite, los sistemas de escáner por láser o los dispositivos de imagen médica, son métodos que, gracias a su alta precisión, son capaces de generar modelos de gran complejidad. Campos como la cartografía, la ingeniería, la informática gráfica o la visión por ordenador hacen uso de este tipo de modelos en aplicaciones como la representación de terreno, CAD, visualización científica, simuladores de vuelo, análisis de elementos finitos, realidad virtual, etc. Por otra parte, estas aplicaciones necesitan, cada vez más, de interactividad con el usuario y, en definitiva, visualizar tal enorme cantidad de datos se convierte en un problema que llega a ser crítico si además es necesario mantener una alta tasa de cuadros por segundo, como por ejemplo ocurre en realidad virtual donde se requieren alrededor de treinta cuadros por segundo.

En todas estas aplicaciones hay que establecer un equilibrio entre el nivel de detalle con el que se modela un objeto y la cantidad de tiempo necesario para procesarlo. Una solución frecuentemente utilizada para conseguir unos tiempos de ejecución aceptables consiste en sustituir el modelo original por una aproximación más simple. Sin embargo, determinar el nivel de detalle que debe tener esta aproximación no es algo que se pueda fijar de antemano. Durante la ejecución de la aplicación, las condiciones en las que se encuentra el modelo a visualizar no tienen porqué permanecer constantes, pueden variar a cada instante y, en consecuencia, la selección del nivel de detalle de la aproximación debe adaptarse a las posibles variaciones de dichas condiciones. Este problema se encuentra tanto en sistemas de bajo nivel con aplicaciones como juegos o entornos virtuales debido, principalmente, a las restricciones de los recursos del sistema, como también en sistemas de alto nivel donde las aplicaciones de simulación o visualización científica manejan cantidades de datos mucho mayores de lo que las arquitecturas gráficas actuales son capaces de visualizar de forma interactiva con el usuario.

El modelado multirresolución representa a un objeto mediante múltiples aproximaciones y permite que la aplicación realice un control eficiente del nivel de detalle, adaptándolo en tiempo de ejecución. Esta adaptación suele basarse en diversos aspectos como son la distancia al observador, importancia en la escena, prestaciones del sistema gráfico, etc. Por ejemplo, en la figura 1.2(a) se muestra un objeto formado por 869.928 caras. En la figura 1.2(b) se está examinando una pequeña parte de él y el resto del modelo queda fuera del campo de visión pudiendo ser ignorado. En la figura 1.2(c) el objeto se encuentra tan lejos del observador que con una aproximación de bajo detalle es suficiente para mantener su apariencia. En consecuencia, un modelo multirresolución debe definir las estructuras de datos y los algoritmos de acceso que permitan almacenar y extraer de forma eficiente las diferentes aproximaciones, teniendo en cuenta que su diseño va a repercutir de forma directa en las prestaciones y determinará su aplicabilidad. De nada sirve un modelo en el que la sobrecarga que produce el proceso de manipulación de los datos sea tan alta que haga inútil los esfuerzos por acelerar la visualización.



(a) Original **(b)** Detalle **(c)** Lejano

Figura 1.2: Tres vistas diferentes del modelo *Dragon*.

1.2 Contexto

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 primeros modelos estaban basados en mantener un número discreto de aproximaciones y fueron desarrollados con el objetivo principal de acelerar la visualización. Más tarde, aparecen los modelos multirresolución que almacenan un rango continuo de aproximaciones y permiten que el nivel de detalle se adapte con mayor exactitud a los requisitos de la aplicación. A los primeros se les ha denominado *modelos multirresolución discretos*, y a los segundos *modelos multirresolución continuos*.

Independientemente del tipo de modelo multirresolución (discreto o continuo), 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. En los últimos años, se han presentado numerosas técnicas de simplificación de forma paralela al desarrollo de los modelos multirresolución y, en muchos casos, estos últimos obligan a que la simplificación se realice de una forma determinada estableciendo una estrecha relación entre el modelo multirresolución y la técnica de simplificación utilizada para construirlo.

Los modelos multirresolución continuos no sólo se han propuesto como solución al problema de la visualización interactiva de modelos poligonales complejos, sino también a otros problemas que conlleva el uso de objetos formados por cientos de miles de polígonos, como son los elevados costes de almacenamiento y de transmisión. Los primeros modelos aparecidos trataban de dar una solución conjunta a los tres problemas (visualización, almacenamiento y transmisión). Sin embargo, en poco tiempo los modelos que se desarrollaban se especializaban en resolver uno de los problemas en concreto.

En la presente tesis doctoral, el esfuerzo se ha dirigido hacia resolver el problema de la visualización. En la literatura, los modelos multirresolución continuos aportan dos posibles soluciones: proporcionar aproximaciones donde el nivel de detalle sea *uniforme*, o proporcionar aproximaciones donde el nivel de detalle sea *variable*. 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 según la visibilidad y la distancia respecto al piloto.

Como se verá en el siguiente capítulo, son diversos los modelos multirresolución que han sido propuestos y que ofrecen una de las dos soluciones o ambas a la vez. Sin embargo, son pocos los que se han desarrollado sin imponer condiciones acerca de la geometría del objeto o sobre la técnica de simplificación válida para construir la representación multirresolución. En esta tesis, el objetivo ha consistido en desarrollar un nuevo modelo multirresolución continuo 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.

1.3 Contribuciones

Se presenta un nuevo modelo multirresolución para visualización interactiva de superficies poligonales complejas que permite, ante una petición, extraer aproximaciones eficientemente manteniendo un bajo coste de almacenamiento. Este nuevo esquema, que llamamos *Mallas Ordenadas Multirresolución* (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. El modelo consta de, por una parte, una estructura de datos básica que almacena las múltiples aproximaciones de un objeto y, por otra, una serie de algoritmos que extraen aproximaciones con nivel de detalle tanto uniforme como variable. Además, se ha realizado una serie de extensiones que permiten explotar la propiedad de la coherencia cuadro a cuadro y utilizar abanicos de triángulos con el fin de aumentar aún más las prestaciones y mejorar el comportamiento general del modelo. Las principales ventajas de este nuevo modelo 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 extracción de una aproximación, ya sea de nivel de detalle uniforme o variable, sean simples también.

- Generalidad. Cualquier método de simplificación puede ser utilizado para construir la representación multirresolución, lo que permite su uso en un mayor abanico de aplicaciones. Tampoco impone restricción alguna respecto al tipo de superficie poligonal a representar, pudiendo ser utilizado tanto para superficies *manifold* como para superficies *non-manifold*.

- Eficiencia. Los experimentos realizados muestran que el coste de almacenamiento de la representación multirresolución es menor que el de los modelos afines (modelos multirresolución continuos que no imponen restricciones acerca de la geometría del objeto y de su simplificación). Los experimentos también muestran que los tiempos de extracción de aproximaciones de nivel de detalle uniforme y variable presentan mejoras importantes respecto a los modelos afines.

1.4 Organización de este documento

Esta tesis está organizada en los siguientes capítulos:

- En el capítulo 2, se presenta un estudio detallado del modelado multirresolución y una revisión del trabajo realizado anteriormente en este campo. Se presentan las aplicaciones más interesantes y se realiza una caracterización de los modelos.

- En el capítulo 3, se introduce a *Mallas Ordenadas Multirresolución* describiendo sus características.

- En el capítulo 4, se presenta la estructura de datos básica sobre la que se asienta el nuevo modelo propuesto y se describe el proceso de construcción de una representación multirresolución. Se realiza un estudio teórico del coste de almacenamiento y se presentan resultados experimentales.

- En el capítulo 5, se trata el problema de la extracción de aproximaciones con nivel de detalle uniforme. Se detalla el algoritmo de acceso y se realiza un estudio teórico del coste temporal. Además, se incluye una comparación con el modelo *Progressive Meshes* a fin de obtener conclusiones sobre el comportamiento de ambos modelos.

- En el capítulo 6, se presenta una extensión del modelo que permite explotar la propiedad de la coherencia cuadro a cuadro. Se detalla la nueva estructura de datos así como los nuevos algoritmos de construcción y visualización. Se realiza un estudio teórico del coste de almacenamiento y del coste temporal y se incluyen resultados experimentales así como una comparación con el modelo básico.

- En el capítulo 7, se describe una nueva extensión del modelo que permite utilizar el abanico de triángulos como primitiva básica de representación. De nuevo, se detalla la estructura de datos y los algoritmos de construcción y de visualización, así como un estudio teórico del coste de almacenamiento y del coste temporal. También se incluyen resultados experimentales y estos se comparan con los obtenidos por el modelo básico.

- En el capítulo 8, se aborda el problema de la extracción de aproximaciones con nivel de detalle variable. Se explica el algoritmo propuesto y una serie de posibles optimizaciones. Se realiza un estudio teórico del coste temporal del algoritmo y se muestran los resultados experimentales obtenidos.

- Finalmente, en el capítulo 9, se resumen las conclusiones a las que se ha llegado tras la realización de este trabajo, enumerando las principales aportaciones así como una discusión sobre las líneas de trabajo futuro.

Capítulo 2

Trabajo Previo

El objetivo principal que se persigue en este capítulo es presentar un estudio detallado del modelado multirresolución y una revisión del trabajo realizado anteriormente en este campo. Una vez definido el término *Superficie poligonal*, se presenta una introducción sobre la simplificación de modelos poligonales como medio para la construcción de una representación multirresolución. A continuación, se describe qué es un modelo multirresolución y cuáles son las aplicaciones más interesantes. Finalmente se presenta una clasificación y una caracterización de los trabajos más destacables.

2.1 Superficie poligonal

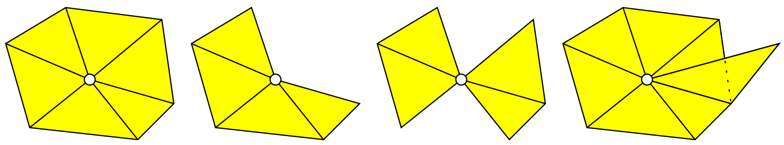
El modelado de superficies mediante mallas poligonales domina actualmente en el campo de los gráficos interactivos. Diversos motivos como su alta disponibilidad o el que la mayor parte de los programas de visualización existentes utilicen este tipo de modelos han hecho que ningún otro tipo de representación goce del mismo nivel de popularidad. De hecho, las aplicaciones que utilizan otro tipo de representación no poligonal convierten sus modelos a polígonos antes de ser procesados. Además, los sistemas gráficos que aceleran la visualización de polígonos han llegado a los pequeños equipos y, hoy en día, equipos de bajas prestaciones incluyen aceleradoras gráficas potentes y de bajo coste.

En general, una superficie poligonal no es más que un conjunto de polígonos planos en el espacio Euclideo tridimensional $\mathbb{R}^3$. Se puede asumir, sin pérdida de generalidad, que la superficie poligonal consta únicamente de triángulos[1], ya que cualquier polígono de un mayor número de lados puede ser triangulado [Heckbert, 1994], es decir, sustituido por un conjunto de triángulos. De esta forma, se puede definir un modelo de superficie poligonal, M, como la tupla $\{\mathcal{V}, \mathcal{T}\}$, donde:

[1]Hasta la fecha, no se tiene conocimiento de que exista un modelo multirresolución que no realice esta suposición.

- $\mathcal{V}$ 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.
- $\mathcal{T}$ 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.

En la presente tesis se referirá a una superficie poligonal como simplemente *superficie* o utilizando el término de *malla*. En la literatura, muchas veces se utilizan ambos términos para designar específicamente a una superficie poligonal *manifold* [Foley et al., 1990]. Una superficie poligonal *manifold* es aquella donde cada arista es utilizada por dos caras (o una en el caso de que la arista pertenezca a un borde), y la vecindad de cada vértice está formada por un bucle cerrado de caras (o en un único abanico de caras si pertenece a un borde). Si no cumple estas condiciones, a la superficie se le llama *non-manifold*. En la figura 2.1 se muestran los distintos casos. En esta tesis, ambos tipos de superficies poligonales quedan incluidos en el término de *superficie* o *malla*.



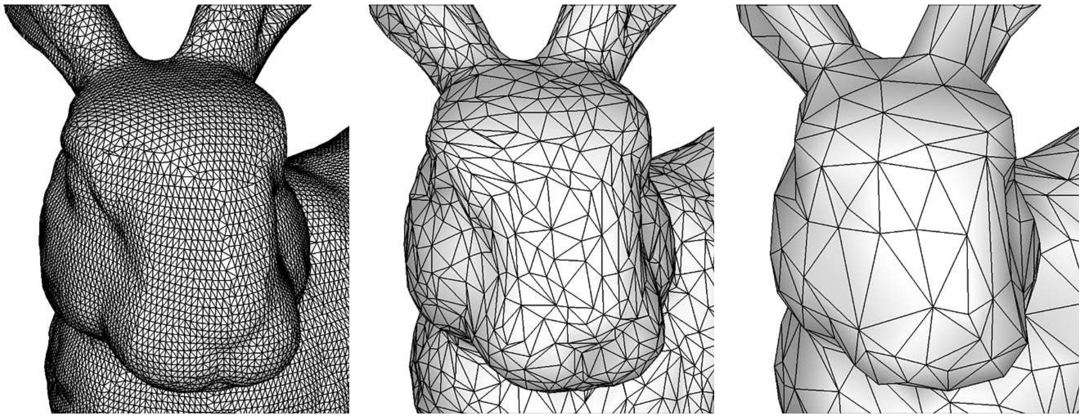
(a) **(b)** **(c)** **(d)**

Figura 2.1: Vecindad del vértice marcado en blanco. a) *Manifold*, b) *manifold con borde*, c) *non-manifold* por no haber un bucle cerrado de caras, d) *non-manifold* por arista compartida por más de dos caras.

2.2 Simplificación

Simplificar es reducir a una forma más simple mediante la eliminación de información repetida o la agrupación de elementos con características similares. Aplicado a una malla M, un método de simplificación produce, de forma automática, una nueva malla, M', que representa a M con un cierto error y que está formada por un número menor de polígonos (figura 2.2). A M' se le denomina *aproximación*, pero en la literatura también aparece como *resolución* o *nivel de detalle* (abreviadamente *lod*, de la expresión en inglés *level-of-detail*).

Si bien es cierto que la simplificación permite obtener la aproximación más adecuada según las necesidades de la aplicación, también lo es que posiblemente la solución a nuestro problema no consista en disponer de una única aproximación del modelo original. Por ejemplo, una aproximación de 100 polígonos es suficiente cuando el objeto se encuentra



(a) 69.451 caras **(b)** 8.307 caras **(c)** 1.557 caras

Figura 2.2: Ejemplo de simplificación del modelo *Bunny*. a) Modelo original, b) y c) modelos simplificados.

muy alejado del observador pero, si la distancia entre el objeto y el observador disminuye, es interesante disponer de una aproximación de mayor detalle. En este caso, lo más acertado es manejar múltiples aproximaciones y elegir la más adecuada utilizando como criterio la distancia al observador. Sin embargo, los métodos de simplificación no son lo suficientemente rápidos como para generar, en tiempo de ejecución, la aproximación que mejor se adapta a las condiciones de la aplicación. Esto nos lleva a utilizar, en lugar de un modelo de representación única, un modelo multirresolución.

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 [Heckbert y Garland, 1997].

Debido al interés específico de esta tesis en el modelado multirresolución de superficies poligonales, en esta sección se presenta una clasificación de los métodos de simplificación aplicables a este tipo de superficies. No se pretende hacer un estudio exhaustivo de los algoritmos existentes, sino hacer referencia sólo a los más representativos. Varios autores han realizado trabajos mucho más completos con el fin de clasificar y analizar los distintos métodos de simplificación [Erikson, 1996, Heckbert y Garland, 1997, Puppo y Scopigno, 1997, Andujar, 1998, Garland, 1999]. También se han presentado trabajos que muestran comparaciones de las prestaciones entre diferentes métodos [Cignoni et al, 1998a, Ciampalini et al., 1997, Lindstrom y Turk, 1998, Lindstrom y Turk, 1999], y se han creado herramientas que miden la calidad de las mallas obtenidas como resultado de la simplificación [Cignoni et al, 1998b].

Antes de comentar los distintos tipos de métodos es interesante destacar un par de características que van a permitir diferenciar entre los distintos algoritmos de simplificación. La primera característica es la conservación de la topología, es decir, si la representación simplificada tiene una topología equivalente a la original o si, por contra, existe reducción topológica. En algunas aplicaciones, como médicas por ejemplo, es necesario conservar la topología del objeto. Sin embargo, esto no es lo común en la mayor parte de las aplicaciones. Reducir la topología permite generar mejores aproximaciones donde, además, una simple textura puede simular la topología original del objeto. La segunda característica es el dominio geométrico de entrada, es decir, si el método impone restricciones geométricas a los objetos a simplificar. En concreto, interesa conocer si el método es aplicable únicamente a superficies *manifold*. Realmente, la mayor parte de las superficies que se pueden encontrar son superfices *manifold*, y siempre puede convertirse una superficie *non-manifold* a una *manifold* [Guéziec et al., 1998b]. Sin embargo, la propia simplificación de un *manifold* puede generar un *non-manifold* [Garland y Heckbert, 1997], pudiendo ser una ventaja el hecho de que un método de simplificación pueda manejar ambos tipos de superficies.

2.2.1 Agrupación de vértices

Estos métodos realizan una división de los vértices del objeto en un conjunto de agrupaciones, *clusters*, y unifica aquellos que residen en la misma agrupación. Suelen ser rápidos y normalmente no conservan la topología. El método de agrupación más simple es el presentado por Rossignac y Borrel [Rossignac y Borrel, 1996] basado en una división regular de una caja englobante. Los vértices que caen en una misma celda son agrupados, lo cual produce que los resultados dependan en gran medida de la propia ubicación de las celdas, además de los heurísticos utilizados para calcular el vértice representante del *cluster*. Centrando las celdas en vértices importantes [Low y Tan, 1997] se consigue mejorar las aproximaciones obtenidas con el método de Rossignac y Borrel.

2.2.2 Unión de regiones

Estos métodos se basan en la idea de que la sustitución de un numeroso conjunto de polígonos coplanares por un conjunto formado por unos pocos no cambia la apariencia del modelo. Kalvin y Taylor [Kalvin y Taylor, 1996] dividen la superficie en un conjunto de regiones utilizando un criterio de coplanaridad. La frontera de cada región es simplificada, y la región resultante triangulada. Este método es conocido como *Superfaces* y se caracteriza por preservar la topología y permitir acotar el error cometido en la aproximación. Otro método es el propuesto por Hinker y Hansen [Hinker y Hansen, 1993] más indicado para superficies suaves de baja curvatura. En general, los métodos basados en unión de regiones están restringidos a usarse con superficies *manifold*.

2.2.3 Wavelets

Los métodos basados en *wavelets* [Stollnitz et al., 1996] proveen de un marco de trabajo matemático para la descomposición de una superficie en una forma base más una secuencia de pequeños detalles. Eliminando los detalles menos significativos se obtienen las distintas aproximaciones. Lounsbery et al. [Lounsbery et al., 1997] desarrollaron un método para descomponer superficies siempre y cuando puedan obtenerse mediante subdivisión recursiva 4 a 1. Posteriormente, Eck et al. [Eck et al., 1995] eliminaban esta restricción realizando un remallado previo, es decir, convirtiendo una malla con conectividad arbitraria en una con subdivisión recursiva. Sin embargo, al añadir una etapa de remallado se introduce un error en la nueva malla. En general, estos métodos no realizan cambios en la topología del objeto.

2.2.4 Eliminación de vértices

Los métodos basados en eliminación de vértices realizan la simplificación de la superficie de forma iterativa. A cada paso, un vértice es seleccionado y eliminado, todas las caras que lo utilizan también son eliminadas, y el agujero resultante es triangulado. Originalmente fue propuesto por Schroeder et al. [Schroeder et al., 1992]. Más tarde, Schroeder [Schroeder, 1997] mejoró el método original para poder aplicarse a superficies *non-manifold* y que permitiera la simplificación de la topología. Se han presentado otros métodos [Ciampalini et al., 1997, Kobbelt et al., 1998a] que difieren, principalmente, en la métrica de error utilizada.

2.2.5 Contracción iterativa

La mayor parte de los algoritmos clasificados en este punto realizan contracción iterativa de pares de vértices, aunque se han presentado métodos basados en contracción de caras [Gieng et al., 1998]. De entre los que realizan contracción de pares de vértices, algunos exigen que el par elegido sea una arista [Algorri y Schmitt, 1996, Guéziec, 1996, Lindstrom y Turk, 1998, Hoppe, 1996, Ronfard y Rossignac, 1996] mientras que otros no [Garland y Heckbert, 1997, Popovic y Hoppe, 1997]. Este tipo de métodos permiten la simplificación de la topología y su aplicación a superficies *non-manifold*. Tal vez es el tipo de simplificación que más interés ha suscitado recientemente y que más estrechamente se ha relacionado con el modelado multirresolución. Probablemente el motivo es que estos métodos de simplificación crean de forma natural una jerarquía de vértices fácilmente representable mediante una estructura de árbol binario.

2.3 Modelado multirresolución

Un modelo multirresolución representa a un objeto mediante un conjunto de aproximaciones de diferente nivel de detalle y permite recuperar cualquiera de ellas bajo demanda. Los modelos multirresolución que aparecieron inicialmente estaban basados en un número de aproximaciones relativamente pequeño (normalmente, entre 5 y 10 [Funkhouser et al., 1992]) y se les ha denominado modelos multirresolución discretos (en la sección 2.4 se revisan este tipo de modelos, sus características y sus limitaciones). 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 (este tipo de modelos se revisan con mayor profundidad en la sección 2.5).

Algunos autores [Puppo y Scopigno, 1997, Huerta, 1998] 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 [Hoppe, 1998b, Garland, 1999] 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 [Floriani et al., 1996, Heckbert y Garland, 1997], principalmente debido a su aplicación en sistemas de información geográfica y simuladores de vuelo [Puppo y Scopigno, 1997]. De hecho, los primeros métodos de refinamiento y simplificación [Fowler, 1979, Lee, 1991] 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* [Taylor y Barret, 1994], *triangulaciones cuaternarias* [Lounsbery et al., 1997], o *triangulaciones jerárquicas más generales* [Lindstrom et al., 1996, Cignoni et al, 1997, Floriani et al., 1996, Scarlatos, 1990, Duchaineau et al., 1997, Hoppe, 1998b].

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. En las secciones 2.5 y 2.6 se realiza un estudio más detallado de estos modelos.

Por último, cabe señalar que el modelado multirresolución no ha sido aplicado exclusivamente a modelos de superficies poligonales. Una familia de esquemas de representación ampliamente utilizados son los modelos de descomposición espacial. Algunas soluciones aplicadas a este tipo de modelos son [Huerta, 1998, Andujar, 1999].

2.3.1 Aplicaciones

Acelerar el proceso de visualización fue la primera aplicación del modelado multirresolución. Los primeros modelos multirresolución surgieron con este propósito. Sin embargo, a raíz de la aparición de los modelos multirresolución continuos, se encontraron nuevas aplicaciones tan interesantes y que han recibido tanta atención como esta. Las que mayor interés han despertado son, además de la visualización interactiva, la transmisión progresiva, la compresión de geometría, y el refinamiento adaptativo. En un principio, los modelos multirresolución intentaron dar solución por igual a este conjunto de problemas. Sin embargo, pronto se especializaron en una u otra aplicación, intentando dar una solución más eficiente ante un problema concreto en lugar de ser una solución general a todos ellos.

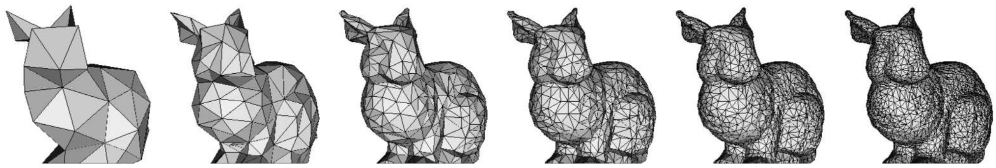
A continuación se realiza un pequeño resumen de las aplicaciones mencionadas anteriormente citando los trabajos más representativos de cada una de ellas. Ya que el objetivo de esta tesis es la aplicación del modelado multirresolución a la visualización interactiva y al refinamiento adaptativo, los trabajos orientados hacia estas aplicaciones son analizados con mayor profundidad en puntos posteriores de este capítulo.

Transmisión Progresiva

El ancho de banda requerido para acceder a modelos 3D, sobre Internet y en aplicaciones gráficas interactivas excede, en gran medida, la capacidad de las actuales líneas de comunicación. Esto produce que el tiempo que transcurre desde que un usuario solicita un modelo 3D, hasta que se visualiza en su monitor, sea demasiado alto. La transmisión progresiva trata de reducir este tiempo de espera.

Una primera solución realmente simple consiste en transmitir una secuencia de aproximaciones en orden de complejidad creciente. Durante la transmisión, al mismo tiempo que se visualiza la última aproximación recibida, se recibe la siguiente de mayor detalle. Este método tiene las desventajas de enviar información redundante y de no poder visualizar una aproximación hasta que se haya recibido por completo.

Una segunda posibilidad, que evitaría ambos problemas, consiste en transmitir, en primer lugar, la aproximación más burda del objeto a modo de malla base. A continuación, se envía una secuencia de registros. Cada registro contiene la información necesaria que permite añadir detalle. A medida que los registros se reciben, estos son añadidos a la malla base, actualizándose y llegando a reconstruir el modelo. De esta forma, el proceso receptor puede visualizar la primera aproximación recibida que, aunque burda, siempre será una aproximación completa del objeto, e ir refinándola y visualizando mayor detalle al mismo tiempo que recibe la secuencia de registros. Además, el usuario podría incluso especificar el grado de detalle con que desea recibir el objeto, o simplemente cortar la transmisión cuando decida que el nivel de detalle recibido es suficiente. En la figura 2.3 se muestran seis instantes de la transmisión del modelo *Bunny* (modelo compuesto por 34.834 vértices y 69.451 caras).



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figura 2.3: Seis instantes en el proceso de transmisión desde la malla base, 99 caras, a una aproximación formada por 8.765 caras. a) 0'005 T, b) 0'03 T, c) 0'06 T, d) 0'11 T, e) 0'19 T, f) 0'33 T, donde *T* representa el tiempo total de transmisión.

En la literatura se pueden encontrar diversos modelos multirresolución que siguen el esquema de representación explicado en el párrafo anterior (malla base + secuencia de actualizaciones) y, normalmente, son denominados *representaciones incrementales*. El modelo propuesto por Hoppe [Hoppe, 1996], llamado *Progressive Meshes*, ha servido como referencia para el desarrollo de otros muchos [Popovic y Hoppe, 1997, Abadjev et al., 1999, Hoppe, 1998a, Guéziec et al., 1998a, Taubin et al., 1998]. Una revisión de mayor profundidad sobre estos modelos se puede encontrar en [Taubin, 1999]. También, los modelos multirresolución basados en wavelets responden al mismo esquema de representación y resultan muy apropiados para realizar una transmisión progresiva [Lounsbery et al., 1997, Eck et al., 1995, Certain et al., 1996].

Compresión

Si transmitir un objeto 3D, formado por cientos de miles de polígonos, a través de una línea de comunicación es un problema, no lo es menos el coste que supone almacenar tal cantidad de datos. Las técnicas de compresión tratan de minimizar el espacio requerido para almacenar una representación y, salvo que se especifique lo contrario, comprimir no implica perder información, lo cual realmente sería una simplificación.

El modelado multirresolución puede utilizarse como medio de compresión. El esquema seguido es básicamente el mismo que el utilizado para la transmisión progresiva. Esto es, una malla base correspondiente a la representación más burda, y una secuencia de registros que permiten refinar y reconstruir el objeto 3D. La información que almacena cada registro afecta a una zona muy localizada de la superficie, de forma que los elementos afectados pueden ser codificados de forma mucho más eficiente.

Los mismos modelos propuestos para transmisión progresiva consiguen codificar una malla con unos requerimientos de espacio menor [Hoppe, 1996, Popovic y Hoppe, 1997, Abadjev et al., 1999, Hoppe, 1998a, Guéziec et al., 1998a, Taubin et al., 1998, Lounsbery et al., 1997, Eck et al., 1995, Certain et al., 1996]. De todos ellos, el esquema presentado por Taubin et al. [Taubin et al., 1998], llamado *Progressive Forest Split*, es el más eficiente requiriendo sólo de $O(V)$ bits para aumentar al doble el número de vértices de una malla formada por V vértices.

Sin embargo, los esquemas de compresión de modelos de resolución única consiguen relaciones de compresión mayores. Taubin et al. [Taubin et al., 1999], introducen un método para comprimir una representación multirresolución producida por un método de agrupación de vértices con relaciones de compresión comparables a las de los métodos aplicados a representaciones de resolución única. Por contra, este esquema no permite la transmisión progresiva del modelo. Ver [Taubin, 1999, Taubin y Rossignac, 1998, Taubin y Rossignac, 1999] para una revisión mas extensa sobre compresión de geometría.

Visualización interactiva

La mayoría de los algoritmos de visualización son ineficientes a la hora de pintar escenas complejas. Estos algoritmos tienen un coste lineal respecto a la complejidad de la escena y esto es inaceptable cuando la complejidad es muy alta. El modelado multirresolución consigue que el coste de visualización de una escena sea proporcional al área que ocupa en la pantalla y sea independiente de su complejidad [Heckbert y Garland, 1994]. Dependiendo de ese área, el objeto o conjunto de objetos que forman la escena son visualizados con un nivel de detalle apropiado [Clark, 1976]. El nivel de detalle apropiado para una determinada vista será aquel cuya apariencia sea igual al nivel de mayor detalle. De esta forma, los objetos cercanos al observador se pintan con gran detalle, mientras que los más distantes son visualizados utilizando un nivel de detalle menor.

En cada cuadro, la selección del nivel de detalle la realiza un algoritmo como resultado de la evaluación de un criterio o conjunto de criterios. El más común es precisamente el de la distancia al observador (figura 2.4). Otros aspectos como las prestaciones del sistema o la necesidad de mantener una tasa fija de imágenes por segundo [Funkhouser y Séquin, 1992] limitan la complejidad de la escena y, en consecuencia, obligan a ajustar el nivel de detalle de los objetos que la forman. Diversos autores [Krus, 1997, Andujar, 1998] han enumerado los distintos factores que se han utilizado para realizar la selección de un determinado nivel de detalle, que se pueden resumir en los siguientes:

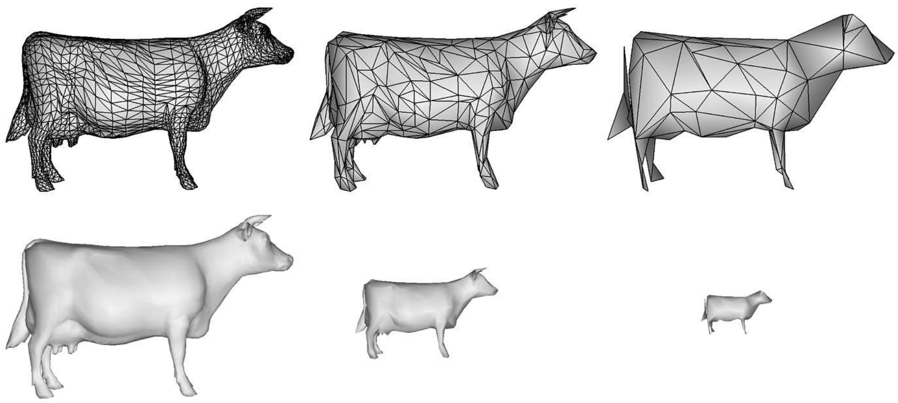


Figura 2.4: Tres aproximaciones del modelo *Cow* simulando el nivel de detalle a utilizar según la distancia al observador.

Distancia. Dado un objeto, a mayor distancia respecto del observador menor será su área proyectada en la pantalla y menor el nivel de detalle. Para hacer una rápida estimación se suelen englobar a los objetos en cajas o esferas.

Incidencia. Ciertos objetos, según el ángulo de la vista, pueden verse de formas distintas. Por ejemplo, una puerta vista de perfil, aún estando cerca del observador, necesitaría de muy pocos polígonos para representarla.

Movimiento. Un objeto animado no necesita representarse con grandes detalles ya que éstos apenas serán perceptibles [Reddy, 1994]. Además, a mayor velocidad, mayor dificultad para percibir el detalle.

Visión periférica. Dibujar con más detalle aquellos objetos que están en el centro del campo de visión (suponiendo que es ahí donde reside el interés), y con menos detalle aquellos que quedan alrededor de dicho centro.

Importancia semántica. Utilizar niveles de detalle altos sólo en aquellos objetos que en un momento dado tienen mayor importancia en la escena, por ejemplo, porque se estén utilizando para hacer una determinada tarea.

Cambios de representación sucesivos. Visualizar un objeto utilizando diferentes niveles de detalle en cuadros consecutivos podría reducir la calidad en la secuencia de imágenes. A la hora de seleccionar un nuevo nivel de detalle, es importante tener en cuenta la diferencia que existe entre este y el último nivel de detalle visualizado.

Refinamiento Adaptativo

Los modelos multirresolución, tal y como se han presentado en el apartado anterior, son muy útiles en una gran variedad de aplicaciones. Sin embargo, si el objeto se encuentra en una posición cercana a la del observador, no es posible hacer uso de aproximaciones burdas. Si además esto ocurre en aplicaciones donde el objetivo es visualizar objetos (aunque sólo sean uno o dos) que contienen un alto grado de detalle, resulta casi imposible conseguir una rápida visualización. Por ejemplo, un bioquímico visualizando la superficie de una molécula, o un físico examinando la iso-superficie de una cabeza humana extraída a partir de un conjunto de datos volumétricos, o simplemente una malla de terreno en un simulador de vuelo. En todos estos casos, lo deseable sería disponer de un mecanismo que permita representar con más detalle las zonas del objeto cercanas al observador, y con menos detalle las zonas que, aún siendo del mismo objeto, se encuentran alejadas del observador. Desde un punto de vista más general, el objetivo es representar un objeto con diferentes niveles de detalle coexistiendo a lo largo de la misma superficie. Este tipo de esquema multirresolución necesita satisfacer dos criterios importantes:

1. Debe ser posible seleccionar en tiempo real los niveles de detalle apropiados a lo largo de las diferentes regiones del mismo objeto.
2. Los distintos niveles de detalle deben unirse unos con otros sin introducir ningún tipo de rotura o discontinuidad en la superficie.

El modelado multirresolución permite resolver este problema y a esta técnica se le conoce como *refinamiento adaptativo* o *resolución variable* (figura 2.5). En ocasiones, las condiciones de la vista se utilizan para determinar la zona de la superficie a representar con mayor detalle y, en estos casos, se utiliza el término *visualización dependiente de la vista*.

Este tipo de aplicación se ha desarrollado principalmente durante los últimos años. Floriani et al. [Floriani et al., 1997a, Puppo, 1998] presentaron un marco de trabajo general llamado *Multi-Triangulation* donde cualquier representación multirresolución queda descrita. En [Floriani et al., 1997b, Floriani et al., 1998] los mismos autores describen una implementación eficiente de una Multi-triangulación y varias estructuras de datos alternativas para su codificación. Una representación particular que ha recibido especial atención son los modelos basados en jerarquías de vértices, motivados principalmente por los métodos de simplificación utilizados para construir la representación multirresolución, basados en contracción de aristas o agrupación de vértices. Así, se encuentran en este grupo los modelos propuestos por Xia et al. [Xia et al., 1997], Hoppe [Hoppe, 1997], Luebke y Erikson [Luebke y Erikson, 1997], Guéziec et al. [Guéziec et al., 1999] y El-Sana y Varshney [El-Sana y Varshney, 1999].

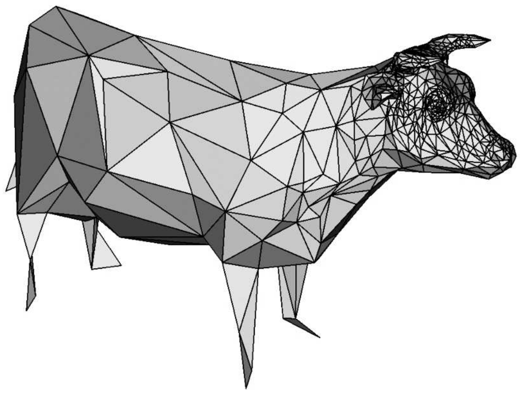


Figura 2.5: Ejemplo de refinamiento adaptativo. El modelo *Cow* se representa con mayor detalle en la cabeza y este disminuye hacia la cola.

Al igual que en el apartado anterior, la decisión de qué regiones de la superficie del objeto se han de visualizar con mayor o menor detalle depende exclusivamente del criterio o conjunto de criterios que se utilicen. Algunos autores [Hoppe, 1997, Xia et al., 1997, Luebke y Erikson, 1997, El-Sana y Varshney, 1999] han definido dichos criterios que se resumen en 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 haya una zona más cercana 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.

Otras aplicaciones

La edición de mallas es otro de los problemas que se presentan cuando estas están formadas por un gran número de polígonos. La edición multirresolución permite que las modificaciones que afectan a zonas grandes de la malla se puedan hacer con pequeño esfuerzo si se utilizan las aproximaciones de menor detalle. Por contra, cuando lo que se desea es realizar modificaciones puntuales (pequeños detalles de la malla) se utilizan aproximaciones de mayor detalle. Cuando el usuario selecciona una aproximación de un nivel de detalle cualquiera, la superficie se representa mediante esa aproximación más el conjunto de todos los pequeños detalles que refinan la superficie. El usuario puede manipular la aproximación visualizada mientras que los detalles permanecen al margen y siempre relativos a dicha aproximación. Esta idea ya aparece en métodos de edición de curvas [Finkelstein y Salesin, 1994] y, posteriormente, Zorin et al. [Zorin et al., 1997] la aplican a mallas generadas por subdivisión. Kobbelt et al. [Kobbelt et al., 1998b] presentan una solución para mallas irregulares, eliminando la restricción de la subdivisión.

La posibilidad de realizar eficientemente cálculos geométricos como localización de un punto, intersección con rayos, detección de colisiones, recortado, etc., son una aplicación más del modelado multirresolución. Las técnicas más sencillas se basan en volúmenes englobantes que, a cambio de un pequeño coste adicional, permiten simplificar los cálculos al no ser necesario recurrir a la representación de la malla. El uso de la multirresolución se basa en la misma idea, es decir, realizar los cálculos con la peor aproximación, como si se tratara de un volumen englobante, y refinar donde sea necesario si hace falta una mayor precisión. También, métodos de radiosidad, por ejemplo, pueden ser acelerados significativamente utilizando técnicas basadas en multirresolución [Teller et al., 1994, Willmott et al., 1999].

2.4 Modelos multirresolución discretos

La forma más sencilla de crear un modelo multirresolución es generar un conjunto de aproximaciones independientes donde cada una de ellas representa al objeto original con un nivel de detalle distinto. Si se pueden obtener estas aproximaciones, ya tenemos un modelo multirresolución discreto. La simplicidad de esta técnica es probablemente su principal atractivo.

Estos modelos empezaron a utilizarse con el objetivo puesto en aumentar las prestaciones del sistema gráfico y así lo han demostrado en aplicaciones de navegación en entornos virtuales tal y como proponen Funkhouser y Séquin [Funkhouser y Séquin, 1992] o Maciel y Shirley [Maciel, 1995]. Hoy en día existen diversas aplicaciones que hacen uso de los modelos discretos como, por ejemplo, *Open Inventor* [Wernecke, 1993], *IRIS Performer* [Rohlf y Helman, 1994] o *VRML* [Bell et al., 1995]. Reddy [Reddy, 1995] presenta un resumen de los motores gráficos utilizados en Realidad Virtual y en qué medida hacen uso de este tipo de modelos.

2.4.1 Transición entre aproximaciones

El principal problema de los modelos discretos es el de la transición entre aproximaciones diferentes. Cuando se está visualizando una aproximación y el criterio de selección determina que se ha de visualizar otra con un nivel de detalle distinto, es posible percibir un salto de calidad entre la representación anterior y la nueva. Este hecho resulta evidente ya que ambas representaciones están formadas por un número diferente de caras, lo cual produce que el aspecto en la imagen final también lo sea. El problema ha sido estudiado y hay dos posibles soluciones:

Transparencia. Además de visualizar la aproximación requerida, dibujar también la aproximación anterior (la que se estaba visualizando hasta el momento) y ambas de forma transparente durante varios cuadros a fin de que la mezcla de las dos elimine la percepción del salto [Funkhouser et al., 1992]. Es cierto que el salto sigue existiendo pero resulta difícil de observar.

Morphing. Con este nombre se denominan las técnicas que son capaces de convertir de forma suave un objeto en otro mediante una interpolación entre las características de ambos objetos.

Ambos métodos incurren en una sobrecarga en el sistema. Por una parte, el método basado en transparencia obliga a que durante una serie de cuadros sean dos las aproximaciones que se hayan de visualizar, con el consecuente aumento del coste de visualización. Por otra, el método basado en *morphing* introduce una sobrecarga computacional debida al cálculo, para cada cuadro, de la interpolación de los vértices entre las dos aproximaciones.

2.4.2 Limitaciones

Además del problema de la transición entre niveles de detalle, señalado en el apartado anterior, los modelos discretos presentan algunas otras limitaciones. El número máximo de aproximaciones que almacenan es una de las más importantes. Si bien es cierto que teóricamente ese máximo podría ser un número alto, el coste que supone almacenar de forma independiente todas y cada una de las aproximaciones obliga a que ese número sea pequeño (entre 5 y 10 normalmente).

Otra de las limitaciones que presentan este tipo de modelos es la imposibilidad de ajustar de forma precisa el nivel de detalle, es decir, de adaptar la resolución correcta del objeto a las condiciones de la aplicación. Al disponer de sólo un número discreto de aproximaciones el algoritmo que, para cada cuadro, selecciona la aproximación más adecuada se verá obligado a escoger en bastantes ocasiones o bien una aproximación de mayor detalle o bien una aproximación de menor detalle del que realmente es necesario.

Por último, este tipo de modelos no pueden utilizarse para realizar refinamiento adaptativo, es decir, representar un objeto con diferente nivel de detalle a lo largo de su superficie tal y como se comentó en la sección 2.3.1.

2.5 Modelos multirresolución continuos

Los modelos multirresolución continuos surgen con el fin de solucionar los problemas que presentan los modelos discretos. Un modelo multirresolución continuo provee de un amplio rango (virtualmente un rango continuo) de aproximaciones distintas que representan al objeto original. La aplicación iterativa de un método de simplificación, comenzando sobre M_0, produce una secuencia de aproximaciones $M_1, M_2, ..., M_{n-1}$, donde M_{n-1} se corresponde con la aproximación de menor detalle. Un modelo multirresolución continuo codifica el modelo original, M_0, la aproximación más burda, M_{n-1}, y toda la secuencia de aproximaciones $M_1, M_2, ..., M_{n-2}$. Los métodos de simplificación de superficies poligonales animan a formalizar modelos de este tipo debido a que la diferencia entre dos aproximaciones consecutivas suele ser de un vértice, una arista o un triángulo.

El número de aproximaciones que almacenan los modelos continuos no es fijo aunque sí proporcional al tamaño del objeto original. Es evidente que manipular eficientemente tal cantidad de información exige de un cuidado diseño de las estructuras de datos y de los algoritmos de acceso, repercutiendo directamente en las prestaciones finales del modelo y de la aplicación. Sería inutil un modelo en el que la sobrecarga que produce manipular las distintas aproximaciones sea tal que impida acelerar la visualización, disminuir los tiempos de transmisión, etc.

No existen unas condiciones que los modelos multirresolución continuos deban satisfacer, pero sí que hay una serie de propiedades que son deseables. Son las siguientes:

Procesamiento de la información eficiente. Si el modelo almacena n distintas aproximaciones, la información debe estar organizada de manera que, en tiempo de ejecución y ante la petición de una de las n aproximaciones, el algoritmo de recorrido pueda extraer los datos en un periodo de tiempo corto.

Coste de almacenamiento similar al del objeto original. Es importante que el coste de almacenamiento del modelo multirresolución no sea considerablemente mayor que el del objeto original, aunque un factor pequeño y constante de incremento suele ser aceptable.

Superficie continua. La superficie extraída no debe presentar rotura o agujero alguno para cualquiera de las aproximaciones posibles.

Transición suave entre aproximaciones. Pasar de una aproximación a otra (ya sea para incrementar o decrementar detalle) se realizará de la forma más suave posible, evitando los cambios bruscos.

Sin pérdida de información. El modelo multirresolución debe ser capaz de representar con exactitud al modelo original.

2.5.1 Multi-Triangulation (MT)

La Multitriangulación [Puppo, 1996, Puppo, 1998] 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 construida 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.

Una MT se basa en las dos siguientes observaciones:

1. Una secuencia de mallas puede ser vista como una malla inicial, la de menor detalle, más una secuencia de modificaciones locales, donde cada modificación se especifica mediante la zona de superficie que modifica y a la que se denomina *fragmento*.
2. Hay fragmentos en la secuencia de mallas que son mutuamente independientes, y otros no.

La segunda observación invita a que los fragmentos se dispongan en un orden parcial, el cual puede ser representado por un grafo acíclico dirigido, teniendo como raíz la malla inicial, y un arco entre cada par de nodos de tal forma que el nodo inicial del arco tiene algún triángulo cubierto por el fragmento correspondiente al nodo final. En [Klein y Strasser, 1997, Floriani et al., 1997b, Floriani et al., 1998] se describen diversas implementaciones de una MT proponiendo varias estructuras de datos alternativas para su codificación.

2.5.2 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. [Eck et al., 1995] presentaron un modelo basado en *wavelets* aplicable a superficies con subdivisión conectada y fue extendido por Certain et al. [Certain et al., 1996] para incluir, además de la geometría, otros atributos como, por ejemplo, el color. Más tarde, Hoppe presentó el esquema conocido como *Progressive Meshes* (PM) [Hoppe, 1996], extendido por Popovic y Hoppe [Popovic y Hoppe, 1997] para permitir cambios en la topología, y por Hoppe [Hoppe, 1998a] para codificar de forma eficiente un objeto PM. Otros modelos son el de Cignoni et al. [Cignoni et al, 1997], Taubin et al. [Taubin et al., 1998], Gueziec et al. [Guéziec et al., 1998a] y Abadjev et al. [Abadjev et al., 1999]. Debido a que todos estos esquemas están estrechamente relacionados con el modelo multirresolución que se propone en esta tesis, son objeto de un mayor estudio en la sección 2.6.

2.5.3 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 conseguir el refinamiento adaptativo (sección 2.3.1), es decir, representar a un objeto utilizando diferentes niveles de detalle coexistiendo al mismo tiempo en distintas zonas de la misma superficie. Este tipo de representaciones 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 [Xia y Varshney, 1996, Hoppe, 1997, Luebke y Erikson, 1997]. 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. [Xia y Varshney, 1996], Hoppe [Hoppe, 1997], Guéziec et al. [Guéziec et al., 1999], El-Sana et al. [El-Sana et al., 1999], El-Sana y Varshney [El-Sana y Varshney, 1999] pertenecen a este grupo. Sólo el esquema propuesto por Luebke y Erikson [Luebke y Erikson, 1997] 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 (ver figura 2.6). 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 (ver figura 2.7), 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 [El-Sana y Varshney, 1999].

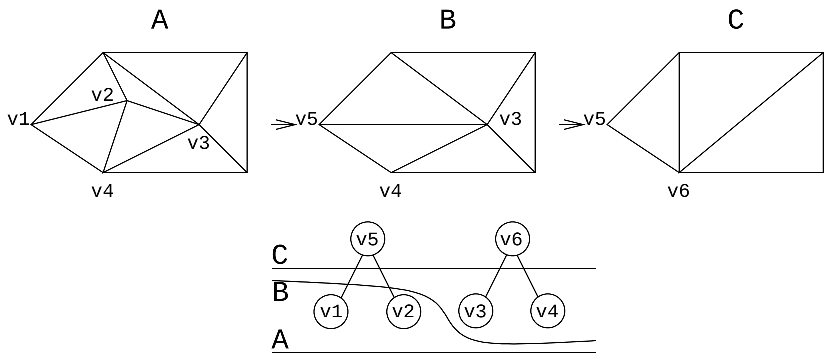


Figura 2.6: Ejemplo de representación jerárquica. En la malla A se simplifica la arista v1-v2 llevando v2 a la posición de v1 creando como resultado v5. En la malla B se simplifica la arsita v3-v4 llevando v3 a la posición de v4 creando v6. Debajo se muestra la representación jerárquica y los cortes asociados a las respectivas mallas.

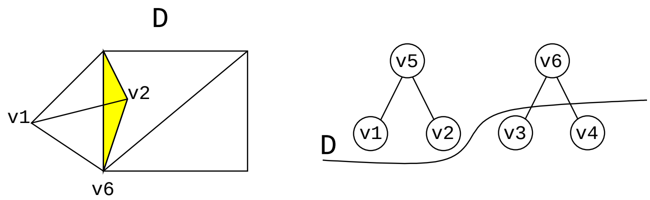


Figura 2.7: Ejemplo de malla errónea extraída de una representación jerárquica. Se han contraído los nodos v3 y v4 originando un problema de plegamiento sobre la malla por no haber contraído previamente los nodos v1 y v2.

Todos estos esquemas están estrechamente relacionados con el modelo que se propone en esta tesis, por lo que también son objeto de un mayor estudio en la sección 2.6.

2.6 Caracterización

En esta sección se proponen diferentes criterios que permiten caracterizar los modelos multirresolución de superficies poligonales. Estos criterios se clasifican según hacen referencia a la entrada del modelo, a su aplicación, o al propio funcionamiento.

2.6.1 Criterios sobre los datos de entrada

Para obtener una representación multirresolución es necesario disponer del objeto del cual se desea generar la correspondiente representación y de una herramienta de simplificación que genere las sucesivas aproximaciones. Estos criterios hacen referencia a aspectos relacionados con ambos.

Dominio geométrico. Restricciones sobre los elementos geométricos (vértices, aristas o caras) y sobre las relaciones topológicas, de forma que los objetos a modelar quedan supeditados a cumplir dichas restricciones. Esta limitación es importante ya que en general las restricciones geométricas pueden aprovecharse con el fin de conseguir representaciones multirresolución más eficientes.

Codificación de otros atributos. Una superficie poligonal, además de vértices y caras, suele tener atributos de apariencia como color, normales y texturas, utilizados a la hora de visualizar la superficie. No todos los modelos se han preocupado de presentar soluciones que permitan codificar eficientemente los atributos de apariencia.

Método de simplificación. Para poder construir una representación multirresolución es necesario un método de simplificación. Ciertos modelos exigen que dicho método actúe de una forma determinada, por ejemplo, eliminando vértices, colapsando aristas, etc. Otros no imponen condición alguna.

2.6.2 Criterios sobre la aplicación

Los modelos multirresolución se han especializado en dar una solución eficiente a una o varias aplicaciones concretas. Estos criterios hacen referencia a las distintas aplicaciones.

Compresión. Almacenar las n distintas aproximaciones de forma que el coste de almacenamiento sea menor que el del modelo original. En general, el ahorro de memoria va acompañado de un mayor coste de acceso a los datos del modelo.

Transmisión progresiva. Transmitir un objeto de forma progresiva tal y como se comentó en la sección 2.3.1. Es fácil que un modelo multirresolución permita la transmisión progresiva, pero sólo algunos se han preocupado en especial por esta aplicación y han optimizado la cantidad de información a transmitir.

Visualización interactiva. Capacidad del modelo para recuperar y visualizar una aproximación de forma eficiente. Se pretende distinguir a los modelos que especialmente se han desarrollado para esta aplicación. En general, estos modelos ponen más interés en el diseño de estructuras de datos que permitan a los algoritmos de acceso recuperar una aproximación en el menor tiempo posible, aunque esto suponga un incremento en el coste de almacenamiento.

Refinamiento adaptativo. Capacidad del modelo para representar una superficie con resolución variable (ver sección 2.3.1). En realidad, el refinamiento adaptativo es una especialización de la visualización interactiva. Al igual que con los modelos aplicados a visualización interactiva, se da prioridad a disminuir el tiempo de visualización aunque sea a costa de aumentar el coste de almacenamiento.

2.6.3 Criterios sobre el funcionamiento del modelo

Estos criterios tratan aspectos relacionados con la representación interna de los datos y del funcionamiento de los algoritmos de acceso.

Almacenamiento de la información. Se distinguen los modelos que almacenan la información de forma explícita de aquellos que codifican ciertos datos de forma implícita y necesitan de una operación que permita recuperarlos. A la hora de extraer un nivel de detalle, los modelos que almacenan los datos de forma explícita no necesitan de operaciones de cálculo y, en consecuencia, tienden a ofrecer un comportamiento temporal más eficiente aunque también un coste de almacenamiento algo mayor.

Coherencia cuadro a cuadro. Capacidad de reducir el coste de extracción de una aproximación utilizando los datos correspondientes a la última aproximación visualizada. Suele ocurrir que la diferencia entre las aproximaciones requeridas en cuadros consecutivos sea pequeña. Explotar la propiedad de la coherencia cuadro a cuadro puede significar un aumento importante de las prestaciones, sobre todo en modelos dedicados a la visualización interactiva o al refinamiento adaptativo.

Geomorphing. Transición suave entre aproximaciones mediante la interpolación geométrica de los vértices. Los modelos que requieren un método de simplificación basado en contracción de aristas suelen poseer esta propiedad, muy difícil de conseguir en los que utilizan otros métodos de simplificación. También es cierto que hacer uso del *geomorphing* produce una sobrecarga notable [Hoppe, 1998a].

2.6.4 Caracterización

En esta sección se recogen los principales modelos multirresolución de superficies poligonales (ver tabla 2.1) exceptuando los aplicados a campos de altura. La caracterización

Acrónimo	Nombre	Referencia
MRA	Multiresolution Analysis	[Certain et al., 1996]
VDS	View-Dependent Simplification	[Xia y Varshney, 1996]
PM	Progressive Meshes	[Hoppe, 1996, Hoppe, 1998a]
PSC	Progressive Simplicial Complexes	[Popovic y Hoppe, 1997]
VDPM	View-Dependent of PM	[Hoppe, 1997]
HDS	Hierarchical Dynamic Simplification	[Luebke y Erikson, 1997]
MT	Multi-Triangulation	[Floriani et al., 1998]
PFS	Progressive Forest Split	[Taubin et al., 1998]
SM	Simplicial Maps	[Guéziec et al., 1998a]
MS	MetaStream	[Abadjev et al., 1999]
SG	Streaming Geometry	[Guéziec et al., 1999]
GVDS	Generalized View-Dependent Simplification	[El-Sana y Varshney, 1999]

Tabla 2.1: Modelos multirresolución objeto de la caracterización.

Modelo	D. Geométrico	Atributos	Simplificación
MRA	Conect. por subdivisión [2]	Si	Wavelets
VDS	Manifold	No	Contracción de aristas [3]
PM	Manifold	Si	Contracción de aristas
PSC	Non-manifold	Si	Unión de pares de vértices [4]
VDPM	Manifold	Si	Contracción de aristas
HDS	Non-manifold	Si	Agrupación de vértices
MT	Non-manifold	No	Eliminación de vértices [5]
PFS	Manifold	Si	Contracción de aristas
SM	Manifold	Si	Contracción de aristas
MS	Non-manifold	Si	Intercambio y Contrac. de aristas
SG	Non-manifold	Si	Agrupación de vértices
GVDS	Non-manifold	No	Unión de pares de vértices

Tabla 2.2: Caracterización utilizando los criterios sobre los datos de entrada.

se ha dividido en tres tablas según los criterios utilizados: datos de entrada, tabla 2.2; aplicación, tabla 2.3; y funcionamiento, tabla 2.4.

[2] En [Eck et al., 1995] los autores eliminan esta restricción mediante un remallado de la superficie

[3] Los autores afirman que aunque en la implementación realizada utilizan un método de simplificación basado en contracción de aristas, el modelo propuesto puede fácilmente ser extendido para usar otros métodos de simplificación

[4] Los pares de vértices no tienen porqué formar una arista

[5] Aunque una representación MT no condiciona el método de simplificación a utilizar, la implementación propuesta en [Floriani et al., 1998] se basa en un método de eliminación de vértices

Modelo	Compresión	Transmisión	Vis. Interactiva	Ref. Adaptativo
MRA	Si	Si	Si	No
VDS	No	No	Si	Si
PM	Si	Si	Si	No [6]
PSC	Si	Si	Si	No
VDPM	No	No	Si	Si
HDS	No	No	Si	Si
MT	No [7]	No	Si	Si
PFS	Si	Si	No [8]	No
SM	No	Si	Si	No
MS	Si	Si	Si	No
SG	Si	Si	Si	No [9]
GVDS	No	No	Si	Si

Tabla 2.3: Caracterización utilizando los criterios sobre la aplicación.

Modelo	Almacenamiento	Coherencia	Geomorphing
MRA	Implícito	No [10]	No
VDS	Implícito	Si	No
PM	Implícito	Si	Si
PSC	Implícito	Si	Si
VDPM	Implícito	Si	Si
HDS	Implícito	Si	No
MT	Ambos	Si	No
PFS	Implícito	No	Si
SM	Implícito	Si	No
MS	Implícito	Si	Si
SG	Implícito	Si	Si
GVDS	Implícito	Si	No

Tabla 2.4: Caracterización utilizando los criterios sobre el funcionamiento.

[6] Hoppe propone una solución al problema del refinamiento adaptativo pero no es una solución que pueda realizarse en tiempo real

[7] Aunque en [Floriani et al., 1998] los autores dan una solución para codificar de forma comprimida una representación MT, los ratios de compresión que consiguen no son comparables a los que obtienen los modelos que especialmente se han diseñado para ello

[8] El objetivo de una representación PFS es almacenar y transmitir una malla de forma progresiva y comprimida, llegando a conseguir unos ratios de compresión muy altos a costa de que el número de aproximaciones sea muy pequeño

[9] Los autores explican como obtener refinamiento adaptativo, sin embargo no parece que el método permita realizarlo en tiempo real. Además, en este caso es necesario que el método de simplificación actúe contrayendo aristas

[10] Sin embargo, los autores proponen guardar información referente a aproximaciones intermedias y obtener la representación deseada utilizando la más próxima como punto de partida

Capítulo 3

Introducción a Mallas Ordenadas Multirresolución

3.1 Características principales

Mallas Ordenadas Multirresolución (MOM) fue desarrollado con el objetivo de permitir la visualización interactiva de superficies poligonales formadas por cientos de miles de polígonos. En un principio, fue ideado como un modelo que fuera capaz de almacenar y extraer de forma eficiente aproximaciones con nivel de detalle uniforme. Sin embargo, MOM se ha mostrado igual de válido para obtener representaciones con nivel de detalle variable sin necesidad de reorganizar la información y utilizando la misma estructura de datos.

Para conseguir el objetivo fijado con las mayores garantías de éxito, se prestó especial interés en desarrollar una estructura de datos que, sin perder la sencillez que caracteriza a los esquemas básicos de representación de los modelos poligonales, permitiera a los algoritmos de recorrido extraer y visualizar una aproximación en el menor tiempo posible. Así, se optó por llevar a cabo un almacenamiento explícito de la información, evitando tener que realizar operaciones de reconstrucción de los datos pero, al mismo tiempo, asumiendo el incremento del coste de almacenamiento que esto suponía y, en consecuencia, la pérdida de interés del modelo en su aplicación a la compresión y a la transmisión progresiva (aunque esta última es posible [Vidal y Ribelles, 1999], no resulta eficiente). Una vez definida una estructura de datos básica, se realizaron extensiones del modelo con el fin de aumentar aún más sus prestaciones. Estas consistieron en hacer uso de la propiedad de la coherencia cuadro a cuadro, y en utilizar la primitiva gráfica *abanico de triángulos* como primitiva básica de representación.

Un objetivo muy importante en el desarrollo de MOM fue que el modelo debía ser lo más general posible. MOM no haría suposiciones acerca de la geometría del objeto y podría representar, por ejemplo, mallas *non-manifold*. Además, no obligaría a la aplicación a utilizar un método de simplificación concreto en el proceso de construcción de la representación multirresolución y así poder aprovechar el enorme abanico de posibilidades existentes [Heckbert y Garland, 1997]. Por contra, este hecho ha impedido que se dé una solución concreta para realizar una operación de *geomorphing* [Hoppe, 1998a] entre aproximaciones diferentes. De hecho, los modelos que lo permiten es gracias al conocimiento de cómo se han generado las aproximaciones, obligando a utilizar un mecanismo de simplificación concreto.

MOM no es el único modelo multirresolución desarrollado con estos mismos objetivos. En concreto, *Multi-Triangulation* [Floriani et al., 1998] y *Hierarchical Dynamic Simplification* [Luebke y Erikson, 1997] son modelos que se han presentado con el mismo planteamiento y, por este motivo, se utilizarán en posteriores capítulos como modelos de referencia. Además, como se ha comentado anteriormente, MOM fue originalmente ideado para manipular sólo aproximaciones de nivel de detalle uniforme. Es por eso por lo que también se ha utilizado el modelo *Progressive Meshes* [Hoppe, 1996] como referencia. Este modelo tiene especial interés por formar parte del producto *DirectX 5.0* de *Microsoft*.

3.2 Representación incremental

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

$$M^r = \{V^r, T^r\} \tag{3.1}$$

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

Para construir la representación multirresolución M^r con 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:

$$M_{i+1} = (M_i - \{\mathcal{V}_i, \mathcal{T}_i\}) \ \cup \ \{\mathcal{V}'_i, \mathcal{T}'_i\}, \ 0 \leq i < n-1 \tag{3.2}$$

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

$$M^r = \bigcup_{i=0}^{n-1} M_i, n \geq 1 \tag{3.3}$$

De las ecuaciones 3.1 - 3.3, se obtiene que M^r puede ser expresado en función de la malla inicial, $M = M_0$, y de todos los vértices y triángulos generados en cada iteración del proceso de simplificación:

$$V^r = V_0 \cup \mathcal{V}'_0 \cup \mathcal{V}'_1 \cup ... \cup \mathcal{V}'_{n-2} = V_0 \cup \bigcup_{i=0}^{n-2} \mathcal{V}'_i \tag{3.4}$$

$$T^r = T_0 \cup \mathcal{T}'_0 \cup \mathcal{T}'_1 \cup ... \cup \mathcal{T}'_{n-2} = T_0 \cup \bigcup_{i=0}^{n-2} \mathcal{T}'_i \tag{3.5}$$

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

$$V^r = \mathcal{V}_0 \cup \mathcal{V}_1 \cup ... \cup \mathcal{V}_{n-2} \ \cup \ V_{n-1} = \bigcup_{i=0}^{n-2} \mathcal{V}_i \ \cup \ V_{n-1} \tag{3.6}$$

$$T^r = \mathcal{T}_0 \cup \mathcal{T}_1 \cup ... \cup \mathcal{T}_{n-2} \ \cup \ T_{n-1} = \bigcup_{i=0}^{n-2} \mathcal{T}_i \ \cup \ T_{n-1} \tag{3.7}$$

La idea básica de una representación MOM se basa en las expresiones de las dos ecuaciones anteriores, 3.6 y 3.7. 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 malla correspondiente a 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 (figura 3.1) y que por supuesto corresponda con el nivel de detalle requerido. Además de los datos que definen la geometría y la conectividad del objeto (para sus n aproximaciones), es preciso almacenar las relaciones de dependencia que existen entre ellos (ver sección 2.5.3). Almacenar estas relaciones de dependencia puede suponer un importante aumento del coste de almacenamiento [El-Sana y Varshney, 1999]. En la figura 3.2 se muestra un ejemplo de las triangulaciones que pueden existir en un pequeño fragmento de superficie junto con las relaciones de dependencia existentes entre las distintas triangulaciones. La estructura de datos que se corresponde con este esquema sería un grafo acíclico dirigido [Puppo, 1998] donde cada triangulación se correspondería con un nodo del grafo y cada relación de dependencia con un arco.

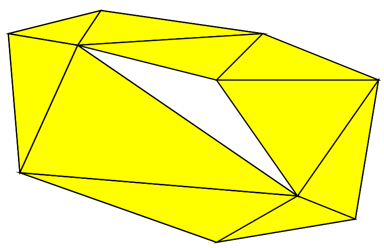


Figura 3.1: Una errónea recuperación de los datos puede producir problemas tales como agujeros.

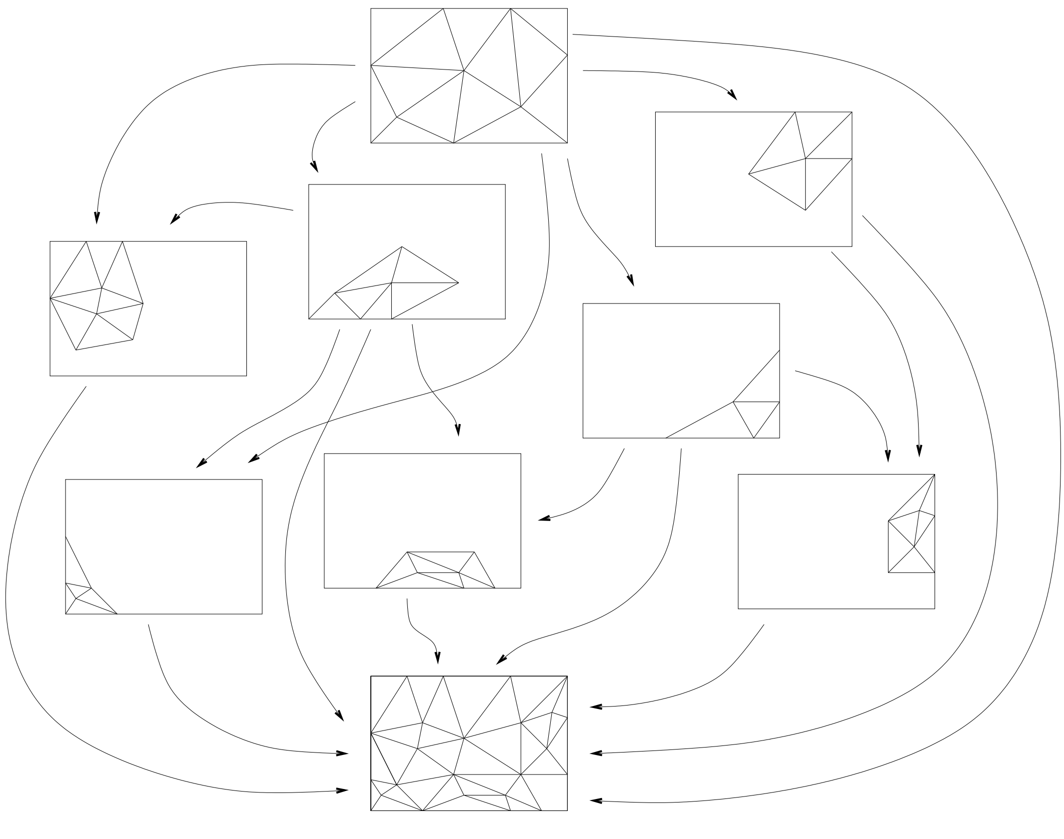


Figura 3.2: Relaciones de dependencia entre las distintas triangulaciones en un pequeño fragmento de superficie.

Mallas Ordenadas Multirresolución 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. El orden existente 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 estará 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 de acuerdo a la ecuación 3.5. Este valor se utilizará, en el proceso de extracción de una aproximación, para conocer si dicho triángulo pertenece o no a la solución.

Consecuentemente, la estructura de datos pasará de ser un grafo acíclico dirigido a una estructura basada en listas. En la figura 3.3 se muestra un esquema de dicha

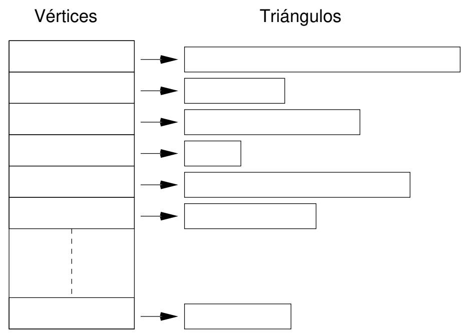


Figura 3.3: Esquema de la estructura de datos basada en listas utilizada por una representación multirresolución MOM. En ella, cada vértice tiene asociada una lista de triángulos.

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.

Capítulo 4

Estructura de Datos Básica

Un objeto MOM se caracteriza por almacenar de forma explícita todos y cada uno de los vértices y triángulos que pueden utilizarse en la representación de cualquier aproximación. La mayor parte de los modelos multirresolución codifican ciertos datos de forma implícita (ver sección 2.6) excepto MT [Floriani et al., 1998] que permite ambas posibilidades. Respecto a la estructura de datos, MOM utiliza una lista de listas. Otras estructuras de datos que se han utilizado para codificar un modelo multirresolución son el grafo acíclico dirigido [Puppo, 1996, Guéziec et al., 1998a], la lista [Hoppe, 1996, Klein y Strasser, 1997], el árbol binario [Xia et al., 1997, Hoppe, 1997, El-Sana y Varshney, 1999] y el árbol en general [Luebke y Erikson, 1997].

En este capítulo se detalla la estructura de datos necesaria para representar un objeto MOM así como su proceso de construcción. También se presenta un análisis teórico del coste de almacenamiento y, finalmente, se muestran los resultados experimentales obtenidos utilizando diversos objetos.

4.1 Estructura de datos

La estructura de datos de un objeto MOM sigue el esquema que se muestra en la figura 4.1, donde todas las listas de triángulos que se pueden observar en la figura 3.3 se almacenan de forma consecutiva en una única lista. En la figura 4.2 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 los datos que pertenecen a la aproximación requerida.

Cada vértice en una representación MOM constará 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 4.2 se explica

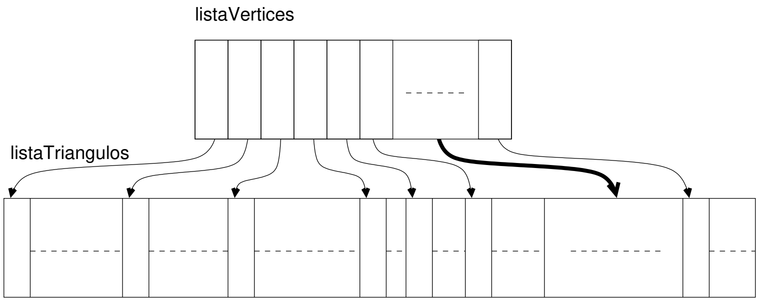


Figura 4.1: Esquema de la estructura de datos básica de una representación MOM (sólo se han dibujado las listas de vértices y triángulos).

```
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 4.2: Estructura de datos básica de una representación MOM.

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 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 3.5.

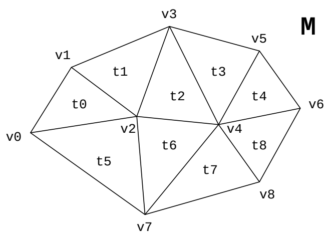
Por último, cada elemento de la lista de control contiene información específica que permite recuperar una determinada aproximación. Estos datos están muy ligados al proceso de construcción de un objeto MOM (explicado en la siguiente sección). En pocas palabras, para cada iteración del proceso de simplificación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

4.2 Proceso de construcción

Para construir una representación MOM son necesarios dos elementos. Por un lado, el objeto, M, del que se quiere obtener una representación multirresolución, M^r, y que estará representado por una lista de vértices (llamada *Vertices*) y una lista de triángulos (llamada *Triangulos*) [Foley et al., 1990]. Por otro lado, el método de simplificación que se va a aplicar sobre el objeto original $M = M_0$, generando las $n-1$ distintas aproximaciones M_1, M_2,...,M_{n-1}. Una representación MOM no imponen ningún tipo de restricción a ninguno de estos dos elementos. Cualquier método de simplificación y cualquier tipo de superficie pueden ser utilizados.

Para facilitar la explicación del proceso de construcción se va a utilizar como ejemplo una malla, M, formada por nueve vértices y nueve triángulos (figura 4.3). Además, el método de simplificación eliminará un solo vértice y será aplicado un total de dos veces. De esta forma, M^r será una representación MOM conteniendo tres aproximaciones.

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. Previamente al inicio de la primera etapa es necesario inicializar M^r con $M = M_0$ ya que esta es justo la primera aproximación. La inicialización afecta únicamente a la lista *listaControlLod*. Tanto la lista *listaVertices* como la lista *listaTriangulos* permanecen



Vertices

v0	v1	v2	v3	v4	v5	v6	v7	v8

Triangulos

v0	v1	v2	v3	v4	v0	v2	v4	v4
v1	v3	v3	v5	v5	v2	v4	v8	v6
v2	v2	v4	v4	v6	v7	v7	v7	v8

Figura 4.3: Ejemplo de malla poligonal y su estructura de datos basada en lista de vértices y lista de triángulos.

vacías. Utilizando el ejemplo, *listaControlLod* se inicializa con cero vértices eliminados y nueve triángulos generados (el número de triángulos de M).

4.2.1 Primera etapa

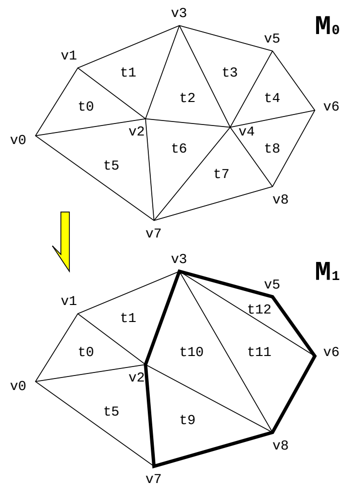
Estas son las tareas a realizar a cada iteración del proceso de simplificación:

1. Añadir a *listaTriangulos* los triángulos eliminados de M y ordenados de menor a mayor según su posición en la lista *Triangulos*.
2. Añadir a *listaVertices* el vértice eliminado de M y hacer que su campo *primerTriangulo* almacene la posición del primer triángulo de los recién añadidos en el punto anterior.
3. Añadir a la lista de control del nivel de detalle, *listaControlLod*, un nuevo elemento formado por el total de vértices eliminados y el total de triángulos generados hasta el momento.

Evidentemente, la malla M también se ve modificada por el método de simplificación incorporando los nuevos vértices y triángulos producidos.

Volviendo al ejemplo, en la figura 4.4 se muestra el resultado de la primera simplificación que consiste en la eliminación del vértice *v4*. Siguiendo los puntos indicados anteriormente las tareas a realizar son:

1. Añadir a *listaTriangulos* *t2, t3, t4, t6, t7* y *t8*. Como se puede observar, cada triángulo está formado por un identificador, que no es más que la posición de él mismo en la lista *Triangulos* de M, y los dos punteros a vértices que siguen apuntando a elementos de la lista de *Vertices* de M.
2. Añadir a *listaVertices* el nuevo vértice que estará formado por *v4* y un cero (la posición en *listaTriangulos* del primer triángulo asociado al vértice).



listaVertices

v4	
0	

listaTriangulos

2	3	4	6	7	8
v2	v3	v5	v7	v8	v6
v3	v5	v6	v2	v7	v8

listaControlLod

0	1
9	13

Figura 4.4: Resultado del proceso de construcción tras la primera simplificación (eliminación de $v4$ en M_0).

3. Añadir a *listaControlLod* que se ha eliminado un vértice y que ahora hay un total de 13 triángulos.

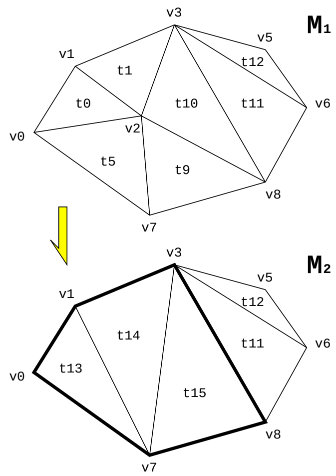
En la figura 4.5 se muestra el resultado de la segunda simplificación que consiste en la eliminación del vértice *v2*, y las tareas a realizar son:

1. Añadir a *listaTriangulos t0, t1, t5, t9* y *t10*.
2. Añadir a *listaVertices* el nuevo vértice formado por *v2* y un seis (la posición en *listaTriangulos* del primer triángulo asociado al vértice).
3. Añadir a *listaControlLod* que se han eliminado un total de dos vértices y que ahora hay un total de 16 triángulos.

4.2.2 Segunda etapa

En este momento, M^r está almacenando desde M_0 hasta M_{n-2} ambas inclusive (M_0 y M_1 en el ejemplo). Las tareas a realizar serán:

1. Añadir M_{n-1} a M^r. Esta última malla se almacena repartiendo los triángulos que lo forman entre sus vértices (con la única condición de que el triángulo utilice dicho vértice), y almacenándolos en sus respectivas listas. Si hay vértices sin triángulos asociados, estos se almacenarán siempre al final de la lista.



listaVertices

v4	v2
0	6

listaTriangulos

2	3	4	6	7	8	0	1	5	9	10
v2	v3	v5	v7	v8	v6	v0	v1	v7	v8	v3
v3	v5	v6	v2	v7	v8	v1	v3	v0	v7	v8

listaControlLod

0	1	2
9	13	16

Figura 4.5: Resultado del proceso de construcción tras una segunda simplificación (eliminación de $v2$ en M_1).

2. Actualizar los dos punteros a vértices que cada triángulo de *listaTriangulos* contiene. Esto es necesario ya que, hasta este momento, dichos punteros apuntan a vértices de la lista *Vertices*. Ahora, todos los vértices se encuentran en la lista *listaVertices* y en posiciones distintas a las que ocupaban en la lista *Vertices*.

Siguiendo con el ejemplo, en la figura 4.6 se muestra el resultado final después de realizar la segunda etapa del proceso de construcción. Las tareas a realizar son:

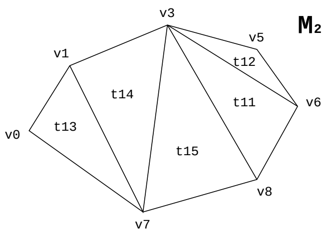
1. Añadir M_2 a M^r. En el ejemplo se han asociado *t13* y *t14* a *v1*, *t12* a *v5* y *t11* y *t15* a *v8*. Después se almacenan *v0, v3, v6* y *v7* como vértices sin triángulos asociados.

2. Actualizar los punteros a vértices en *listaTriangulos*. Así, por ejemplo, el primer triángulo apuntará ahora a *v1* y *v6* en lugar de a *v2* y *v3*.

4.2.3 Generalización del proceso de construcción

Las dos etapas explicadas en los puntos anteriores son válidas si es un sólo vértice el que se elimina en cada iteración del proceso de simplificación. Generalizar el proceso para cuando sean eliminados más de un vértice en una simplificación resulta sencillo y se reduce a resolver las dos siguientes cuestiones:

- ¿Cómo asociar los triángulos a los vértices? La única regla a seguir es que un triángulo se puede asociar a un vértice siempre y cuando el triángulo utilice dicho vértice. Además, por razones de eficiencia ninguno de los vértices se quedará sin triángulos asociados.



listaVertices

v4	v2	v1	v5	v8	v0	v3	v6	v7
0	6	11	13	14	~	~	~	~

listaTriangulos

2	3	4	6	7	8	0	1	5	9	10	13	14	12	11	15
v1	v6	v3	v8	v4	v7	v5	v2	v8	v4	v6	v8	v6	v7	v6	v8
v6	v3	v7	v1	v8	v4	v2	v6	v5	v8	v4	v5	v8	v6	v7	v6

listaControlLod

0	1	2
9	13	16

Figura 4.6: Resultado final del proceso de construcción de una representación MOM.

- ¿En qué orden se deben almacenar los vértices? No importa el orden entre ellos, pero el orden de los vértices y el de los triángulos asociados a cada uno de ellos han de coincidir.

Por ejemplo, en la figura 4.7 se muestra el resultado tras eliminar *v2* y *v4* de M en la misma iteración del proceso de simplificación (en un sólo paso). Se podría asociar *t0, t1* y *t5* a *v2* y *t2, t3, t4, t6, t7* y *t8* a *v4*. A la hora de añadir *v2* y *v4* en *listaVertices* no importaría el orden pero, si se añade *v2* en primer lugar, entonces *t0, t1* y *t5* deben añadirse antes que *t2, t3, t4, t6, t7* y *t8*, y viceversa.

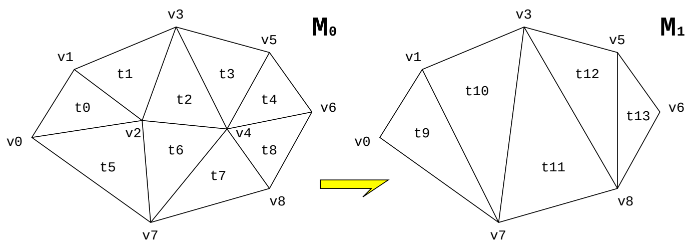


Figura 4.7: Eliminación de dos vértices en una sóla simplificación.

4.2.4 Consideraciones

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.

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.

4.3 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. Recordando la estructura de datos de una representación MOM (figura 4.2), 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 aproximaciones almacenadas en M^r. A continuación se va a calcular el número de vértices, $|V^r|$, de triángulos, $|T^r|$, y de aproximaciones, n, en función del número de vértices, $|V|$, de triángulos, $|T|$, y de aristas, e, de M.

4.3.1 Cálculo de $|V^r|$ y $|T^r|$

4.3.1.1 Peor caso

M^r almacena de forma explícita cada uno de los vértices y triángulos que pueden utilizarse para representar cualquiera de sus n aproximaciones. De esta forma, se puede definir V^r y T^r como:

$$V^r = \bigcup_{i=0}^{n-1} V_i \qquad T^r = \bigcup_{i=0}^{n-1} T_i \tag{4.1}$$

donde V_i y T_i son los conjuntos de vértices y triángulos, respectivamente, de la malla M_i, $0 \leq i \leq n-1$. En el peor caso, los V_i serán conjuntos disjuntos, y lo mismo los T_i:

$$\bigcap_{i=0}^{n-1} V_i = \emptyset \qquad \bigcap_{i=0}^{n-1} T_i = \emptyset \tag{4.2}$$

de modo que:

$$|V^r| = \sum_{i=0}^{n-1} |V_i| \qquad |T^r| = \sum_{i=0}^{n-1} |T_i| \tag{4.3}$$

donde $|V_i|$ y $|T_i|$ son el número de vértices y triángulos, respectivamente, de la malla M_i $0 \leq i \leq n-1$. Además, ya que cualquier método de simplificación siempre reduce el número de vértices y triángulos al menos en uno, se obtiene que:

$$|V_i| < |V_j| \quad \longleftrightarrow i > j \tag{4.4}$$

$$|T_i| < |T_j| \quad \longleftrightarrow i > j \tag{4.5}$$

con lo que se obtiene que:

$$|V_0| = |V|,\ |V_1| \leq |V|-1,\ |V_2| \leq |V|-2,\ ...,\ |V_i| \leq |V|-i \tag{4.6}$$

$$|T_0| = |T|,\ |T_1| \leq |T|-1,\ |T_2| \leq |T|-2,\ ...,\ |T_i| \leq |T|-i \tag{4.7}$$

y por lo tanto:

$$|V^r| \leq \sum_{i=0}^{n-1} |V|-i \qquad |T^r| \leq \sum_{i=0}^{n-1} |T|-i \tag{4.8}$$

Para el cálculo de $|V^r|$, en el peor caso n estará acotado por $|V|$:

$$|V^r| \leq \sum_{i=0}^{|V|-1} |V|-i = \sum_{i=0}^{|V|-1} |V| - \sum_{i=0}^{|V|-1} i = |V|^2 - |V|(|V|+1)/2 = |V|(|V|-1)/2 \tag{4.9}$$

Obteniendo que $|V^r|$ es $O(|V|^2)$. Para el cálculo de $|T^r|$, en el peor caso n estará acotado por $|T|$:

$$|T^r| \leq \sum_{i=0}^{|T|-1} |T|-i = \sum_{i=0}^{|T|-1} |T| - \sum_{i=0}^{|T|-1} i = |T|^2 - |T|(|T|+1)/2 = |T|(|T|-1)/2 \tag{4.10}$$

Obteniendo que $|T^r|$ será $O(|T|^2)$.

4.3.1.2 Caso medio

Por término medio existen 6 triángulos alrededor de cada vértice de la malla [Hoppe, 1999], y se cumple que $|V| \simeq |T|/2$ [Hoppe, 1997]. Supóngase que esta media se cumple para cada aproximación almacenada en la representación multirresolución:

$$|V_i| \simeq \frac{|T_i|}{2}, \quad 0 \leq i < n-1 \tag{4.11}$$

A partir de la ecuación 4.11 y de la fórmula de Euler para poliedros se obtiene que:

$$e_i \simeq 3\ |V_i|, \quad 0 \leq i < n-1 \tag{4.12}$$

donde e_i es el número de aristas de la malla M_i, $0 \leq i \leq n-1$. Ahora es necesario cuantificar lo que supone una simplificación, es decir, cuantos vértices y triángulos son eliminados y cuantos vértices y triángulos son añadidos por simplificación realizada. Estos valores dependen del método de simplificación empleado. Por este motivo se presentan dos análisis correspondientes al uso de un método basado en eliminación de vértices y otro basado en contracción de aristas.

Simplificación por eliminación de vértices

Este tipo de métodos [Schroeder et al., 1992] se basan en la eliminación de un vértice de la malla por cada simplificación (figura 4.8). El agujero que se produce por la eliminación de los triángulos que utilizan el vértice eliminado se triangula. En consecuencia, aparecen triángulos nuevos pero nunca vértices nuevos. De esta forma $|V^r| = |V|$.

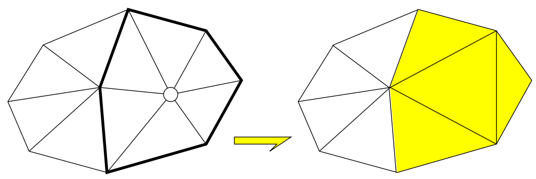


Figura 4.8: Simplificación por eliminación de un vértice. Al eliminar un vértice, desaparecen los triángulos que lo rodean y el agujero producido se triangula.

Respecto al número de triángulos, $|T^r|$, si de media hay 6 triángulos alrededor de cada vértice, son 4 los que harán falta para triangular el agujero, obteniendo un total de $|T| + 4n$, es decir, el número de triángulos de la malla original más 4 por aproximación. Como n está acotado por el número de vértices de M_0, se obtiene que $|T^r| \leq |T| + 4|V|$, y por la ecuación 4.11 se obtiene que $|T^r| \leq 3|T|$.

Simplificación por contracción de aristas

Este tipo de métodos [Algorri y Schmitt, 1996] se basan en la contracción de una arista de la malla por cada simplificación (figura 4.9). El nuevo punto puede coincidir con uno de los vértices de la arista o realmente ser un punto nuevo. En el mejor caso, el punto nuevo siempre coincide con uno de los vértices de la arista y el análisis del coste coincide con el presentado cuando se utiliza un método de eliminación de vértices. En el peor caso, el punto nuevo nunca coincide con uno de los extremos de la arista y esta situación es la que se va a analizar.

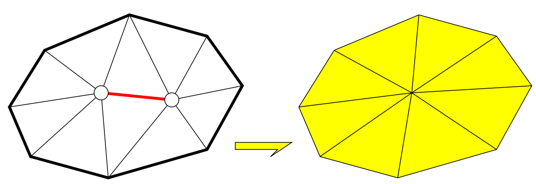


Figura 4.9: Simplificación por contracción de una arista. Al contraer la arista, desaparecen los triángulos que utilizan alguno de sus vértices y el agujero producido se triangula.

Respecto al número de vértices, por cada arista se eliminan siempre dos vértices y aparece uno nuevo. Así, se obtendría que el número de vértices total es de $|V|$ más uno por contracción. Por la ecuación 4.12, y como por cada contracción se eliminan generalmente 3 aristas (si la arista pertenece a un borde son sólo dos), se obtiene que el número total de contracciones posibles es de $|V|$ y, en consecuencia, $|V^r| \leq 2|V|$.

Respecto al número de triángulos, alrededor de cada arista hay 10 triángulos por término medio, y 8 son añadidos para triangular el agujero. De esta forma, el coste es de $|T| + 8n$, es decir, del número de triángulos de la malla original más 8 por aproximación. Como n está acotado por el número de contracciones, $|V|$, utilizando la ecuación 4.11, se obtiene que $|T^r| \leq 5|T|$.

Resumen

En la tabla 4.1 se resumen los costes de almacenamiento teóricos obtenidos para el caso medio. El coste de almacenamiento del modelo original es $3|V| + 3|T|$ si se representa mediante una lista de vértices y una lista de triángulos. Utilizando la ecuación 4.11, se obtiene que la representación multirresolución es 2'67 veces mayor si se utiliza un método de simplificación basado en eliminación de vértices en el proceso de construcción, y entre 2'67 y 4'44 veces mayor si se utiliza un método de simplificación basado en contracción de aristas.

Coste: $4\|V^r\| + 3\|T^r\| + 2n$	$\|V^r\|$	$\|T^r\|$	n	Total	Ratio
Eliminación de vértices	$\|V\|$	$3\|T\|$	$\|V\|$	$6\|V\| + 9\|T\|$	2'67
Contracción de aristas (mejor caso)	$\|V\|$	$3\|T\|$	$\|V\|$	$6\|V\| + 9\|T\|$	2'67
Contracción de aristas (peor caso)	$2\|V\|$	$5\|T\|$	$\|V\|$	$10\|V\| + 15\|T\|$	4'44

Tabla 4.1: Costes teóricos de almacenamiento en el caso medio.

4.3.2 Cálculo de n

El número máximo de aproximaciones depende del tipo de simplificación que se realice. Si el método de simplificación utilizado elimina vértices, el número de aproximaciones estará acotado por el número de vértices de M. Si el método de simplificación utilizado contrae aristas, el número de aproximaciones estará acotado por el número de aristas de M (aunque, en general, la contracción de una arista produce la eliminación de otras dos). Y si el método de simplificación utilizado contrae triángulos, el número de aproximaciones estará acotado por el número de triángulos de M. Se puede concluir que n estará acotado, independientemente del método de simplificación utilizado, por el máximo de entre el número de vértices, triángulos y aristas de M. En cualquier caso, n será siempre lineal respecto a $|V|, |T|$, o e, y relativamente pequeño comparado con los costes obtenidos para las listas de vértices y triángulos.

4.4 Resultados

Los experimentos han sido realizados utilizando:

Hardware. Silicon Graphics RealityEngine 2, con un MIPS R10000 a 194 MHz y 256 Mb RAM.

Software. La codificación de los modelos multirresolución ha sido realizada en *C++* y se ha utilizado *OpenGL* como librería gráfica. Para crear los distintos objetos multirresolución se ha utilizado el método de simplificación propuesto por Garland y Heckbert [Garland y Heckbert, 1997] basado en contracción de aristas obligando a que el punto de contracción coincida con uno de los dos vértices de la arista.

Datos. Las mallas provienen de *Stanford University Computer Graphics Laboratory* (http://www-graphics.stanford.edu/data/3Dscanrep/) y *Cyberware* (http://www.cyberware.com/models/).

En la tabla 4.2 se resumen las características y los costes de almacenamiento de los objetos utilizados en los experimentos (ver figuras 4.10 y 4.11). Para cada uno de ellos se indica el número de vértices y triángulos del modelo original y su coste de almacenamiento (en megabytes) asumiendo una estructura de datos basada en una lista de vértices y una lista de triángulos. También se asume 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 4.1).

En la tabla 4.3 se muestra una comparación de los costes de almacenamiento para las representaciones multirresolución MOM, *Progressive Meshes* (PM) [Hoppe, 1998a] y *Multi-Triangulation* (MT) [Floriani et al., 1998]. Para hacer esta comparación se han utilizado las representaciones propuestas por los autores diseñadas a propósito para visualización interactiva.

Los resultados muestran que la representación multirresolución PM experimenta un aumento en el coste de almacenamiento respecto al de los objetos originales. 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 [Hoppe, 1997], versión de PM que es capaz de extraer aproximaciones con nivel de detalle variable, necesita de, por

	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
Knee	37.888	75.264	1'295	34.373	223.561	3'422	2'64
Phone	83.045	165.963	2'850	81.668	441.181	6'940	2'44
Teeth	116.660	233.316	4'005	116.388	681.534	10'468	2'62
Venus	134.359	268.714	4'613	134.300	786.962	12'081	2'62
Balljoint	137.073	274.144	4'706	136.971	805.211	12'351	2'63
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
Dragon	437.645	869.928	14'964	430.180	2.379.139	37'187	2'49
Hip	530.168	1.060.346	18'202	529.312	3.119.354	47'826	2'63
Buda	543.653	1.085.636	18'646	543.106	2.754.083	43'957	2'36

Tabla 4.2: Lista de objetos con sus características y costes de almacenamiento.

	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

Tabla 4.3: Costes de almacenamiento para las representaciones multirresolución MOM, PM y MT (en megabytes).

ejemplo, 7'8MB para el objeto *Bunny*, coste bastante superior al de una representación MOM o PM.

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.

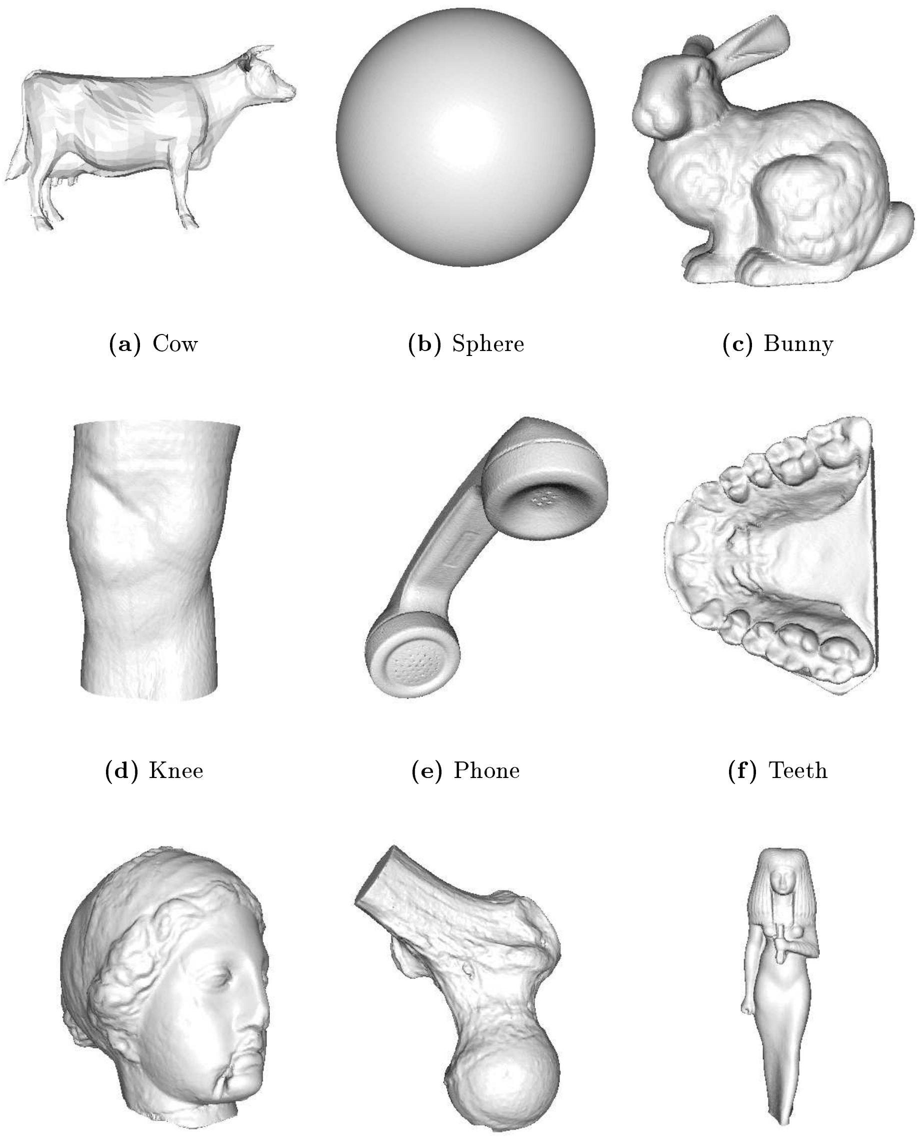
Hierarchical Dynamic Simplification [Luebke y Erikson, 1997] no ha sido incluido en la tabla porque los autores no han presentado un análisis del coste de almacenamiento de su modelo. Sin embargo, sí que han publicado algunos datos concretos como, por ejemplo, que el modelo *Bunny* necesita alrededor de cinco megabytes comprimidos utilizando el compresor de ficheros *gzip*. Este coste es muy alto comparado con cualquiera de los otros tres modelos.

Por último, señalar que en el capítulo 7 se presentará una extensión de MOM que, utilizando abanicos de triángulos, permite rebajar aún más los costes de almacenamiento (alrededor de un 15%) sin que ello suponga una disminución en las prestaciones del modelo.

4.5 Conclusiones

Los resultados experimentales muestran como las representaciones multirresolución MOM tienen un coste de almacenamiento menor que los modelos multirresolución presentados con características similares (MT y HDS). Además, el coste de almacenamiento es comparable a PM, que siendo un modelo válido únicamente para representar aproximaciones con nivel de detalle uniforme, impone una serie de condiciones para poder ser aplicado como que la malla sea *manifold* y que el método de simplificación utilizado en el proceso de construcción se base en contracción de aristas.

En resumen, la estructura de datos propuesta se caracteriza por ser simple y compacta, facilitando el desarrollo de algoritmos de recorrido eficientes, así como la propia construcción de objetos MOM.



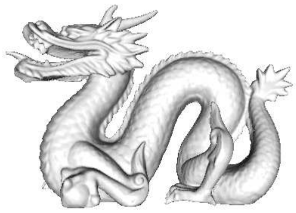
(a) Cow **(b)** Sphere **(c)** Bunny

(d) Knee **(e)** Phone **(f)** Teeth

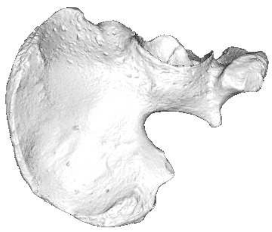
(g) Venus **(h)** Balljoint **(i)** Isis

Figura 4.10: Imágenes de algunos de los objetos utilizados en los experimentos.

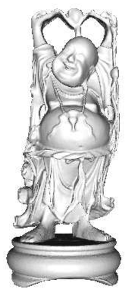
(a) Steve



(b) Dragon



(c) Hip



(d) Buda

Figura 4.11: Más imágenes de algunos de los objetos utilizados en los experimentos.

Capítulo 5

Nivel de Detalle Uniforme

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 [Hoppe, 1996, Luebke y Erikson, 1997, Xia et al., 1997, Puppo, 1998], o bien partiendo de la aproximación de mayor nivel de detalle y simplificando [Lindstrom et al., 1996]. El algoritmo que se presenta en este capítulo, gracias a la ordenación de los datos y a su almacenamiento de forma explícita, no necesita partir ni de la aproximación de mayor detalle ni de la de menor detalle. Directamente es capaz de acceder a los datos que forman la aproximación requerida sin necesidad de ejecutar una secuencia de simplificaciones o refinamientos.

En este capítulo se presenta el algoritmo que recorre la estructura de datos propuesta en el capítulo 4 y extrae los triángulos que pertenecen a una aproximación de un nivel de detalle dado. Se analiza el coste temporal teórico y se muestran los resultados experimentales obtenidos utilizando diferentes objetos. Para realizar los experimentos se ha definido un conjunto de tests que intentan medir las prestaciones de un modelo multirresolución cuando es utilizado para visualización interactiva y nivel de detalle uniforme y así poder comparar los resultados obtenidos con otras representaciones multirresolución. En concreto, se utilizan este conjunto de tests para realizar una comparación con la representación *Progressive Meshes*.

5.1 Algoritmo de visualización

Descripción

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.

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 5.1 se muestran las operaciones correspondientes al cálculo del número de triángulos.

```
// Dada una aprox., M_L, obtener de la lista de control sus datos
TG= listaControlLod[L].triangulosGenerados;
VE= listaControlLod[L].verticesEliminados;

// NT es el número de triángulos de M_L
NT= TG - listaVertices[VE].primerTriangulo;
```

Figura 5.1: Cálculo del número de triángulos que pertenecen a una aproximación dada de nivel de detalle uniforme.

Visualización de los triángulos

Ahora que ya se conoce el número de triángulos a visualizar, queda por determinar a partir de que vértice comenzar a recorrer la estructura de datos y que triángulos pintar de entre los asociados a cada vértice.

Cómo saber a partir de que vértice comenzar ya quedó entredicho en el punto anterior. Si *VE* es el número de vértices eliminados, se comenzará justo por el vértice número *VE*.

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 5.2 se muestra el algoritmo que realiza las tareas descritas y en la figura 5.3 se muestran ocho aproximaciones distintas de un objeto MOM (donde el nivel de detalle 0 es la aproximación de menor detalle y el 1 la de mayor).

```
// Dada una aprox., M_L, obtener de la lista de control sus datos
TG= listaControlLod[L].triangulosGenerados;
VE= listaControlLod[L].verticesEliminados;

// NT es el número de triángulos de M_L
NT= TG - listaVertices[VE].primerTriangulo;

// El primer vértice será el v_i
i= VE;

primero= listaVertices[i].primerTriangulo;
mientras NT > 0 hacer
  // Recorrer la lista de triángulos asociados a v_i
  ultimo= listaVertices[i+1].primerTriangulo;
  pv_i= direccion (listaVertices[i]);
  mientras primero < ultimo y listaTriangulos[primero].id < TG hacer
    pintaTriangulo (pv_i,
                    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 5.2: Algoritmo de visualización de una aproximación con nivel de detalle uniforme.

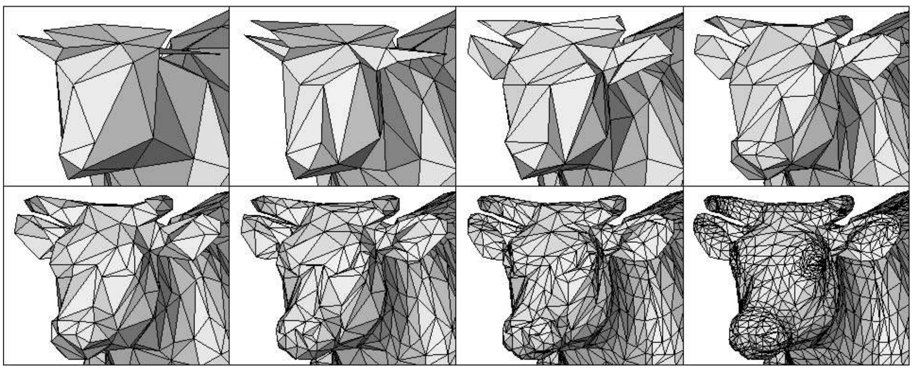


Figura 5.3: De izquierda a derecha y de arriba a abajo se muestran ocho aproximaciones del objeto *Cow* correspondientes a los siguientes niveles de detalle: 0, 0'03, 0'07, 0'14, 0'21, 0'38, 0'64 y 1.

Ejemplo

Recordando el ejemplo utilizado en el capítulo 4 (ver figura 5.4), los resultados de aplicar el algoritmo para la visualización de cada una de las tres aproximaciones se muestran en la tabla 5.1. Para cada malla, se puede observar el número de vértices eliminados, variable VE, el número de triángulos generados , variable TG, y el número de triángulos de la malla, variable NT. Además, se indican los vértices que el algoritmo recorre y para cada uno de ellos qué triángulos, de entre sus asociados, son visualizados.

listaVertices

v4	v2	v1	v5	v8	v0	v3	v6	v7
0	6	11	13	14	~	~	~	~

listaTriangulos

2	3	4	6	7	8	0	1	5	9	10	13	14	12	11	15
v1	v6	v3	v8	v4	v7	v5	v2	v8	v4	v6	v8	v6	v7	v6	v8
v6	v3	v7	v1	v8	v4	v2	v6	v5	v8	v4	v5	v8	v6	v7	v6

listaControlLod

0	1	2
9	13	16

Figura 5.4: Ejemplo de la estructura de datos de un objeto MOM.

M_0 VE= 0 TG= 9 NT= 9	M_1 VE= 1 TG= 13 NT= 7	M_2 VE= 2 TG= 16 NT= 5
v4 → t2, t3, t4, t6, t7, t8	v2 → t0, t1, t5, t9, t10	v1 → t13, t14
v2 → t0, t1, t5	v1 → ninguno	v5 → t12
	v5 → t12	v8 → t11, t15
	v8 → t11	

Tabla 5.1: Ejemplo del algoritmo de visualización de una aproximación con nivel de detalle uniforme utilizando el objeto MOM de la figura 5.4.

Consideraciones

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.

5.2 Coste temporal

A grandes rasgos, el funcionamiento del algoritmo (figura 5.2), es el siguiente: dada una aproximación, M_L, se calcula el número 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.

El algoritmo obtiene el vértice i a partir del cual comenzar a recorrer de forma secuencial la lista de vértices. Respecto al bucle externo, en el peor caso, el algoritmo visita lo que queda de lista de vértices, es decir, $|\mathcal{V}^r| - i$. Respecto al bucle interno, el algoritmo siempre visualiza las caras correspondientes a M_L, es decir, $|T_L|$. Sumando ambos valores se obtiene $|\mathcal{V}^r| - i + |T_L|$. Procediendo de forma análoga al cálculo de $|\mathcal{V}^r|$ (ver sección 4.3.1.1), se puede deducir que 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 $O(|V_L| + |T_L|)$. Los objetos utilizados en la sección de resultados cumplen que $|V| \approx |T|/2$ y se puede observar que se obtiene un comportamiento lineal respecto al número de triángulos de la aproximación requerida.

5.3 Resultados

Para poder evaluar el comportamiento temporal del modelo se han definido un conjunto de tests. Cada test genera una secuencia de peticiones de visualización de aproximaciones de diferente nivel de detalle y la diferencia entre ellos es la forma en que se genera dicha secuencia. El nivel de detalle vendrá dado por un valor comprendido en el rango [0..1] donde 0 representa el menor detalle y 1 el mayor. En primer lugar se explicarán los tests y a continuación se muestran los resultados obtenidos.

5.3.1 Test aleatorio

Este test genera una secuencia aleatoria de niveles de detalle. El objetivo es medir la velocidad de respuesta del modelo cuando se le solicitan aproximaciones cuyos niveles de detalle no guardan ningún tipo de correlación entre ellos, permitiendo cambios bruscos del nivel de detalle entre peticiones consecutivas. El siguiente código genera la secuencia aleatoria:

```
srand (seed);
para i= 1 hasta m_r hacer
  LOD[i] = f(rand());
fin para
```

donde m_r es la longitud de la secuencia, *seed* es una constante entera positiva y f es una función que normaliza el número aleatorio obligando a que esté en el rango [0..1]. Este código genera siempre la misma secuencia aleatoria para un valor de *seed* constante. Así, es posible usar el test en las mismas condiciones con diferentes modelos. Para hacer que el resultado del test sea independiente de la longitud de la secuencia, se considerará el valor medio del tiempo medido en los experimentos, $\bar{t}$, calculado como el tiempo total, t_r, dividido por el número de aproximaciones solicitadas, m_r:

$$\bar{t} = \frac{t_r}{m_r}$$

5.3.2 Test lineal

El test lineal genera una secuencia de forma que las aproximaciones solicitadas consecutivamente están correlacionadas linealmente. La diferencia entre aproximaciones consecutivas será un valor constante y proporcional al número total de aproximaciones.

Se deben tener en cuenta dos características de este test: el orden de la secuencia (incremental o decremental) y el tamaño del paso o distancia, es decir, cuántas aproximaciones hay entre dos solicitadas consecutivamente. El siguiente código genera una secuencia lineal incremental de paso k:

```
m_l = n/k;
para i= 1 hasta m_l hacer
  LOD[i] = f(i*k);
fin para
```

donde f es la función que normaliza el resultado obligando a que esté en el rango [0..1]. El orden de la secuencia puede conducir a diferentes resultados, es decir, el comportamiento del modelo puede no ser el mismo cuando añaden detalle que cuando lo eliminan. De esta forma, este test se desdobla en dos: el test con orden incremental y el test con orden decremental. El resultado final será el valor medio de ambos tests.

El valor del paso es también significativo ya que algunos modelos obtienen la nueva aproximación a partir de la última aproximación requerida, y se propone utilizar un conjunto de pasos calculados como un porcentaje del número de aproximaciones que almacena el modelo:

$$k = 0.1\%n, 0.2\%n, 0.5\%n, 1\%n, 2\%n, 5\%n$$

para obtener secuencias de longitudes m_l = *1000, 500, 200, 100, 50* y *20*, respectivamente.

5.3.3 Test exponencial

El test exponencial genera una secuencia de forma que las aproximaciones solicitadas consecutivamente están correlacionadas exponencialmente. La curvatura de la exponencial será diseñada de tal forma que el paso entre aproximaciones de bajo detalle sea pequeño y aumente a medida que también lo haga el detalle. Análogamente al test lineal, se propone desdoblar el test en dos, incremental y decremental, y utilizar el conjunto de valores m_e= *1000, 500, 200, 100, 50* y *20*, para chequear los modelos bajo diferentes condiciones. El siguiente código genera una secuencia exponencial incremental de longitud m_e (figura 5.5):

$x = 0$;
para i= 1 **hasta** m_e **hacer**
  LOD[i] = $\frac{(1+C)^x-1}{C}$;
  x = x + $\frac{1}{m_e-1}$;
fin para

donde C tendrá un valor de 20 para que aproximadamente el 80% de las aproximaciones solicitadas estén en el conjunto de las $n/2$ de menor detalle y el 20% estén en el conjunto de las $n/2$ de mayor detalle.

5.3.4 Resultados

Los experimentos han sido realizados utilizando el entorno de trabajo descrito en la sección 4.4. Estos experimentos han consistido en medir tiempos de extracción. El tiempo de extracción es el tiempo que invierte el algoritmo de visualización en obtener los datos que forman la aproximación requerida. No se incluye el tiempo que el sistema gráfico utiliza en visualizar estos datos. En la tabla 5.2 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 5.6 se muestra el comportamiento del algoritmo de visualización para estos cuatro objetos. El eje X representa el nivel de detalle en el rango [0..1], donde 0 es el nivel de menor detalle y 1 el de mayor, y el eje Y representa el tiempo de extracción en milisegundos. Se puede observar que el algoritmo tiene un

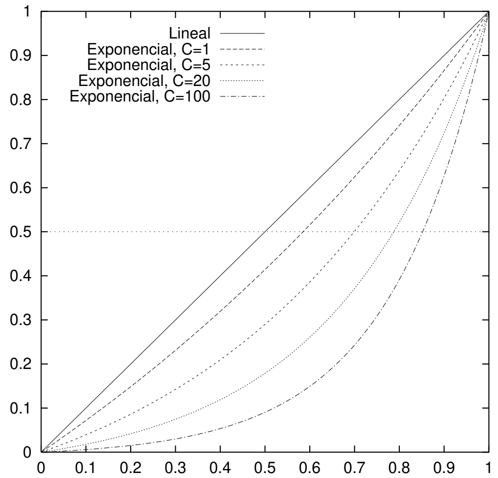


Figura 5.5: Secuencia exponencial en el rango [0..1].

comportamiento lineal respecto al nivel de detalle de la aproximación requerida. Además, en la figura 5.7 se muestra que la relación entre el número de triángulos (eje X) y el tiempo de extracción (eje Y) para cada aproximación también es lineal, coincidiendo con el resultado obtenido en el análisis teórico.

	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

Tabla 5.2: Tiempos de extracción (ms.) para diferentes aproximaciones de cuatro objetos MOM.

	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

Tabla 5.3: Tiempos de extracción (ms.) de la representación MOM del modelo *Bunny* y de la representación MT del modelo *Marcy*.

En la tabla 5.3 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

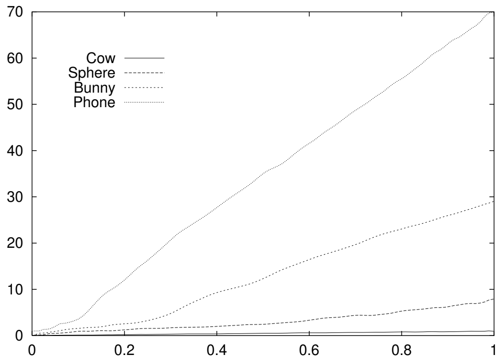
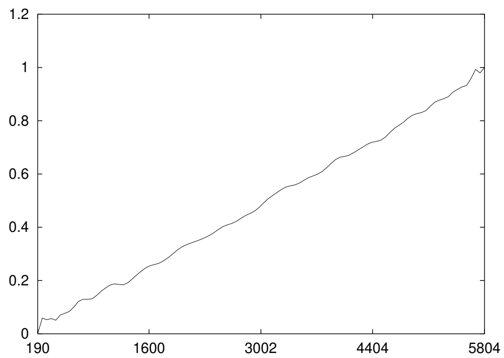
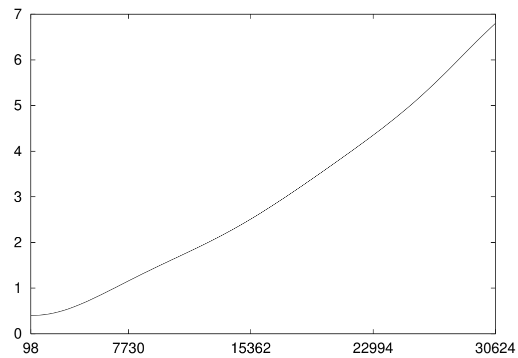


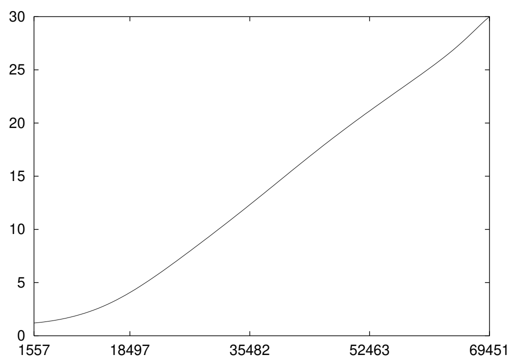
Figura 5.6: Relación entre el nivel de detalle y el tiempo de extracción (ms.) para cuatro objetos MOM.



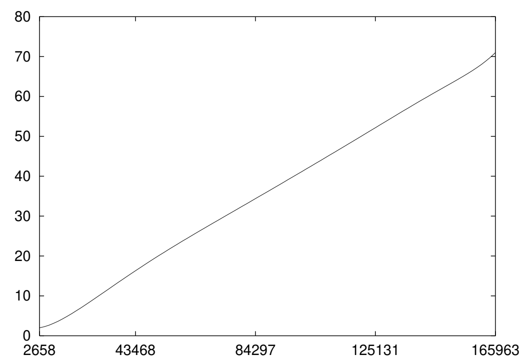
(a) Cow



(b) Sphere



(c) Bunny



(d) Phone

Figura 5.7: Relación entre el número de triángulos extraído y el tiempo de extracción (ms.) para cuatro objetos MOM.

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 comparar extensamente este modelo y MOM. La implementación de PM ha sido optimizada hasta conseguir las prestaciones publicadas por el autor [Hoppe, 1998a]. Los experimentos han consistido en medir los tiempos de extracción de diferentes secuencias de niveles de detalle, secuencias determinadas por los tests aleatorio, lineal y exponencial.

Las figuras 5.8, 5.9, 5.10 y 5.11 muestran los resultados obtenidos para los objetos *Cow, Sphere, Bunny* y *Phone*, respectivamente. Una descripción de las características de estos objetos se encuentra en la tabla 4.2. Cada figura consiste en:

(a) Imagen del objeto.

(b) Tabla que muestra los tiempos de extracción medios (en milisegundos) para cada modelo implementado (MOM y PM), para diferentes longitudes de secuencias de aproximaciones (20, 50, 100, 200, 500 y 1000) y diferentes tipos de secuencias (aleatoria, lineal y exponencial).

(c) Test aleatorio. El eje de abscisas representa el nivel de detalle en el rango [0..1], donde 0 es el nivel de menor detalle y 1 el de mayor. El eje de ordenadas representa el tiempo de extracción (ms.). Para cada objeto y para cada modelo sólo se muestran los resultados para las secuencias de 20 y 100 aproximaciones, ya que las demás secuencias presentan el mismo comportamiento.

(d) Test lineal. Los ejes representan lo mismo que en (c). Para cada objeto MOM, el comportamiento obtenido es el mismo para las distintas secuencias de aproximaciones y sólo se representa una única curva, mientras que para PM se representan todas ellas.

(e) Test exponencial. Los ejes representan lo mismo que en (c). Al igual que en el test lineal, para cada objeto MOM el comportamiento obtenido es el mismo para las distintas secuencias de aproximaciones y sólo se representa una única curva, mientras que para PM se representan todas ellas.

(f) Esta gráfica representa el tiempo de extracción cuando la aproximación requerida es la $M_{n/2}$ y la última solicitada fue cualquier otra. El objetivo es mostrar cómo influye en el tiempo de extracción de una aproximación la distancia que hay entre ésta y la última solicitada. Los ejes representan lo mismo que en (c).

El test aleatorio muestra que MOM es más eficiente que PM cuando aparecen cambios bruscos en la secuencia de peticiones. Este mejor comportamiento se mantiene tanto al cambiar la complejidad de la malla como al variar la longitud de la secuencia de las aproximaciones.

El test lineal evidencia cómo PM mejora su comportamiento al disminuir la distancia entre dos aproximaciones solicitadas consecutivamente, y cómo MOM es independiente de dicha distancia y presenta un comportamiento constante respecto a la longitud de la secuencia. De hecho, en el caso de MOM, los tiempos medios obtenidos con el test lineal son similares a los obtenidos con el test aleatorio y, en el caso de PM, los tiempos medios obtenidos con el test lineal sí que son bastante mejores. Por último, PM consigue mejorar el comportamiento de MOM en las secuencias largas, es decir, sólo cuando el paso es pequeño.

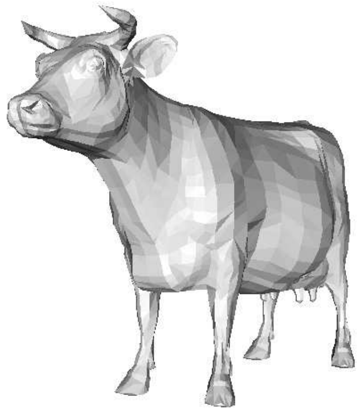
Comparando los resultados obtenidos con los tests lineal y exponencial, se puede observar que los tiempos medios de extracción se han reducido considerablemente, tanto para PM como para MOM, debido a que la mayor parte de las aproximaciones solicitadas se corresponden con las de menor detalle. En el caso de MOM, los resultados del test exponencial llegan a ser la mitad de los obtenidos en el test lineal. En el caso de PM, la mejora es más importante a medida que la distancia entre aproximaciones consecutivas disminuye. De nuevo, PM consigue ser más eficiente que MOM sólo cuando los pasos son pequeños.

Por último, los resultados que se muestran en las gráficas (f) de todos los objetos, evidencian la dependencia de las prestaciones de PM respecto de la distancia que existe entre las aproximaciones solicitadas consecutivamente, y cómo MOM se muestra indiferente respecto a este hecho.

5.4 Conclusiones

En este capítulo se ha presentado un algoritmo que permite extraer aproximaciones de nivel de detalle uniforme recorriendo la estructura de datos presentada en el capítulo anterior.

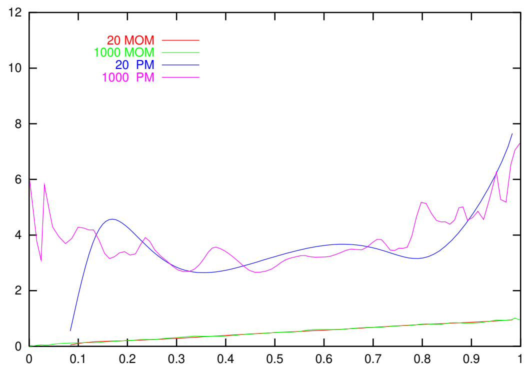
Aunque no se ha podido comparar de forma extensa con MT y HDS, los pocos datos existentes muestran que MOM es mucho más eficiente. Sin embargo, sí que se ha podido realizar una amplia comparación con el modelo PM. El resultado de esta comparación muestra la influencia que produce en las prestaciones de ambos 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 que MOM cuando la diferencia en el nivel de detalle de las dos aproximaciones es muy pequeña.



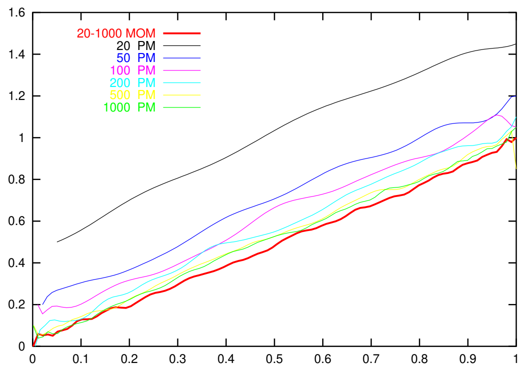
(a) Cow.

#LODs	Aleatorio		Lineal		Exponencial	
	MOM	PM	MOM	PM	MOM	PM
20	0.5	3.5	0.5	1.0	0.3	0.9
50	0.5	3.5	0.5	0.7	0.3	0.5
100	0.5	3.8	0.5	0.6	0.3	0.4
200	0.5	3.7	0.5	0.6	0.3	0.4
500	0.5	3.7	0.5	0.5	0.3	0.3
1000	0.5	3.8	0.5	0.5	0.3	0.3

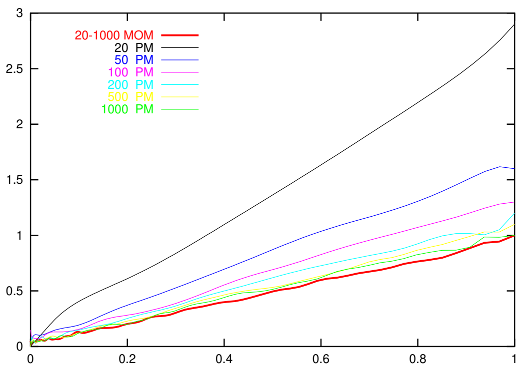
(b) Tiempos medios (ms.).



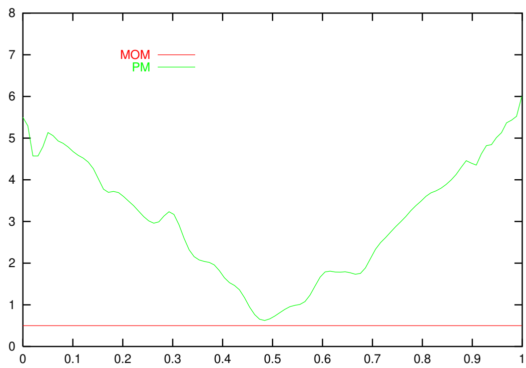
(c) Test aleatorio.



(d) Test lineal.

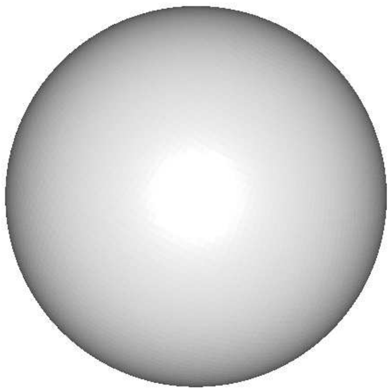


(e) Test exponencial.



(f) Lod recuperado: 1403.

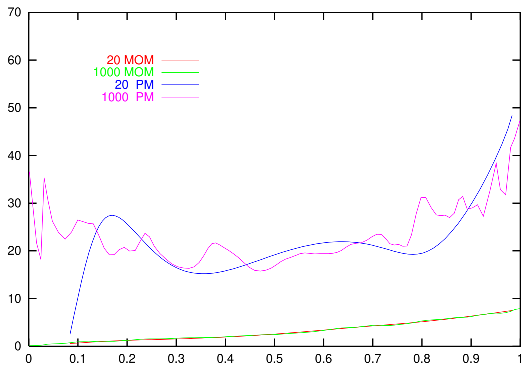
Figura 5.8: Resultados obtenidos para el modelo *Cow*.



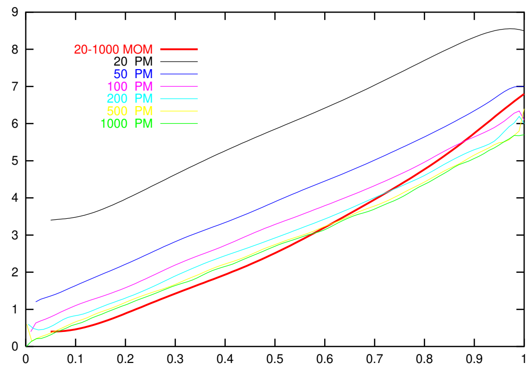
(a) Sphere.

#LODs	Aleatorio		Lineal		Exponencial	
	MOM	PM	MOM	PM	MOM	PM
20	2.9	21.2	3.0	6.0	1.7	5.1
50	3.4	21.0	2.8	4.0	1.5	2.9
100	3.2	22.9	2.8	3.3	1.5	2.2
200	3.1	22.5	2.8	3.0	1.5	1.8
500	3.0	22.3	2.8	2.8	1.5	1.6
1000	3.2	23.1	2.8	2.8	1.5	1.6

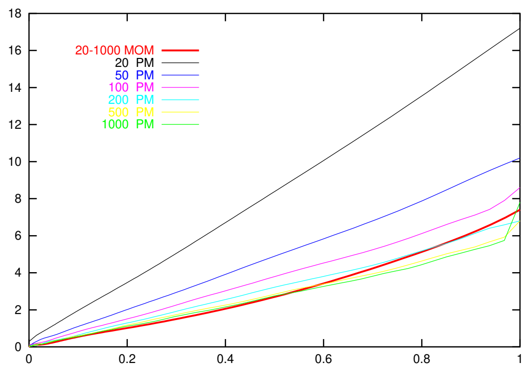
(b) Tiempos medios (ms.).



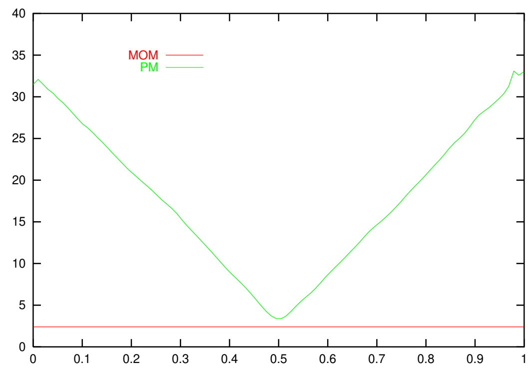
(c) Test aleatorio.



(d) Test lineal.

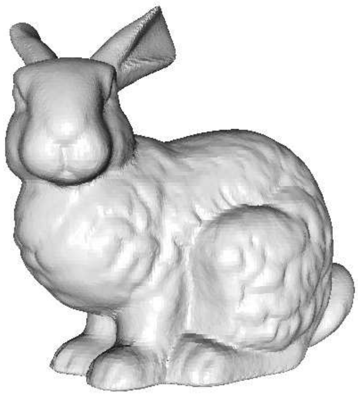


(e) Test exponencial.



(f) Lod recuperado: 7632.

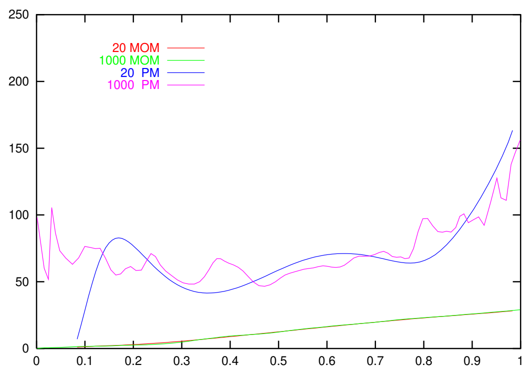
Figura 5.9: Resultados obtenidos para el modelo *Sphere*.



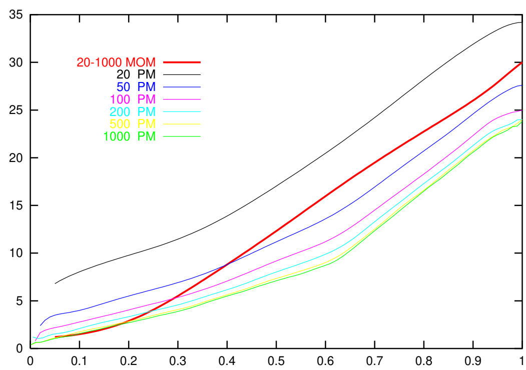
(a) Bunny.

#LODs	Aleatorio		Lineal		Exponencial	
	MOM	PM	MOM	PM	MOM	PM
20	12.5	66.1	13.7	18.9	7.0	15.8
50	14.5	67.2	13.2	12.9	6.8	8.7
100	13.4	71.4	13.1	10.9	6.7	6.6
200	13.1	69.5	13.0	9.9	6.6	5.6
500	12.6	68.5	13.0	9.3	6.6	5.0
1000	13.3	71.5	13.0	9.1	6.6	4.8

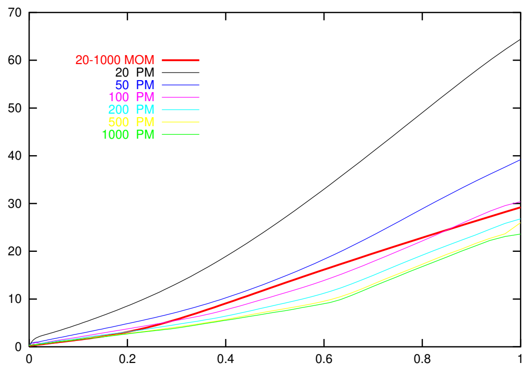
(b) Tiempos medios (ms.).



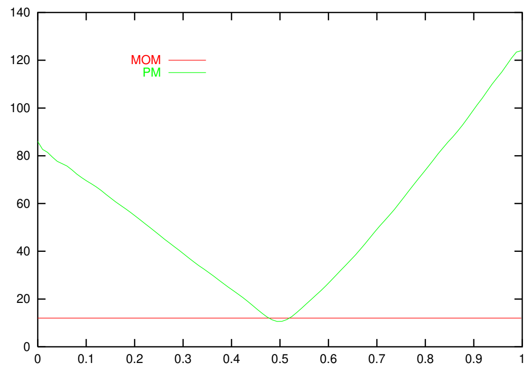
(c) Test aleatorio.



(d) Test lineal.



(e) Test exponencial.



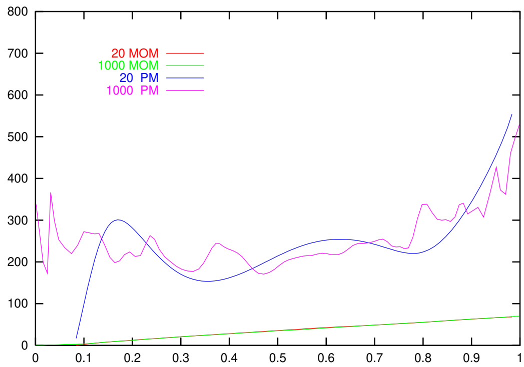
(f) Lod recuperado: 16971.

Figura 5.10: Resultados obtenidos para el modelo *Bunny*.

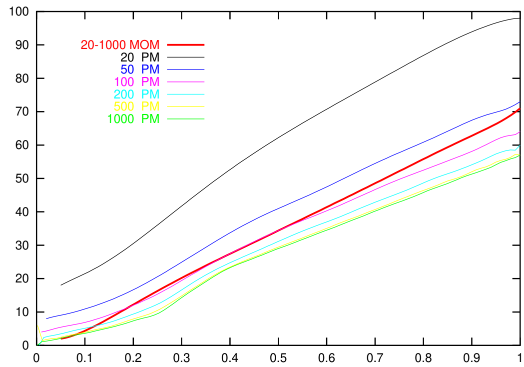
(a) Phone.

#LODs	Aleatorio		Lineal		Exponencial	
	MOM	PM	MOM	PM	MOM	PM
20	33.7	235.0	36.0	60.8	19.3	51.5
50	38.0	236.5	34.9	40.2	18.8	28.0
100	35.5	251.8	34.6	33.4	18.6	20.8
200	34.7	246.2	34.4	30.1	18.6	17.4
500	33.4	241.1	34.4	28.3	18.5	15.5
1000	35.0	251.2	34.3	27.8	18.5	14.8

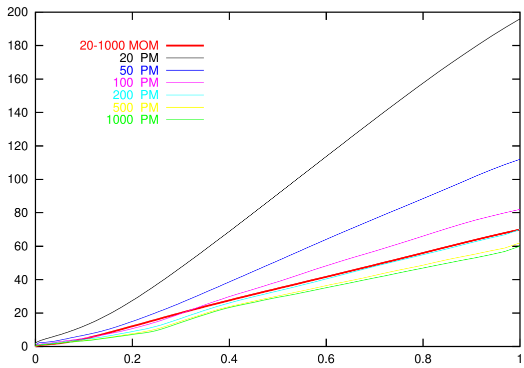
(b) Tiempos medios (ms.).



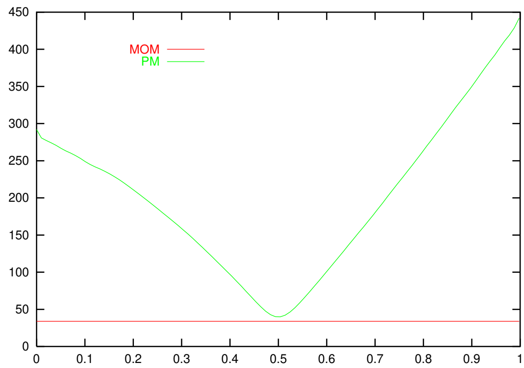
(c) Test aleatorio.



(d) Test lineal.



(e) Test exponencial.



(f) Lod recuperado: 41522.

Figura 5.11: Resultados obtenidos para el modelo *Phone*.

Capítulo 6

Coherencia Cuadro a Cuadro

El algoritmo de visualización de aproximaciones con nivel de detalle uniforme presentado en el capítulo 5 no es capaz de explotar la propiedad de la coherencia cuadro a cuadro. Este algoritmo no aprovecha el esfuerzo realizado en determinar qué triángulos pertenecen a la aproximación visualizada en el cuadro actual y de nuevo recorre la estructura de datos en busca de todos y cada uno de los triángulos que pertenecen a la aproximación requerida en el cuadro siguiente. Si entre dos cuadros consecutivos hay un alto porcentaje de triángulos que son comunes a ambas aproximaciones, aplicar la propiedad de la coherencia puede suponer un aumento importante en la eficiencia del algoritmo de visualización dependiendo únicamente de la distancia que exista entre las aproximaciones requeridas mejorando a medida que disminuye dicha distancia.

Por ejemplo, utilizando el modelo *Cow* que se muestra en la figura 6.1 se desea hacer una operación de zoom que durará dos segundos con una tasa de 25 cuadros por segundo. En la figura 6.2 se muestran las aproximaciones correspondientes a los cuadros 24 (izquierda) y 25 (derecha). En el cuadro 25 (3.004 triángulos) se han pintado en negro las regiones del modelo que son diferentes respecto de la aproximación requerida en el cuadro 24 (2.892 triángulos). La diferencia exacta entre ambas representaciones es de sólo 112 triángulos. En pocas palabras, poder determinar estos 112 triángulos de forma eficiente es el problema a resolver.

La coherencia cuadro a cuadro, entre otras, es una de las propiedades que se han utilizado en el capítulo 2 (sección 2.6) para caracterizar los distintos modelos multirresolución. La mayor parte de ellos, como se muestra en la tabla 2.4, hacen uso de esta propiedad. Los respectivos algoritmos de visualización de estos modelos parten de la aproximación de menor detalle y realizan una serie de operaciones de refinamiento hasta alcanzar la aproximación requerida. Hacer uso de la propiedad de la coherencia les permite extraer una nueva aproximación partiendo de la última aproximación requerida, no teniendo que recorrer el modelo comenzando de nuevo por la aproximación de menor detalle, lo cual supondría una seria desventaja. Gracias a la propiedad de la coherencia consiguen disminuir considerablemente los tiempos de extracción. En el caso de MOM,

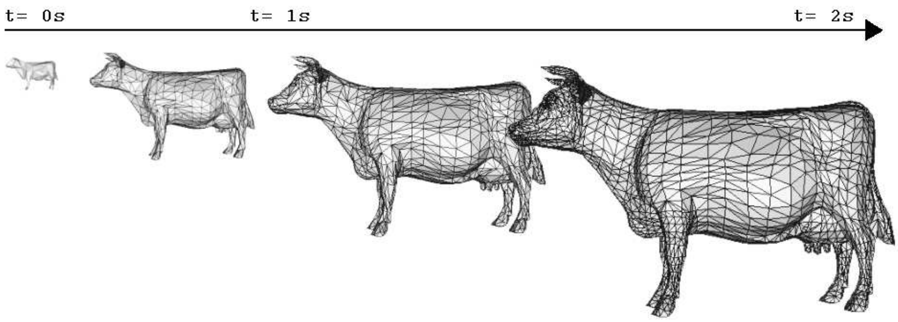


Figura 6.1: En una operación de zoom, de dos segundos de duración, el objeto pasa de su peor representación a la mejor.

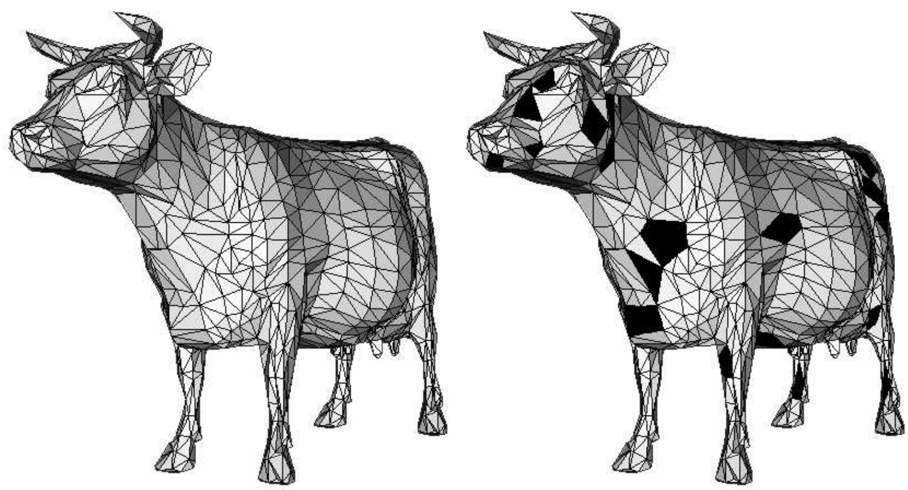


Figura 6.2: Diferencia entre aproximaciones requeridas en cuadros consecutivos. A la izquierda el cuadro 24 (2.892 triángulos), a la derecha el cuadro 25 (3.004 triángulos) y, en negro, las regiones que han cambiado.

tal y como se verá en la sección de resultados, también existe una mejora en los tiempos de extracción. Sin embargo, esta mejora no es tan relevante como en otros modelos ya que con el algoritmo del capítulo anterior, MOM es capaz de acceder directamente a los datos que forman la aproximación requerida sin tener que partir de una aproximación en concreto y realizar operaciones de simplificación o refinamiento.

En este capítulo se presenta una extensión de MOM que permitirá hacer uso de la propiedad de la coherencia cuadro a cuadro. De aquí en adelante, esta nueva variante será denominada *MOM-CCC*.

6.1 Ampliación de la estructura de datos

La estructura de datos básica de una representación MOM (presentada en el capítulo 4) no está diseñada para poder explotar la propiedad de la coherencia cuadro a cuadro. Es necesario realizar una ampliación y ésta se resume en los dos siguientes puntos:

1. Cada vértice ha de conocer de entre sus triángulos asociados cuáles son los que pertenecen a la última aproximación requerida. El objetivo es que no sea necesario consultar, para cada vértice, si cada uno de sus triángulos asociados pertenece o no a la nueva aproximación.

2. Cada triángulo ha de conocer a qué vértice se encuentra asociado. El objetivo es saber qué vértice actualizar como consecuencia de añadir o eliminar un triángulo de la última aproximación extraída.

Respecto al primer punto, la información que se guarda para cada vértice se amplía añadiendo un nuevo campo (campo *ultimoTriangulo*) que permite conocer el último triángulo extraído para ese vértice. Respecto al segundo, cada triángulo se amplía con el índice (campo *vertice*) del vértice al que está asociado. Este nuevo dato se almacena en una lista (campo *listaPunterosVertices*) ordenada según el campo *id* de cada triángulo y no como un campo más en la estructura *Triangulo*. En la figura 6.3 se muestra un ejemplo en el que se han esquematizado la estructura básica de MOM y la nueva estructura MOM-CCC donde es fácil observar las diferencias entre ambas. La nueva estructura de datos se muestra en la figura 6.4.

6.2 Proceso de construcción

El nuevo proceso de construcción parte de una representación básica de MOM y consiste en realizar la ampliación de la estructura de datos explicada en la sección anterior. A diferencia del proceso de construcción de la representación básica (sección 4.2), que era realizado fuera de línea, este nuevo proceso se caracteriza por:

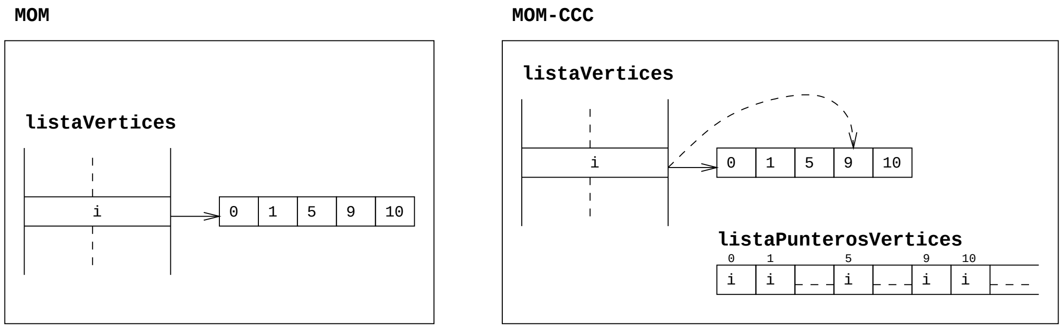


Figura 6.3: Esquemas de una estructura MOM y de una estructura MOM-CCC. A la izquierda, cada vértice almacena un puntero a su primer triángulo asociado. A la derecha, cada vértice almacena, además, un puntero que permite conocer los triángulos que pertenecen a la última aproximación extraída, y cada triángulo almacena el índice del vértice al que está asociado.

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

Figura 6.4: Estructura de datos de una representación MOM-CCC. Los nuevos campos añadidos a la estructura de datos de la figura 4.2 aparecen en cursiva.

- realizarse en tiempo de ejecución una vez la representación MOM resida en memoria, es decir, primero se lee y después se realiza la ampliación;
- los nuevos datos se generan a partir de una aproximación semilla dada, M_L, que puede ser elegida a conveniencia.

Dos son las tareas que se realizan en este nuevo proceso de construcción:

1. Dada una aproximación, M_L, actualizar para cada vértice el último triángulo asociado que pertenezca a M_L.
2. Para cada triángulo almacenado en la representación MOM, almacenar el vértice al que se encuentra asociado.

Estas dos tareas se realizan simultáneamente en un único proceso. En la figura 6.5 se muestra el algoritmo de construcción y se puede observar que es muy similar al algoritmo de visualización de una aproximación con nivel de detalle uniforme de la figura 5.2. A diferencia de éste, se recorren en su totalidad tanto la lista de vértices (bucle externo) como, para cada uno de ellos, la lista de triángulos asociados (bucle interno).

En la figura 6.6 se muestran los resultados de aplicar el algoritmo de construcción con $M_L = M_1$ utilizando el ejemplo del capítulo 4 (figura 4.6).

6.3 Coste de almacenamiento

Para realizar el estudio del coste de almacenamiento de la nueva estructura de datos (figura 6.4) se va a suponer, al igual que en el estudio realizado en la sección 4.3 correspondiente a la estructura de datos básica, que el coste de almacenar un valor entero, real o puntero es de una palabra para todos ellos. Así, un elemento de tipo *Vertice* tendrá un coste de 5 palabras, un elemento de tipo *Triangulo* tendrá un coste de 3 palabras, un elemento de tipo *PunteroVertice* tendrá un coste de 1 palabra y, por último, un elemento de tipo *ControlLod* tendrá un coste de 2 palabras. Teniendo en cuenta que el número de elementos de tipo *Triangulo* y el de *PunteroVertice* son el mismo, el coste total será de $5|V^r| + 4|T^r| + 2n$. El cálculo de $|V^r|$, $|T^r|$ y n es el mismo que el presentado en la sección 4.3 relativo a la estructura de datos básica de una representación MOM.

En la tabla 6.1 se resumen los costes de almacenamiento teóricos obtenidos para el caso medio (ver sección 4.3.1.2). La nueva representación multirresolución presenta un coste alrededor de un 30% más, comparada con el modelo básico. Comparada con la del modelo original, si éste se representa mediante una lista de vértices y una lista de triángulos, es 3'44 veces mayor si se utiliza un método de simplificación basado en eliminación de vértices en el proceso de construcción, y entre 3'44 y 5'78 veces mayor si se utiliza un método de simplificación basado en contracción de aristas.

```
// Dada una aprox., M_L, obtener los triángulos generados
TG= listaControlLod[L_s].triangulosGenerados;

// NT es el numero de triángulos del objeto MOM
NT= nTriangulos;

primero= 0;
i= 0;
// Mientras hayan triángulos por visitar
mientras NT > 0 hacer
  ultimo= listaVertices[i+1].primerTriangulo;
  ultimoTriangulo= primero;
  NT= NT - (ultimo - primero);
  // Recorrer la lista de triángulos asociados a v_i
  para T= primero hasta ultimo - 1 hacer
    id= listaTriangulos[T].id;
    listaPunterosVertices[id]= i;
    si id < TG entonces ultimoTriangulo= T + 1;
  fin para
  listaVertices[i].ultimoTriangulo= ultimoTriangulo;
  i= i + 1; // siguiente vértice
  primero= ultimo;
fin mientras
```

Figura 6.5: Algoritmo de construcción de una representación MOM-CCC.

listaVertices

v4	v2	v1	v5	v8	v0	v3	v6	v7
0	6	11	13	14	~	~	~	~
6	11	11	14	15	~	~	~	~

listaTriangulos

2	3	4	6	7	8	0	1	5	9	10	13	14	12	11	15
v1	v6	v3	v8	v4	v7	v5	v2	v8	v4	v6	v8	v6	v7	v6	v8
v6	v3	v7	v1	v8	v4	v2	v6	v5	v8	v4	v5	v8	v6	v7	v6

listaPunterosVertices

1	1	0	0	0	1	0	0	0	1	1	4	3	2	2	4

listaControlLod

0	1	2
9	13	16

Figura 6.6: Ejemplo de la estructura de datos de un objeto MOM-CCC ($M_L = M_1$).

	$\lvert V^r\rvert$	$\lvert T^r\rvert$	n	Total	Ratio
MOM: $4\lvert V^r\rvert + 3\lvert T^r\rvert + 2n$					
Eliminación de vértices	$\lvert V\rvert$	$3\lvert T\rvert$	$\lvert V\rvert$	$6\lvert V\rvert + 9\lvert T\rvert$	2'67
Contracción de aristas (mejor caso)	$\lvert V\rvert$	$3\lvert T\rvert$	$\lvert V\rvert$	$6\lvert V\rvert + 9\lvert T\rvert$	2'67
Contracción de aristas (peor caso)	$2\lvert V\rvert$	$5\lvert T\rvert$	$\lvert V\rvert$	$10\lvert V\rvert + 15\lvert T\rvert$	4'44
MOM-CCC: $5\lvert V^r\rvert + 4\lvert T^r\rvert + 2n$					
Eliminación de vértices	$\lvert V\rvert$	$3\lvert T\rvert$	$\lvert V\rvert$	$7\lvert V\rvert + 12\lvert T\rvert$	3'44
Contracción de aristas (mejor caso)	$\lvert V\rvert$	$3\lvert T\rvert$	$\lvert V\rvert$	$7\lvert V\rvert + 12\lvert T\rvert$	3'44
Contracción de aristas (peor caso)	$2\lvert V\rvert$	$5\lvert T\rvert$	$\lvert V\rvert$	$12\lvert V\rvert + 20\lvert T\rvert$	5'78

Tabla 6.1: Costes teóricos de almacenamiento en el caso medio para representaciones MOM y MOM-CCC.

6.4 Algoritmo de visualización

El algoritmo que se presenta en esta sección parte de la aproximación visualizada en el cuadro actual y realiza una secuencia de operaciones para, en el siguiente cuadro, visualizar la aproximación requerida. Este modo de trabajo es el mismo que, por ejemplo, propone Hoppe para *Progressive Meshes* [Hoppe, 1996, Hoppe, 1998a]. Estas operaciones son *Siguiente*, que avanza a la siguiente aproximación de mayor detalle, y *Anterior*, que avanza a la siguiente aproximación de menor detalle. Dependiendo de si la malla requerida supone un aumento o una disminución en el detalle respecto de la actual, se ejecutará la operación *Siguiente* o *Anterior*, respectivamente, tantas veces como distancia haya entre ambas aproximaciones. Una vez alcanzada la aproximación requerida se visualizan los triángulos seleccionados. A continuación se detallan ambas operaciones así como el nuevo algoritmo de visualización.

6.4.1 Operación *Siguiente*

La operación *Siguiente* realiza dos tareas (figura 6.7):

1. Determina los triángulos que dejan de pertenecer a la siguiente aproximación de mayor detalle. Utilizando el ejemplo de la figura 6.6 que muestra el estado de la representación MOM para el lod M_1 y, teniendo en cuenta que se desea alcanzar M_0, las caras que no van a pertenecer a M_0 serán *t9, t10, t11* y *t12*.

2. Para cada uno de los triángulos que ya no pertenecen a la nueva aproximación (*t9, t10, t11* y *t12*), se obtiene el vértice al que está asociado (*v1, v1, v4* y *v3*) y lo elimina de la solución, es decir, decrementa el puntero *ultimoTriangulo* en una posición.

```
// Dado una aprox., M_{L_a}, recuperar los triángulos generados
TG_a= listaControlLod[L_a].triangulosGenerados;
L_b= L_a - 1; // Siguiente aprox. de mayor detalle
TG_b= listaControlLod[L_b].triangulosGenerados;

// Recorrer los triángulos entre TG_b y TG_a
inicio= TG_b;
fin= TG_a;
para T= inicio hasta fin - 1 hacer
  v= listaPunterosVertices[T].vertice;
  listaVertices[v].ultimoTriangulo= listaVertices[v].ultimoTriangulo- 1;
fin para
```

Figura 6.7: Operación *Siguiente*.

6.4.2 Operación *Anterior*

La operación *Anterior* (figura 6.8) realiza dos tareas similares a la operación *Siguiente*, y son:

1. Determina los triángulos que van a pertenecer a la siguiente aproximación de menor detalle. Siguiendo con el ejemplo de la figura 6.6 y, teniendo en cuenta que se desea alcanzar M_2, las caras que ahora van a pertenecer a la solución serán la *t13, t14* y *t15*.
2. Para cada uno de los triángulos que pertenecen a la nueva aproximación (*t13, t14* y *t15*), se obtiene el vértice al que está asociado (*v2, v2* y *v4*) y lo añade, es decir, incrementa el puntero *ultimoTriangulo* en una posición.

6.4.3 Algoritmo principal

En la figura 6.9 se muestra el algoritmo principal. Este algoritmo presenta dos partes claramente diferenciadas, y son las siguientes:

1. En primer lugar, ejecuta la secuencia de operaciones *Siguiente* o *Anterior* hasta alcanzar la aproximación requerida.
2. Después, visualiza las caras que pertenecen a la aproximación requerida, de una manera muy similar a como lo hace el algoritmo que no explota la coherencia (figura 5.2). Sin embargo, se pueden observar algunas diferencias. La más importante es que el bucle interno **mientras** ha desaparecido y un bucle **para** aparece en su lugar debido a que ahora no hace falta chequear si el triángulo se ha de dibujar o no. Esta pequeña mejora supone un aumento en la eficiencia tal y como se mostrará en la sección de resultados.

```
// Dada una aprox., M_{L_a}, recuperar los triángulos generados
TG_a= listaControlLod[L_a].triangulosGenerados;
L_b= L_a + 1; // Siguiente aprox. de menor detalle
TG_b= listaControlLod[L_b].triangulosGenerados;

// Recorrer los triángulos entre TG_b y TG_a
inicio= TG_a;
fin= TG_b;
para T= inicio hasta fin - 1 hacer
  v= listaPunterosVertices[T].vertice;
  listaVertices[v].ultimoTriangulo= listaVertices[v].ultimoTriangulo+ 1;
fin para
```

Figura 6.8: Operación *Anterior*.

6.5 Coste temporal

Para analizar el coste temporal del algoritmo de visualización (figura 6.9) se van a estudiar por separado cada una de las dos partes de las que consta el algoritmo y que se detallan en la sección anterior.

La primera parte hace referencia al conjunto de operaciones *Siguiente* y *Anterior* que se han de realizar para pasar de la aproximación actual a la nueva aproximación requerida. Si M_i es la aproximación actual, una operación *Siguiente* elimina de M_i un número dado de triángulos, $|\mathcal{T}'_{i-1}|$, y presenta un coste lineal respecto a dicho número de triángulos. Si M_{i-1} es la aproximación actual, la operación *Anterior* añade un número dado de triángulos a la solución actual, $|\mathcal{T}'_{i-1}|$, y también presenta un coste lineal respecto a dicho número de triángulos. Así, el coste de pasar de una aproximación M_a a una aproximación M_b viene dado por la siguiente ecuación:

$$\sum_{i=a}^{b-1} |\mathcal{T}'_i|, a < b \tag{6.1}$$

En el peor caso, $|\mathcal{T}'_i| = |T_i|, 0 \leq i < n-1$ (ver sección 4.3.1.1), y en el sumatorio se obtendría un coste cuadrático respecto a $|T|$.

En el caso medio (ver sección 4.3.1.2), $|\mathcal{T}'_i|$ es 4 si el método de simplificación utilizado en el proceso de construcción se basa en la eliminación de vértices, y 8 si el método se basa en la contracción de aristas. Bien la operación *Siguiente* o bien la operación *Anterior* se repiten p veces, siendo p la diferencia entre la aproximación actual y la requerida ($b-a$ en la ecuación 6.1). Por lo tanto, teniendo en cuenta que $|\mathcal{T}'_i|$ es una constante, el coste de esta parte del algoritmo es $O(p)$.

El análisis teórico de la segunda parte del algoritmo coincide con el realizado en la sección 5.2. En dicha sección se concluía que en el caso medio, el orden del algoritmo

```
// Realizar la secuencia de operaciones
si LodActual < LodRequerido entonces // ir a una aprox.  de menor detalle
  mientras LodActual < LodRequerido hacer
    Anterior ();
    LodActual= LodActual + 1;
  fin mientras
sino // ir a una aprox.  de mayor detalle
  mientras LodActual > LodRequerido hacer
    Siguiente ();
    LodActual= LodActual - 1;
  fin mientras
fin si

// Dada una aprox., M_L, recuperar de la lista de control sus datos
TG= listaControlLod[L].triangulosGenerados;
VE= listaControlLod[L].verticesEliminados;

// NT es el numero de triángulos de M_L
NT= TG - listaVertices[VE].primerTriangulo;

// El primer vértice será el v_i
i= VE;

// Mientras hayan triángulos por pintar
mientras NT > 0 hacer
  // Recorrer la lista de triángulos asociados a v_i
  primero= listaVertices[i].primerTriangulo;
  ultimo= listaVertices[i].ultimoTriangulo;
  pv_i= direccion (listaVertices[i]);
  para cont= primero hasta ultimo - 1 hacer
    pintaTriangulo (pv_i,
                    listaTriangulos[cont].v1,
                    listaTriangulos[cont].v2);
  fin para
  i= i + 1; // siguiente vértice
  NT= NT - (ultimo - primero);
fin mientras
```

Figura 6.9: Algoritmo de visualización.

era lineal respecto al número de vértices y triángulos que pertenecen a la aproximación requerida M_i, $O(|V_i| + |T_i|)$.

Sumando ambos costes, el coste temporal del algoritmo que se obtiene es $O(p + |V_i| + |T_i|)$.

6.6 Resultados

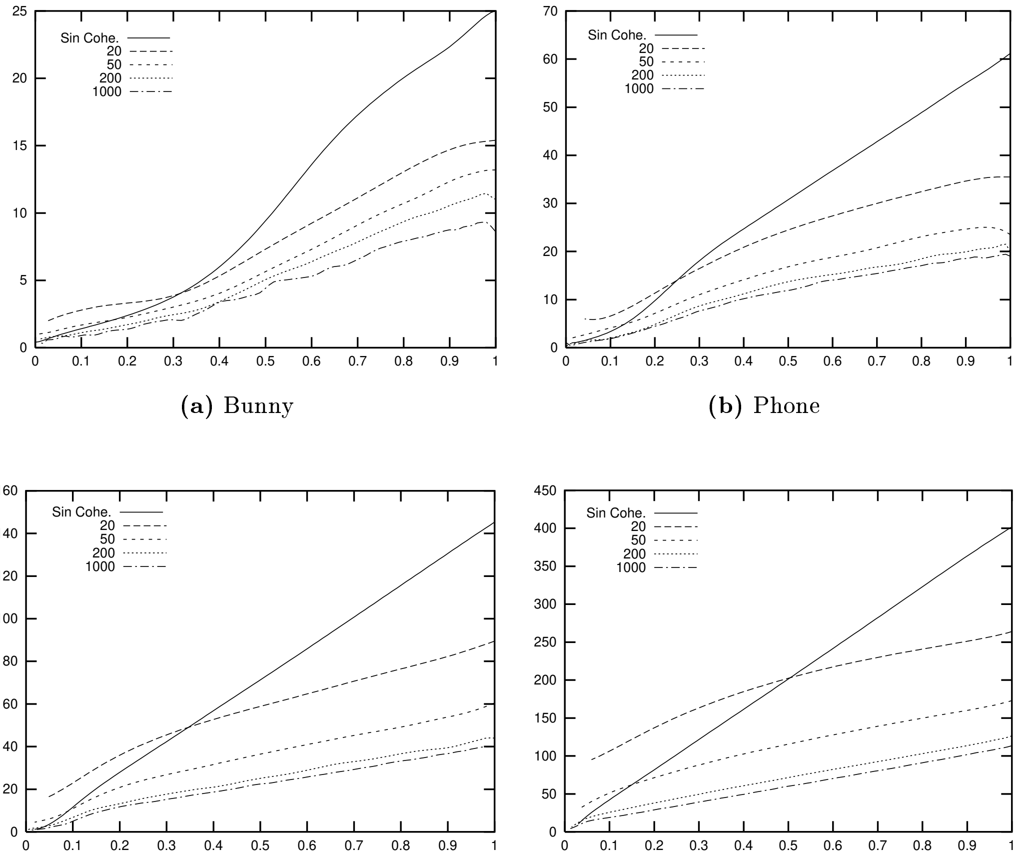
Los experimentos han sido llevados a cabo utilizando el entorno de trabajo descrito en la sección 4.4. Estos experimentos han consistido en medir los tiempos de extracción para MOM-CCC y compararlos con los obtenidos para MOM.

La figura 6.10 muestra cuatro gráficas con las prestaciones de ambos algoritmos para diferentes objetos. El eje X representa la aproximación requerida (donde 0 representa la de menor detalle y 1 la de mayor detalle), y el eje Y representa el tiempo de extracción en milisegundos. Se dibujan cuatro curvas para el algoritmo MOM-CCC ya que sus prestaciones son dependientes de la última aproximación requerida. Cada curva ha sido obtenida mediante la petición de una secuencia de aproximaciones de diferente longitud (20, 50, 200 y 1000) correlacionadas linealmente, de forma que la diferencia entre dos aproximaciones requeridas consecutivamente es un paso constante. El comportamiento coincide con el análisis teórico del coste del algoritmo, es decir, dado un paso constante p el coste del algoritmo es lineal respecto a la complejidad de la aproximación requerida. Se puede observar que si el número de aproximaciones solicitadas se incrementa (el paso es más pequeño), el tiempo de extracción disminuye, dado que el número de operaciones ejecutadas (*Siguiente* o *Anterior*) entre dos aproximaciones visualizadas consecutivamente es menor. Además, la mejora es más patente cuando se están manipulando aproximaciones complejas, y nula con aproximaciones de muy poco detalle. La tabla 6.2 muestra los tiempos medios de los resultados experimentales que se muestran en esta figura.

	#lods	Bunny	Phone	Isis	Buda
MOM	-	13	31	73	203
	20	9	25	64	217
MOM-CCC	50	6	17	39	125
	200	5	14	27	80
	1000	4	12	24	67

Tabla 6.2: Tiempos medios de extracción (ms.).

La figura 6.11 muestra cuatro gráficas de los tiempos de extracción de la aproximación $M_{n/2}$ para ambos algoritmos cuando la última aproximación visualizada pudo ser cualquier otra. Se puede observar el intervalo de aproximaciones, alrededor de la requerida, para el que el algoritmo MOM-CCC se comporta mejor que MOM y viceversa. El

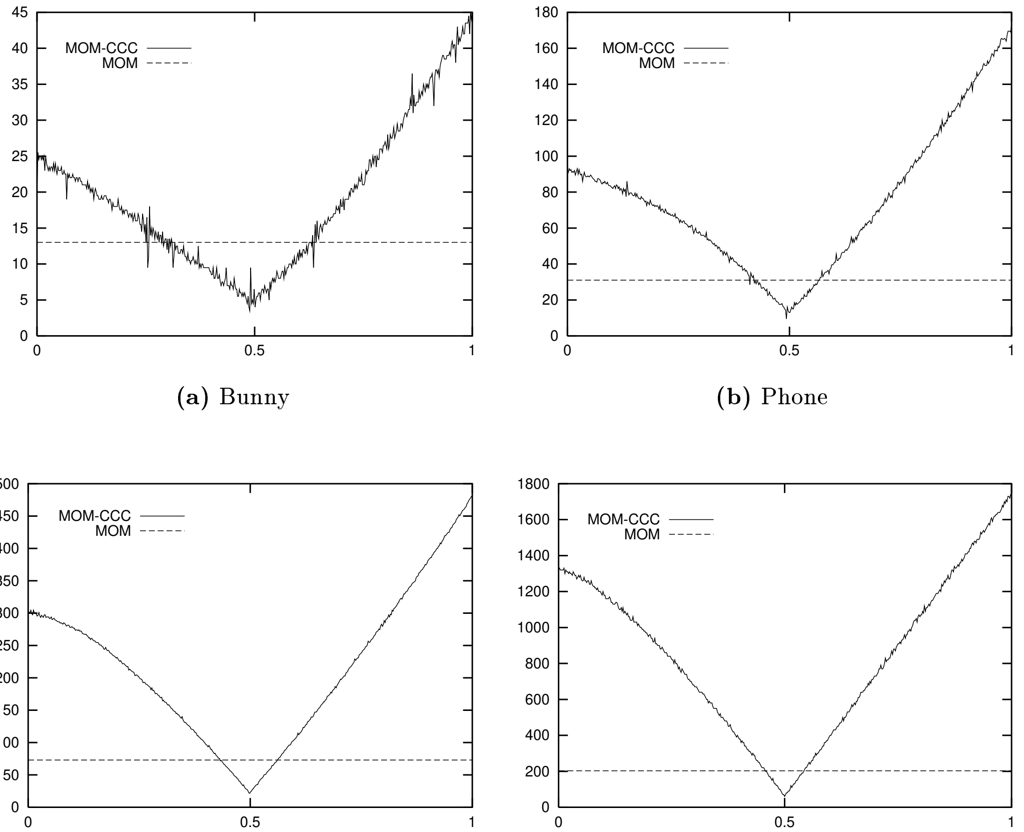


(a) Bunny **(b)** Phone

(c) Isis **(d)** Buda

Figura 6.10: Prestaciones de los algoritmos MOM y MOM-CCC.

tiempo de recuperación de MOM es constante porque es independiente de la última aproximación visualizada. El algoritmo MOM-CCC tiene su mejor comportamiento cuando la aproximación requerida está cerca de la que fue solicitada en el cuadro anterior, y es peor a medida que la distancia aumenta. Se puede también observar cómo efectivamente el algoritmo tiene una respuesta lineal respecto al número de operaciones *Siguiente* o *Anterior* que ha de realizar para alcanzar la aproximación requerida, tal como cabe esperar según el análisis teórico del coste.



(a) Bunny **(b)** Phone

(c) Isis **(d)** Buda

Figura 6.11: Prestaciones de ambos algoritmos (MOM y MOM-CCC) cuando $M_{n/2}$ es solicitada.

La tabla 6.3 muestra el coste de almacenamiento de la estructura de datos básica de MOM y de la nueva propuesta en este capítulo. Se puede observar que el incremento es siempre cercano al 29%. Este incremento coincide con el incremento teórico obtenido en el estudio del coste de almacenamiento para el caso medio (ver sección 6.3).

	Bunny	Phone	Isis	Buda
Vértices	34.835	83.045	187.871	543.653
Triángulos	69.451	165.963	375.736	1.085.636
MB.	1'193	2'850	6'450	18'646
n	33.990	81.668	187.370	543.106
MB. MOM	2'876	6'940	15'667	43'957
MB. MOM-CCC	3'703	8'939	20'173	56'537
MB. Inc %	28'8	28'8	28'8	28'6

Tabla 6.3: Costes de almacenamiento.

6.7 Conclusiones

En este capítulo se ha presentado una extensión del modelo básico de MOM que permite explotar la propiedad de la coherencia cuadro a cuadro. La estructura de datos se ha ampliado para poder lograr el objetivo, pero la nueva información se genera en tiempo de ejecución una vez el modelo ha sido leído. Aparece un incremento en el coste de almacenamiento y los resultados experimentales muestran que este es alrededor de un 29%.

Los resultados experimentales también han mostrado las ventajas e inconvenientes de explotar la propiedad de la coherencia en MOM. Se ha estudiado el comportamiento de la nueva representación variando la distancia entre las aproximaciones requeridas. El algoritmo que hace uso de la propiedad se muestra beneficioso cuando las aproximaciones requeridas en cuadros consecutivos es pequeña y el nivel de detalle es alto.

Capítulo 7

Representación con Abanicos de Triángulos

Para acelerar el proceso de visualización de modelos poligonales se han desarrollado diversos mecanismos como son, por ejemplo, las primitivas básicas de dibujo tira y abanico de triángulos (ver figura 7.1). Además, este tipo de primitivas están disponibles en la mayor parte de las librerías gráficas, por ejemplo, en *OpenGL*. El uso de estas primitivas produce una reducción del tiempo de computación y un importante ahorro de memoria debido a que se opera sobre un número menor de vértices ya que para dibujar una tira o un abanico de n triángulos hay que enviar al procesador gráfico $n + 2$ vértices en lugar de $3n$.

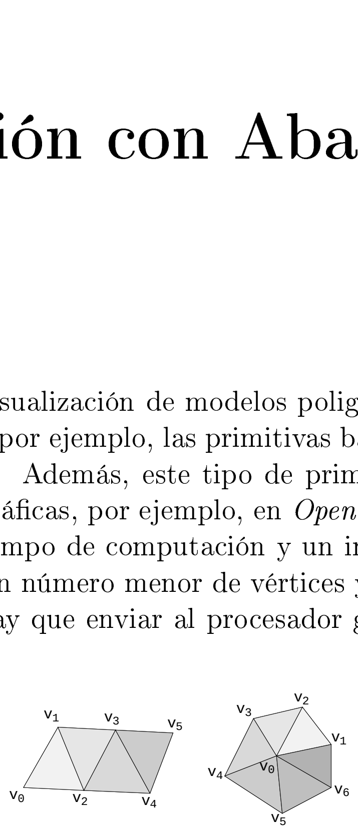


Figura 7.1: Ejemplo de tira y abanico de triángulos. A la izquierda, tira definida por *v0, v1, v2, v3, v4, v5*, y, a la derecha, abanico definido por *v0, v1, v2, v3, v4, v5, v6, v1*

Si se trabaja con modelos estáticos, la conectividad de la malla permanece constante y resulta sencillo hacer uso de las tiras y abanicos de triángulos, ya que sólo hay que generarlas una vez y además se puede hacer en un proceso fuera de línea. Sin embargo, en un modelo multirresolución, la conectividad de la superficie cambia al cambiar el nivel de detalle, probablemente a cada cuadro, requiriendo una generación dinámica de las tiras o abanicos.

Muy pocos autores se han preocupado por, además de extraer aproximaciones de forma eficiente, hacer uso de las tiras o abanicos de triángulos para reducir aún más los tiempos de visualización. Hoppe [Hoppe, 1997], propone obtener tiras de triángulos utilizando un algoritmo voraz una vez se hayan determinado los triángulos que forman la aproximación requerida. El-Sana et al. [El-Sana et al., 1999] presentan *Skip-Strips*, que mantiene las tiras de triángulos aunque cambie el nivel de detalle de la aproximación. Una estructura *Skip-Strip* se construye en tiempo de ejecución a partir del modelo multirresolución. Al mismo tiempo se obtienen las tiras del modelo original. Cada vez que el nivel de detalle cambie, la estructura *Skip-Strip* se actualiza basándose en la aproximación requerida, lo que permite la actualización de las tiras y finalmente su visualización. Las tiras que genera son cortas y propone que sean concatenadas en una etapa previa a la visualización. Floriani et al. [Floriani et al., 1998], utilizan los abanicos de triángulos como medio de compresión para codificar de forma implícita una representación MT. Consiguen reducir el coste de almacenamiento en un 60% respecto de la versión explícita de MT, pero a costa de aumentar los tiempos de extracción en un factor de 10.

Mallas Ordenadas Multirresolución se ha presentado como un método eficiente para la visualización interactiva de superficies poligonales. En este capítulo se aborda cómo construir un objeto MOM de forma que se utilice el abanico de triángulos como primitiva básica de representación a fin de que en el proceso de extracción la aproximación se recupere directamente como un conjunto de abanicos y no haya que generarlos en una etapa posterior. De aquí en adelante, esta nueva variante será denominada *MOM-Fan*. Un ejemplo de diferentes niveles de detalle de un objeto visualizado con abanicos se puede ver en la figura 7.2, donde cada abanico ha sido coloreado con un color distinto y su borde ha sido resaltado.

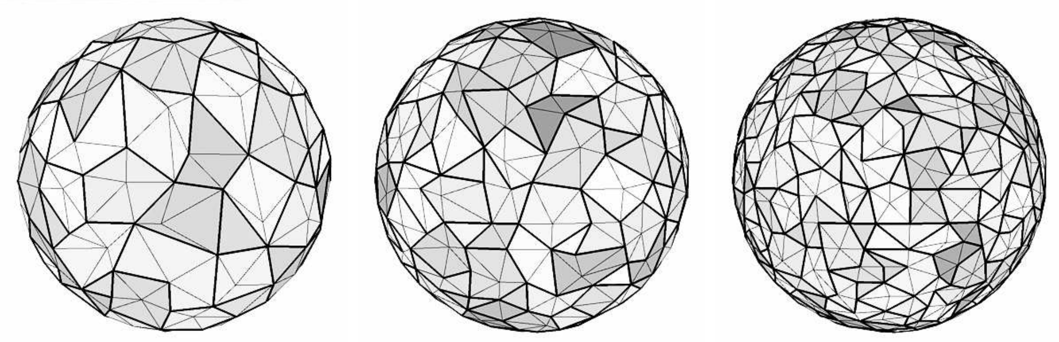


Figura 7.2: Tres aproximaciones del modelo *Sphere* visualizados utilizando abanicos.

7.1 Estructura de datos

La principal diferencia respecto de la estructura de datos básica (ver capítulo 4) es que ahora se han de almacenar los triángulos asociados a cada vértice como la secuencia de vértices que forman el abanico. Es decir, cada vértice va a apuntar a la secuencia de vértices que define el abanico, en lugar de a un conjunto de triángulos. En la figura 7.3 se muestra la nueva estructura de datos y en la figura 7.4 se muestra un ejemplo en el que se han esquematizado la estructura básica de MOM y la nueva estructura MOM-Fan donde es fácil observar las diferencias entre ambas. Los cambios respecto a la estructura de datos básica se centran en tres puntos:

```
estructura Vertice {
  real coordenadas[3];
  entero segundoVerticeAbanico;
};
estructura VerticeAbanico {
  entero id;
  estructura Vertice *v;
};
estructura ControlLod {
  entero verticesEliminados;
  entero triangulosGenerados;
};
estructura MOM-Fan {
  entero nVertices, nVerticesAbanico, nLods;
  estructura Vertice listaVertices[nVertices];
  estructura VerticeAbanico listaVerticesAbanico[nVerticesAbanico];
  estructura ControlLod listaControlLod[nLods];
};
```

Figura 7.3: Estructura de datos de una representación MOM-Fan. Las diferencias con la estructura básica aparecen en cursiva.

Estructura Vertice Cada vértice representa el vértice inicial de un abanico de triángulos y almacena, además de sus coordenadas (campo *coordenadas*), un puntero al segundo vértice del abanico (campo *segundoVerticeAbanico*).

Estructura VerticeAbanico Esta estructura sustituye a la estructura *Triangulo*. Cada elemento consiste en un puntero al vértice en la lista de vértices (campo v) y en un identificador (campo *id*). El identificador referencia al triángulo del abanico representado por el vértice v.

Lista listaVerticesAbanico Esta lista sustituye a la lista *listaTriangulos*. En ella se almacenan las secuencias de vértices que forman los abanicos (excepto el vértice inicial de cada uno de ellos).

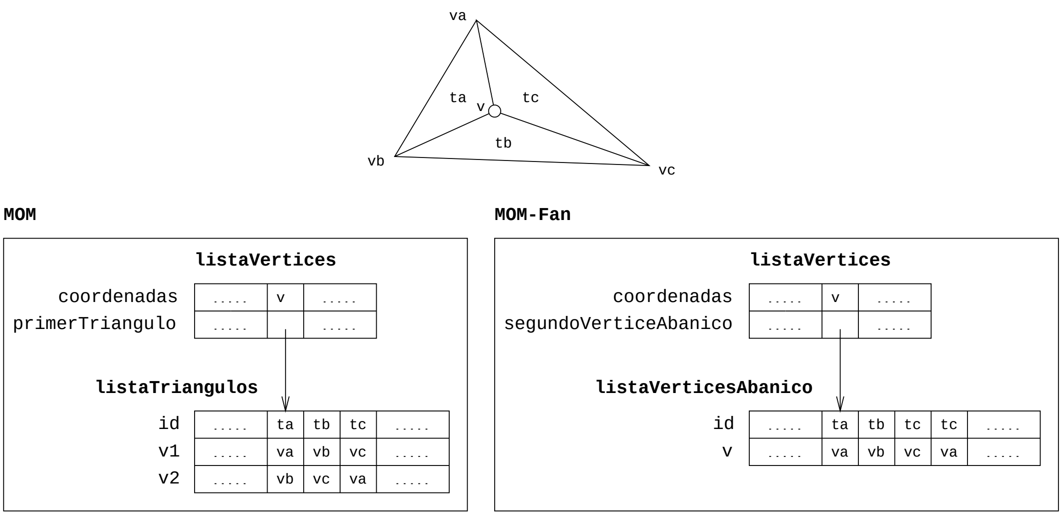


Figura 7.4: Esquemas de una estructura MOM y de una estructura MOM-Fan. En ambas se representa al vértice v y los triángulos que lo rodean.

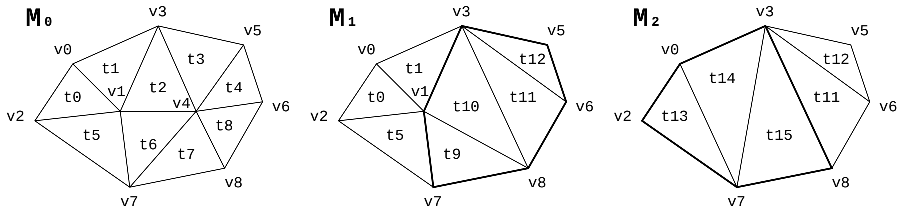
7.2 Proceso de construcción

El proceso de construcción es el mismo que el explicado en la sección 4.2 para la estructura de datos básica. La única diferencia reside en que ahora hay que crear un abanico con los triángulos que rodean el vértice eliminado y almacenar la secuencia de vértices que lo forman en lugar de almacenar los triángulos como tales. Además, ahora el orden de los vértices viene dado por el propio abanico mientras que antes el orden de los triángulos venía dado por el campo *id*.

En la figura 7.5 se muestra un ejemplo de construcción de una representación MOM-Fan y también se incluye la respectiva representación MOM para su comparación.

7.3 Coste de almacenamiento

Para realizar el estudio del coste de almacenamiento de la nueva estructura de datos (figura 7.3) se va a suponer, al igual que en el estudio realizado en la sección 4.3 correspondiente a la estructura de datos básica, que el coste de almacenar un valor entero, real o puntero es de una palabra para todos ellos. Así, un elemento de tipo *Vertice* tendrá un coste de 4 palabras, un elemento de tipo *VerticeAbanico* tendrá un coste de 2 palabras y, por último, un elemento de tipo *ControlLod* tendrá un coste de 2 palabras. De esta forma, el coste total será de $4|V^r| + 2|A^r| + 2n$, donde $|A^r|$ es el número de vértices abanico almacenados en la representación multirresolución. El cálculo de $|V^r|$ y n es el mismo que el presentado en la sección 4.3 relativo a la estructura de datos básica de una



MOM

listaVertices

coordenadas	v4	v1	v3	v0	v2	v5	v6	v7	v8
primerTriangulo	0	6	11	15	~	~	~	~	~

listaTriangulos

id	2	3	4	6	7	8	0	1	5	9	10	11	12	14	15	13
v1	v1	v3	v5	v7	v8	v6	v2	v0	v7	v8	v3	v6	v5	v7	v8	v7
v2	v3	v5	v6	v1	v7	v8	v0	v3	v2	v7	v8	v8	v6	v0	v7	v2

listaControlLod

verticesEliminados	0	1	2
triangulosGenerados	9	13	16

MOM-Fan

listaVertices

coordenadas	v4	v1	v3	v0	v2	v5	v6	v7	v8
segundoVerticeAbanico	0	7	13	18	~	~	~	~	~

listaVerticesAbanicos

id	2	3	4	8	7	6	6	0	1	10	9	5	5	12	11	15	14	14	13	13
v	v1	v3	v5	v6	v8	v7	v1	v2	v0	v3	v8	v7	v2	v5	v6	v8	v7	v0	v7	v2

listaControlLod

verticesEliminados	0	1	2
triangulosGenerados	9	13	16

Figura 7.5: Ejemplo de representación MOM y MOM-Fan. Las celdas rayadas indican la inicialización. En gris oscuro, los datos añadidos después de la eliminación de $v4$, y en gris claro, los datos añadidos después de la eliminación de $v1$, ambos en la primera etapa. En blanco, los datos añadidos como resultado de la segunda etapa.

representación MOM. Sin embargo, el cálculo de $|A^r|$, aunque similar, difiere un poco del de $|T^r|$.

7.3.1 Cálculo de $|A^r|$

El número de vértices que forman un abanico de t triángulos es siempre de $t+2$. Teniendo en cuenta que el vértice inicial se almacena en la lista de vértices, son los restantes $t+1$ vértices los que realmente se almacenan en la lista de vértices abanico. En la sección 4.3 se concluía que, en el peor caso, $|T^r|$ era de orden cuadrático respecto al número de triángulos de la malla original. Como $|T^r| < |A^r|$, se mantiene el orden cuadrático de $|A^r|$.

En el caso medio, el número de vértices abanicos será el número de triángulos, $|T^r|$, más uno por abanico. En el caso de utilizar un método de simplificación basado en la eliminación de vértices, $|T^r| \leq 3|T|$ por lo que $|A^r| \leq 3'5|T|$. Si se utiliza un método basado en la contracción de aristas, $|T^r| \leq 5|T|$ por lo que $|A^r| \leq 5'5|T|$.

En la tabla 7.1 se resumen los costes de almacenamiento teóricos obtenidos para el caso medio. La nueva representación multirresolución, comparada con MOM, presenta un coste alrededor de un 17% menos. Comparada con la del modelo original, si este se representa mediante una lista de vértices y una lista de triángulos, es 2'22 veces mayor si se utiliza un método de simplificación basado en eliminación de vértices en el proceso de construcción, y entre 2'22 y 3'56 veces mayor si se utiliza un método de simplificación basado en contracción de aristas.

	$\lvert V^r\rvert$	$\lvert T^r\rvert/\lvert A^r\rvert$	n	Total	Ratio
MOM: $4\lvert V^r\rvert + 3\lvert T^r\rvert + 2n$					
Eliminación de vértices	$\lvert V\rvert$	$3\lvert T\rvert$	$\lvert V\rvert$	$6\lvert V\rvert + 9\lvert T\rvert$	2'67
Contracción de aristas (mejor caso)	$\lvert V\rvert$	$3\lvert T\rvert$	$\lvert V\rvert$	$6\lvert V\rvert + 9\lvert T\rvert$	2'67
Contracción de aristas (peor caso)	$2\lvert V\rvert$	$5\lvert T\rvert$	$\lvert V\rvert$	$10\lvert V\rvert + 15\lvert T\rvert$	4'44
MOM-Fan: $4\lvert V^r\rvert + 2\lvert A^r\rvert + 2n$					
Eliminación de vértices	$\lvert V\rvert$	$3'5\lvert T\rvert$	$\lvert V\rvert$	$6\lvert V\rvert + 7\lvert T\rvert$	2'22
Contracción de aristas (mejor caso)	$\lvert V\rvert$	$3'5\lvert T\rvert$	$\lvert V\rvert$	$6\lvert V\rvert + 7\lvert T\rvert$	2'22
Contracción de aristas (peor caso)	$2\lvert V\rvert$	$5'5\lvert T\rvert$	$\lvert V\rvert$	$10\lvert V\rvert + 11\lvert T\rvert$	3'56

Tabla 7.1: Costes teóricos de almacenamiento en el caso medio para representaciones MOM y MOM-Fan.

7.4 Algoritmo de visualización

Supongamos que se desea visualizar la aproximación de mayor detalle del ejemplo de la figura 7.5. En este caso, deben visualizarse los triángulos que aparecen en la malla M_0.

Los abanicos que se deben generar, ordenados según la aparición de los vértices en la estructura de datos son:

- (*v4 v1 v3 v5 v6 v8 v7 v1*) que corresponden a los triángulos (*t2 t3 t4 t8 t7 t6*);
- (*v1 v2 v0 v3*) y (*v1 v7 v2*), que corresponden a los triángulos (*t0 t1*) y (*t5*), respectivamente.

El primer vértice del abanico siempre corresponde al vértice eliminado. El resto de vértices están almacenados en *listaVerticesAbanicos*, y asociados al vértice eliminado. Sin embargo, puede ocurrir que algunos vértices no pertenezcan a la aproximación requerida. En el ejemplo, debido a que *t10* y *t9*, implícitamente asociados a *v1*, no se deben visualizar, hay un salto desde el vértice *v3* al vértice *v7* (ver malla M_1). Para resolver estos saltos sin tener que partir los abanicos es necesario introducir el vértice *v1* entre *v3* y *v7*, de modo que se producen dos triángulos degenerados. Así, mientras el primer abanico se resuelve sin triángulos degenerados (*v4 v1 v3 v5 v6 v8 v7 v1*), el segundo abanico (*v1 v2 v0 v3 v1 v7 v2*) incluye dos, que son (*v1 v3 v1*) y (*v1 v1 v7*).

El algoritmo de visualización, que se muestra en la figura 7.6, requiere calcular inicialmente el número de triángulos a visualizar. Después, recorre la estructura de datos extrayendo los abanicos de triángulos que pertenecen a la aproximación requerida.

Cálculo del número de triángulos

Dada una aproximación, M_L, en la lista de control se encuentra almacenado el número de vértices que no pertenecen a dicho lod, *VE*, y el número de triángulos generados, TG. A partir de estos datos es fácil obtener cuántos triángulos pertenecen a M_L:

```
VE= listaControlLod[L].verticesEliminados;
TG= listaControlLod[L].triangulosGenerados;
NT= TG-(listaVertices[VE].segundoVerticeAbanico- VE);
```

Algoritmo

Dada una aproximación, para cada vértice no eliminado (indicado en la lista de control) se pinta el abanico asociado a él y se cuentan los triángulos dibujados, para terminar cuando no queden más triángulos por pintar. De cada abanico se procesan sólo los t primeros vértices de los $t+1$ vértices de que consta, ya que cada uno de ellos referencia a uno de los t triángulos representados, mientras que el último vértice sirve solamente para completar el abanico. Al procesar cada uno de estos vértices puede ocurrir dos cosas:

```
para cada vértice v desde VE y mientras haya triángulos por pintar hacer
  interrupcionabanico <- cierto // fuerza caso a.1)
  para cada vértice v_i del abanico asociado a v, desde el primero
  hasta el penúltimo hacer
    si listaVerticesAbanico[v_i].id < TG entonces // caso a) pintar v_i
      si interrupcionabanico entonces
        pinta(v) // caso a.1) pinta vértice inicial
        interrupcionabanico <- falso // no repetir caso a.1)
      fin si
      pinta(v_i)
      NT <- NT-1
    sino // caso b) no visualizar v_i
      si no interrupcionabanico entonces
        // es necesario pintar v_i para cerrar triángulo
        pinta(v_i)
        interrupcionabanico <- cierto // fuerza caso a.1)
      fin si
    fin si
  fin para
  // último vértice v_i de la lista para cerrar abanico
  si no interrupcionabanico entonces
    pinta(v_i) // se pinta solamente si no hay interrupción
  fin si
fin para
```

Figura 7.6: Algoritmo de visualización.

a) El identificador del vértice indica que ese triángulo debe pintarse.

b) El identificador del vértice indica que ese triángulo no debe pintarse.

Cuando ocurre un caso b), significa que hay una interrupción en el abanico, de modo que si éste continúa más adelante es necesario insertar el vértice inicial. Por tanto, la primera vez que se cumple el caso a) después de una interrupción se debe realizar esta inserción (caso a.1 en el algoritmo).

7.5 Coste temporal

Dada una aproximación, M_L, se recorre la lista de vértices (bucle externo) y a su vez, para cada uno de ellos, su lista de vértices abanico asociada (bucle interno).

El coste temporal del algoritmo de extracción de la aproximación M_L depende de:

- Respecto al bucle externo, el número total de vértices no eliminados, que es como máximo $|V^r| - i$, donde i es el vértice a partir del cual se comienza a recorrer de forma secuencial la lista de vértices.
- Respecto al bucle interno, el número total de vértices abanico, que es como máximo $|A^r| - 2i$.

Como se vio en capítulos anteriores, en el peor caso $|V^r| - i$ es de orden cuadrático respecto a $|V_L|$. No se puede acotar $|A^r| - 2i$ por otro término menor que $|A^r|$ y como se vio al calcular el coste teórico de almacenamiento, en el peor caso $|A^r|$ es de orden cuadrático respecto a $|T|$.

En el caso medio (ver sección 4.3.1.2), $|V^r| - i$ es de orden lineal respecto a $|V_L|$ y $|A^r|$ es de orden lineal respecto a $|T|$, obteniendo un coste de $O(|V_L| + |T|)$.

7.6 Resultados

Los experimentos han sido llevados a cabo utilizando el entorno de trabajo descrito en la sección 4.4.

En la tabla 7.2 se resumen las características y los costes de almacenamiento de los objetos utilizados en los experimentos. Para cada uno de ellos se indica el número de vértices y triángulos del modelo original, y su coste de almacenamiento suponiendo una estructura basada en una lista de vértices y una lista de triángulos [Foley et al., 1990]. También se ha asumido que el coste de una palabra (entero, real o puntero) es de 4

	Original			MOM			MOM-Fan	
	Vértices	Triáng.	MB	n	Triáng.	MB	V. Aba.	MB
Cow	2.905	5.804	0'100	2.803	14.982	0'237	17.852	0'202
Sphere	15.315	30.624	0'526	15.264	83.486	1'306	98.793	1'104
Bunny	34.835	69.451	1'193	33.990	182.192	2'876	216.759	2'445
Phone	83.045	165.963	2'850	81.668	441.181	6'940	523.844	5'887
Isis	187.871	375.736	6'450	187.370	993.559	15'667	1.181.373	13'309
Buda	543.653	1.085.636	18'646	543.106	2.754.083	43'957	3.297.593	37'598

Tabla 7.2: Costes de almacenamiento.

bytes. Respecto a la representación multirresolución MOM, se indica el número de aproximaciones, el número de triángulos y el coste total de almacenamiento. Respecto a la representación MOM-Fan, se indica el número de vértices abanico y el nuevo coste de almacenamiento.

Se puede observar que el número de vértices abanico almacenados en la nueva representación es mayor que el de triángulos de la representación MOM. Sin embargo, dado que el coste de almacenar un vértice abanico es menor que el de un triángulo, el coste de almacenamiento de la nueva lista supone un ahorro de memoria alrededor del 20% respecto de la lista de triángulos. La repercusión de esta reducción sobre el coste total del modelo es muy importante, aproximadamente el 15%, debido a que precisamente se ha reducido el coste de la lista que mayor peso tiene en el modelo (compárese el número de triángulos con el de vértices o aproximaciones). Esta reducción es muy similar a la obtenida en el estudio teórico que era del 17%.

Los experimentos realizados han consistido en medir el tiempo que el algoritmo de visualización invierte en obtener y visualizar los datos de una aproximación dada. Esto difiere de los experimentos que en capítulos anteriores se realizaron y que sólo medían el tiempo de extracción. El motivo por el que ahora se ha añadido el tiempo de visualización es porque uno de los dos principales objetivos que se persigue con el uso de los abanicos de triángulos es acelerar la visualización disminuyendo el tiempo de dibujo y no el tiempo de extracción.

En la figura 7.7(a) se muestran los comportamientos de las nuevas representaciones utilizando los modelos *Bunny* y *Buda* (algunas imágenes de estos modelos se muestran en las figuras 7.8 y 7.9). El eje X representa el nivel de detalle, donde 0 se corresponde con el menor y 1 con el mayor. El eje Y representa el tiempo (en segundos) invertido por el modelo en obtener los datos que pertenecen a una aproximación dada y visualizarlos. El comportamiento es prácticamente el mismo que el que se obtenía con el esquema previo de MOM y se puede observar la respuesta lineal respecto al número de triángulos de la aproximación visualizada.

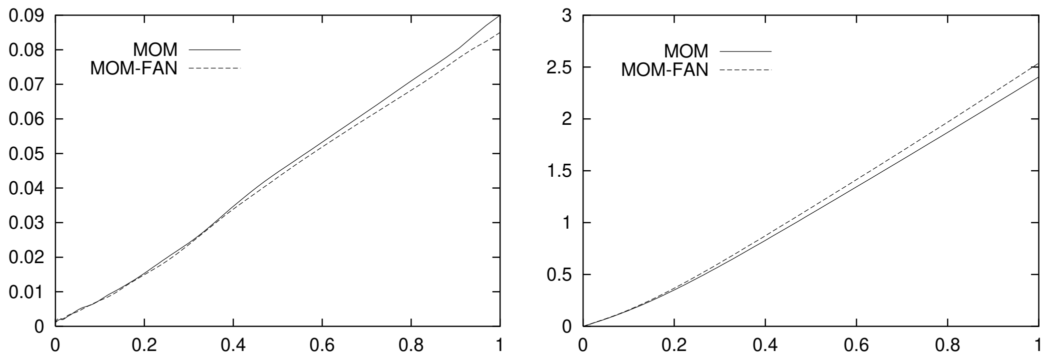
El uso de abanicos supone una disminución en el número de vértices que se envían por nivel de detalle al sistema gráfico. Este hecho queda reflejado en la figura 7.7(b) donde para ambos modelos se puede observar que la representación MOM-Fan envía un 40% menos de vértices que la representación MOM.

Sin embargo, la mejora que supone utilizar abanicos de triángulos se ve mermada por el ligero aumento en el proceso de recuperación de los datos debido a una mayor complejidad del algoritmo, la corta longitud de los abanicos (3'3 triángulos de valor medio por abanico), la sobrecarga que supone el ejecutar para cada abanico la pareja de instrucciones *glBegin* y *glEnd* en la implementación realizada con *OpenGL*, y los triángulos degenerados. En la figura 7.7(c) se representan el número de triángulos no degenerados y el número de triángulos degenerados enviados al sistema gráfico por aproximación, alrededor de un 23% del total de triángulos no degenerados.

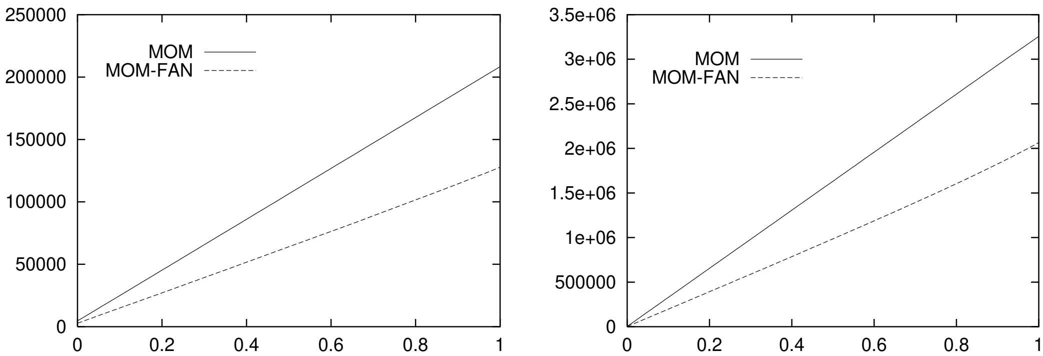
7.7 Conclusiones

En este capítulo se ha presentado una nueva extensión al esquema básico de MOM que permite almacenar y extraer las distintas aproximaciones como abanicos de triángulos. El objetivo que se ha perseguido es doble, disminuir el tiempo de visualización y reducir el coste de almacenamiento. Los resultados experimentales muestran una reducción de un 15% del coste de almacenamiento respecto a la representación básica de MOM.

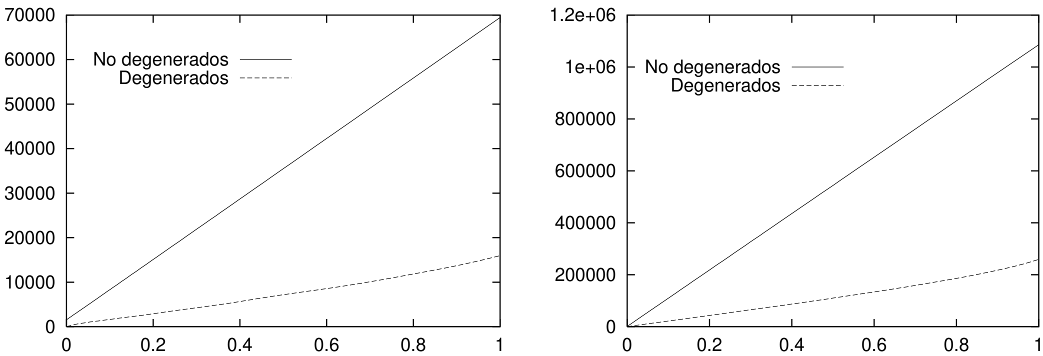
Sin embargo, el comportamiento de la nueva representación multirresolución respecto al tiempo de visualización es similar al de su antecesor. Una longitud media de los abanicos corta, un porcentaje alto de triángulos degenerados y el adecuar los abanicos al nivel de detalle requerido en tiempo de ejecución, hace que no se obtenga una mejora global en el tiempo de visualización del modelo.



(a) Tiempos de visualización (en segundos)



(b) Vértices enviados en MOM y MOM-Fan



(c) Triángulos degenerados y no degenerados

Figura 7.7: Resultados experimentales obtenidos con los modelos *Bunny* (gráficas de la izquierda) y *Buda* (gráficas de la derecha).

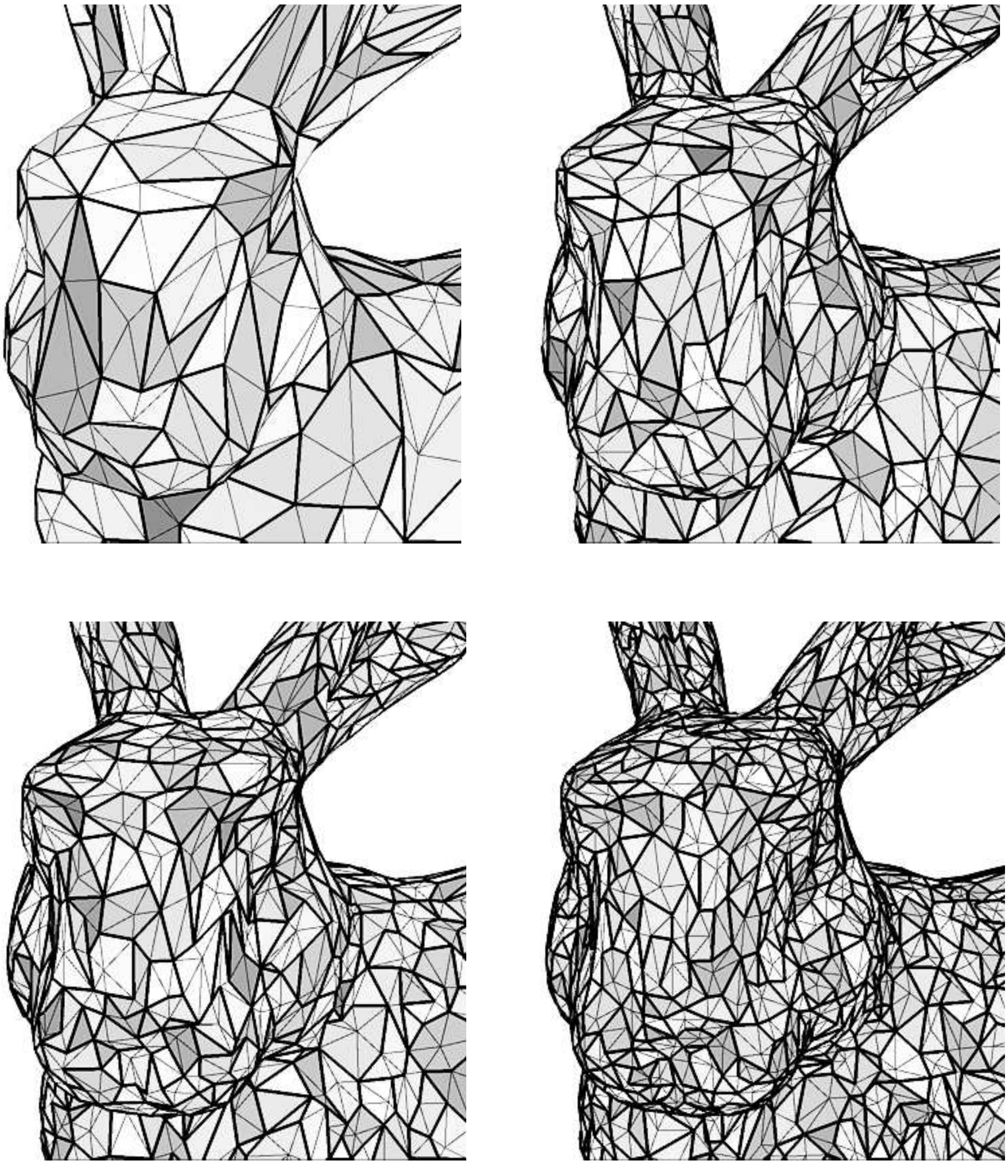


Figura 7.8: Cuatro aprox. del modelo *Bunny* visualizados utilizando abanicos.

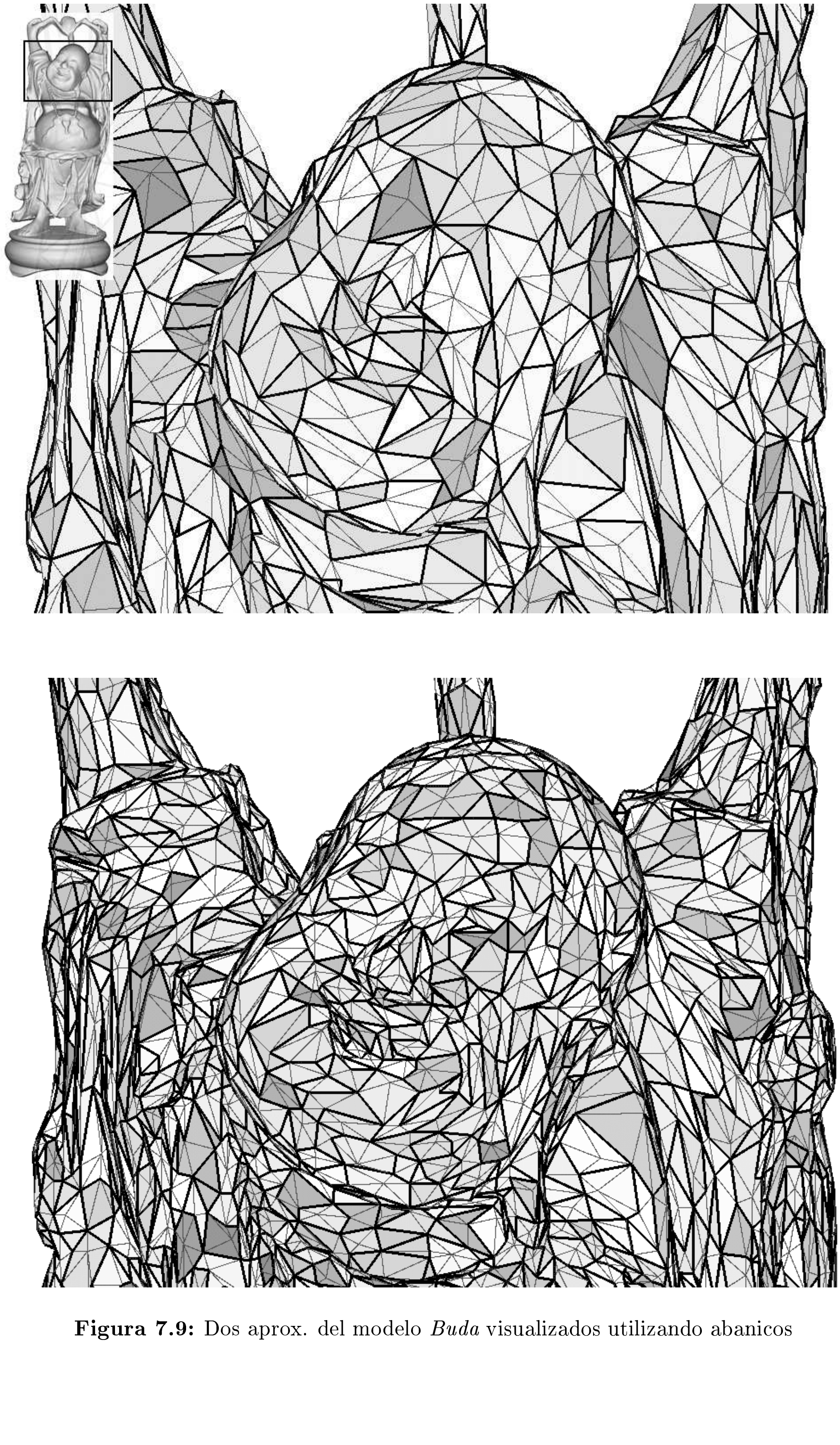


Figura 7.9: Dos aprox. del modelo *Buda* visualizados utilizando abanicos

Capítulo 8

Nivel de Detalle Variable

En este capítulo se presenta un algoritmo capaz de extraer aproximaciones donde el nivel de detalle varía a lo largo de la superficie del objeto. En general, los modelos multirresolución que permiten la extracción de aproximaciones con nivel de detalle variable se basan en estructuras de datos jerárquicas de tipo árbol o grafo [Xia y Varshney, 1996, Hoppe, 1997, Luebke y Erikson, 1997, Floriani et al., 1998]. Por ejemplo, Hoppe cambió la representación incremental [Hoppe, 1996] de Progressive Meshes a una representación jerárquica [Hoppe, 1997] para poder extraer aproximaciones de nivel de detalle variable. Por otra parte, tal y como se comentó en la sección 2.5.3, extraer este tipo de aproximaciones puede producir problemas de plegamientos en la malla sino se tienen en cuenta las posibles dependencias entre las distintas triangulaciones. Las listas de dependencias fueron introducidas por Xia et al. [Xia et al., 1997] y mejoradas posteriormente por Hoppe [Hoppe, 1997] 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 [El-Sana y Varshney, 1999]. La principal característica del algoritmo que se va a presentar es que ni necesita que la información almacenada en un objeto MOM sea reorganizada en una nueva estructura de datos, ni necesita almacenar nueva información de dependencias entre las distintas actualizaciones. La estructura de datos básica presentada en el capítulo 4 contiene la información suficiente.

Extraer una representación con nivel de detalle variable exige, tal y como se comentó en el capítulo 2, de un criterio o conjunto de criterios que decidan qué parte del objeto simplificar y qué parte mantener con detalle. El algoritmo que se propone es independiente de los criterios que se decida utilizar y pueden ser fácilmente añadidos. Por ejemplo, en la figura 8.1 el usuario ha seleccionado de forma interactiva la zona a representar con mayor detalle (zona de la esfera). Sólo esa zona se dibuja con la aproximación de mayor detalle. El resto del objeto se pintará con el menor detalle posible manteniendo siempre la continuidad en la malla. Así, aparece un conjunto de triángulos que pertenecen a la aproximación de menor detalle (triángulos de color gris oscuro) y otro conjunto de triángulos (triángulos de color gris medio) que realizan la transición entre los de mayor y los de menor detalle, asegurando dicha continuidad.

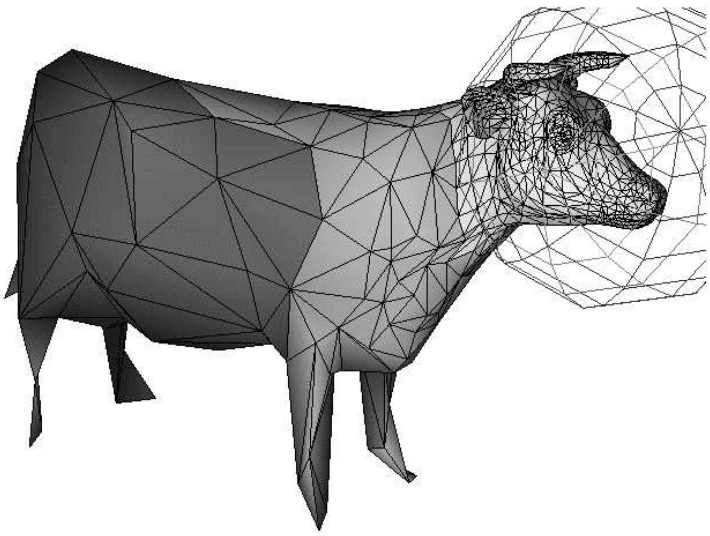


Figura 8.1: Ejemplo de aproximación con nivel de detalle variable.

8.1 Algoritmo de visualización

8.1.1 Simplificación selectiva

En el capítulo 5 se comentó que los algoritmos de extracción de aproximaciones se podían dividir según partan de la aproximación de menor detalle o de la aproximación de mayor detalle. Los modelos multirresolución que extraen aproximaciones con nivel de detalle variable también pueden clasificarse de la misma forma y, en concreto, los modelos que se muestran en la tabla 2.3 definen algoritmos basados en refinamiento selectivo, es decir, parten de la aproximación de menor detalle y refinan (incrementan detalle) allí donde es requerido. El algoritmo que se presenta en este capítulo, a diferencia de éstos y del propio algoritmo de extracción de aproximaciones de nivel de detalle uniforme de MOM, se basa en lo que se denomina simplificación selectiva, es decir, parte de la aproximación de mayor detalle y lo elimina de donde no haga falta preservarlo.

Aparentemente, este método puede parecer menos eficiente que los basados en refinamiento selectivo debido a que, probablemente, la malla recuperada estará más cerca de la aproximación de menor detalle que de la de mayor. Sin embargo, la simplicidad de la estructura de datos, con una disposición ordenada de la información, y un sencillo algoritmo de recorrido, permiten obtener resultados similares e incluso, en algunos casos, mejores que los obtenidos por algoritmos basados en refinamiento.

8.1.2 Funcionamiento básico

El funcionamiento básico de este nuevo algoritmo muestra claras diferencias con el de extracción de una aproximación de nivel de detalle uniforme explicado en el capítulo 5, aunque ambos recorran la misma estructura de datos.

Para recuperar una aproximación de nivel de detalle uniforme, M_L, se calcula, en primer lugar, el número de triángulos que forman M_L. Después, se determina a partir de qué vértice recorrer la estructura de datos y, por último, se recorre la estructura visualizando aquellos triángulos cuyo campo id sea menor que el número de triángulos generados establecido para M_L.

Ahora, para recuperar una aproximación con nivel de detalle variable:

1. es imposible conocer de antemano el número de triángulos que van a pertenecer a la solución;
2. en principio, todos los triángulos almacenados en la representación MOM son candidatos a pertenecer a la solución;
3. este hecho obliga a recorrer la estructura de datos siempre desde su primer vértice y hasta el último de ellos que tenga, al menos, un triángulo asociado.

El algoritmo que se propone recorre secuencialmente la estructura de datos y decide si ha de realizar o no cada una de las simplificaciones almacenadas S_i [1], $0 \leq i < n-1$. El algoritmo utilizará el criterio definido por la aplicación para tomar esa decisión. Sin embargo, este criterio no se evaluará para todas las simplificaciones. Ya que el método parte de la aproximación de mayor detalle, M_0, y elimina detalle donde no hay que preservarlo, sólo será necesario evaluar el criterio en aquellas simplificaciones que contengan algún triángulo de T_0. Por tanto, el criterio se evaluará para cada simplificación S_i si y sólo si existe al menos un triángulo t tal que $t \in \mathcal{T}_i$ y $t \in T_0$. En otro caso no será necesario evaluarlo.

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 8.2, 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 $R = R \cup \mathcal{T}'_i$ 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 $t \in R$.

[1] Como se vio en la sección 3.2, cada simplificación se puede representar como 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 añadidos a M_i para generar, finalmente, M_{i+1}.

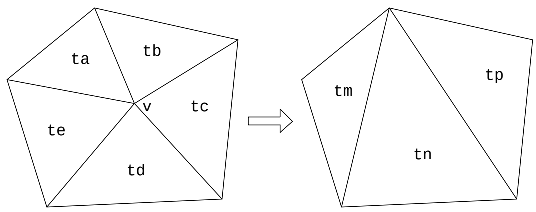


Figura 8.2: Los triángulos *tm, tn y tp* no pueden aparecer en la solución si el vértice v no es eliminado previamente ($\mathcal{V}_i = \{v\}, \mathcal{T}_i = \{ta, tb, tc, td, te\}, \mathcal{V}_i' = \{\}, \mathcal{T}_i' = \{tm, tn, tp\}$).

Para codificar el conjunto R se va a añadir una lista de marcas, de longitud $|T^r|$ bits, que permita señalar aquellos triángulos que pertenecen a R. Esta lista debe inicializarse cada vez que se extraiga una nueva aproximación.

8.1.3 Algoritmo

El algoritmo de visualización de una aproximación con nivel de detalle variable se muestra en la figura 8.3. 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.

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.

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

// PRIMERA ETAPA
// para cada simplificación almacenada en la lista de control
para cada simplificación S_i ∈ [S_0..S_{n-2}] hacer
  si test(S_i)= CIERTO entonces
    // marcar las caras que no se generan
    ∀t ∈ T'_i flag[t]= NO_VISUALIZABLE
    // para cada vertice a ser eliminado
    para cada vértice v ∈ V_i 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 ∈ M_{n-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 8.3: Algoritmo de visualización de una aproximación con nivel de detalle variable.

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.

8.1.4 Ejemplo

En la figura 8.1 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.

Algunos ejemplos de aproximaciones con nivel de detalle variable obtenidos con el algoritmo de la figura 8.3 se pueden observar en las figuras 8.4 y 8.5.

8.2 Coste temporal

Para analizar el coste temporal del algoritmo de visualización (figura 8.3) se van a estudiar por separado cada una de las dos etapas de las que consta el algoritmo y que se detallan en la sección anterior.

La segunda etapa simplemente recorre la lista de vértices pertenecientes a la aproximación de peor nivel de detalle, esto es, $|V_{n-1}|$ vértices, y sus triángulos asociados, que son $|T_{n-1}|$ triángulos. Por tanto, su coste es $O(|V_{n-1}| + |T_{n-1}|)$.

La primera etapa recorre la lista de simplificaciones y, en función de un cierto criterio, realiza o no una serie de operaciones para cada simplificación. En primer lugar, se va a evaluar el coste del test que se realiza para cada simplificación, y a continuación, el coste del algoritmo en el peor caso.

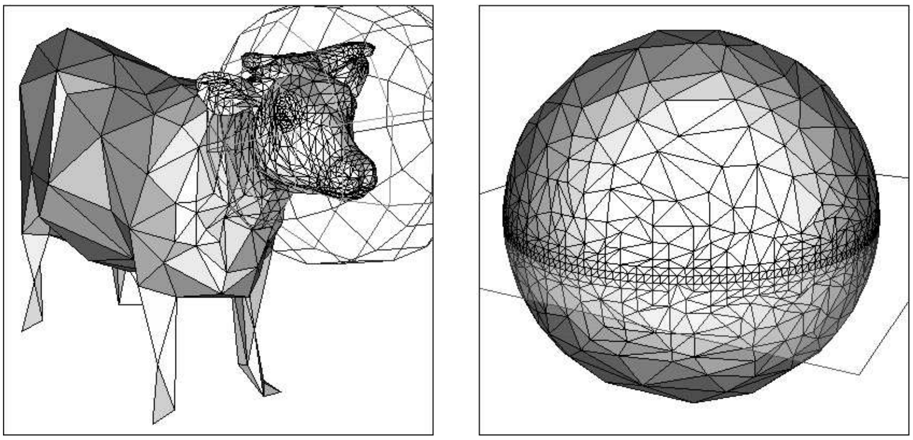


Figura 8.4: Ejemplos de aproximaciones con nivel de detalle variable. A la izquierda, el usuario ha seleccionado con una esfera la parte donde preservar el detalle. A la derecha, la intersección de un plano con una esfera se representa con el máximo nivel de detalle.

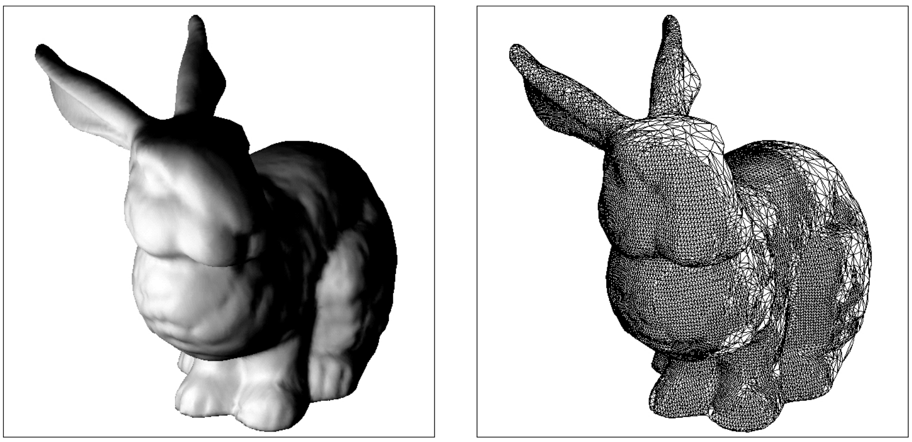


Figura 8.5: Ejemplos de aproximaciones con nivel de detalle variable donde la zona del objeto iluminada se representa con el máximo detalle. A la izquierda se muestra la visualización del objeto y a la derecha la malla generada.

Coste temporal de la operación *test()*

Realizar la operación *test()* sobre una simplificación S_i consiste en evaluar un criterio sobre la misma, y en función del resultado, realizar una serie de comprobaciones sobre los conjuntos de triángulos $\mathcal{T}_i$ y $\mathcal{T}'_i$.

En el peor caso, el coste de realizar la operación *test()* sobre una simplificación S_i es $O(K+|\mathcal{T}_i|+|\mathcal{T}'_i|)$, donde K es el coste de aplicar el criterio, que puede ser constante o no. Por ejemplo, podría ser del orden del número de vértices eliminados en la simplificación, $|\mathcal{V}_i|$. El coste de realizar dicha operación sobre todas las simplificaciones sería en el caso peor:

$$\sum_{i=0}^{n-2} (K + |\mathcal{T}_i| + |\mathcal{T}'_i|) < nK + 2|T^r| \tag{8.1}$$

Este sumatorio es $O(nK + 2|T^r|)$. Suponiendo que el coste del criterio es constante para una simplificación, se puede despreciar el primer término por ser de orden menor, obteniendo que el coste de realizar todos los tests es $O(|T^r|)$.

Caso peor

El caso peor se da cuando la operación *test()* da cierto como resultado para todas las simplificaciones. En este caso, para calcular el coste de la primera etapa se tiene que tener en cuenta el coste de cuatro operaciones:

- Ejecutar *test()* para todas las simplificaciones: $O(nK + 2|T^r|)$.
- Marcar los triángulos de $\mathcal{T}'_i$ de todas las simplificaciones como no visualizables:
$$\sum_{i=0}^{n-2} |\mathcal{T}'_i| = |T^r| - |T_{n-1}|$$
- Recorrer la lista de vértices $\mathcal{V}_i$ de todas las simplificaciones:
$$\sum_{i=0}^{n-2} |\mathcal{V}_i| = |V^r| - |V_{n-1}|$$
- Recorrer la lista completa de triángulos asociados a todos los vértices de $\mathcal{V}_i$ de todas las simplificaciones: esto es, recorrer toda la lista de triángulos T^r, excepto los triángulos asociados a la aproximación de peor nivel de detalle, $|T^r| - |T_{n-1}|$.

Sumando los costes de la primera y segunda etapa obtenemos que el coste es $O(nK + |T^r|+|V^r|)$. Como se vio en la sección 4.3.1.1, $|T^r|$ y $|V^r|$ son, en el caso peor, cuadráticos respecto al tamaño de la malla original, $|T|$ y $|V|$, respectivamente, pero son de orden lineal respecto a $|T|$ y $|V|$ en el caso medio.

8.3 Mejora de las prestaciones

El algoritmo propuesto en la sección 8.1, y que de aquí en adelante llamaremos algoritmo de fuerza bruta, puede presentar un comportamiento poco eficiente 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 cada simplificación ya que probablemente ésta sólo se evaluará a cierto para un número muy reducido de casos.
2. Recorrido de toda la secuencia de simplificaciones. Esta tarea es realmente costosa y son muy pocas las simplificaciones que a fin de cuentas van a tener triángulos que pertenezcan a la solución.

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 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.

8.3.1 Agrupamiento

Se propone aplicar un método de agrupamiento para generar una jerarquía de sólo dos niveles de profundidad. Cada agrupación reunirá las simplificaciones que actúan sobre la misma zona de la superficie del modelo (figura 8.6). El propio método de simplificación utilizado en la construcción del modelo multirresolución puede servir para generar las agrupaciones.

Cada agrupación contendrá la información necesaria de forma que el test de refinamiento pueda aplicarse directamente sobre él. Las simplificaciones reunidas en una agrupación serán chequeadas sólo cuando el resultado del test indique que haya que hacerlo. Es evidente que el éxito de esta actuación dependerá del número de agrupaciones en las que haya que chequear sus simplificaciones. Si este número es pequeño, el número de veces que se aplicará el test se reducirá en un alto porcentaje.

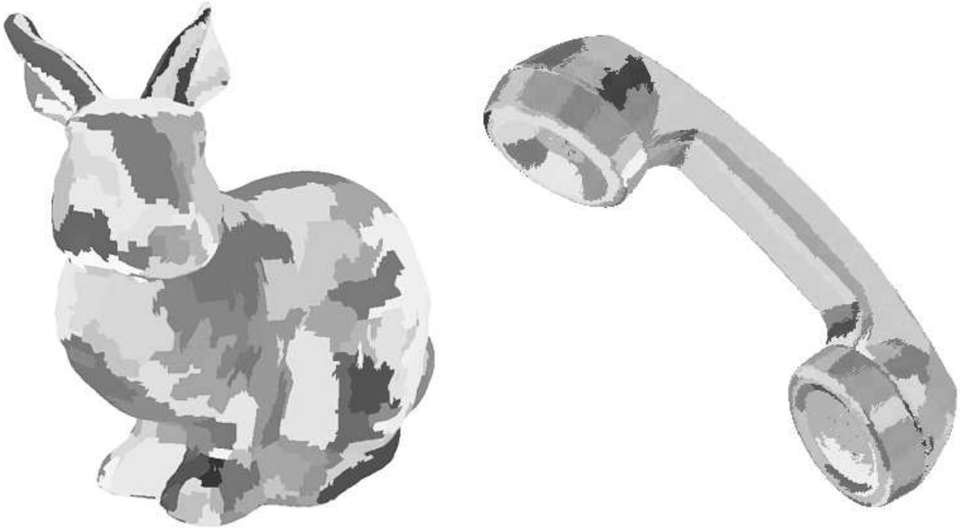


Figura 8.6: Agrupaciones de los modelos *Bunny* (izquierda) y *Phone* (derecha). El modelo *Bunny* consta de 264 agrupaciones mientras que el modelo *Phone* consta de 408. Cada agrupación ha sido dibujada de un color diferente.

Para implementar la jerarquía es necesario ampliar la estructura de datos creando una nueva lista que almacene la secuencia de clusters. Cada cluster almacenará una lista de simplificaciones así como la información necesaria para poderle aplicar el test de refinamiento.

8.3.2 Lista dinámica

El objetivo es reducir el número de simplificaciones a recorrer manteniendo una lista dinámica formada por 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. La eficiencia de esta actuación dependerá únicamente del número de simplificaciones que formen la nueva lista.

Para poder generar la lista dinámica es necesario que se amplíe la estructura de datos de forma que cada triángulo almacenado en M^r conozca el vértice al que está asociado. Esta ampliación coincide con una de las que se propusieron en el capítulo 6 para que una representación MOM pudiera explotar la propiedad de la coherencia cuadro a cuadro.

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, también han de añadirse a la lista. Al final 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 cuadro, o actualizarse aplicando coherencia cuadro a cuadro. 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 cuadro. Después se añadirán las nuevas simplificaciones de la misma forma que se ha explicado en el párrafo anterior.

8.4 Resultados

Los experimentos han sido llevados a cabo utilizando el entorno de trabajo descrito en la sección 4.4.

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 8.1. El segundo es el algoritmo de fuerza bruta al que se le han añadido las agrupaciones y la lista dinámica, y que llamaré algoritmo de *Agrupamiento*. El tercero es el resultado de explotar la coherencia cuadro a cuadro en el algoritmo de agrupamiento y que llamaré 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 8.7 y 8.10 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 cuadros por segundo que se consiguen. También se ha incluido una curva, denominada *Uniforme*, con el número de cuadros que se obtienen cuando se visualiza el modelo original. Las figuras 8.8 y 8.11 son detalles de las figuras anteriores, respectivamente, donde se pueden observar con mayor claridad los resultados obtenidos. Las tablas 8.1 y 8.2 recogen las prestaciones obtenidas por los tres algoritmos para tres aproximaciones concretas. En las figuras 8.9 y 8.12 se muestran imágenes de dichas aproximaciones.

Los comentarios que se pueden hacer a raíz de los experimentos realizados son los mismos para los dos objetos. 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. Respecto a los tres algoritmos presentados, el algoritmo de fuerza bruta es el que menos mejora produce aunque tiene la ventaja de que no necesita ampliar la estructura de datos ni reorganizar ningún tipo de información.

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.

8.4.1 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 8.13 se muestra el número de triángulos obtenidos en cada cuadro para MOM y MT utilizando el modelo *Bunny*. El eje X representa el cuadro, y el eje Y el número de triángulos. En la figura 8.14 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 cuadro, 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.

8.5 Conclusiones

En este capítulo se ha presentado un algoritmo que permite la extracción de aproximaciones con nivel de detalle variable recorriendo la misma estructura de datos presentada en el capítulo 4. No hay necesidad de reorganizar los datos en una nueva estructura, ni de almacenar explícitamente ningún nuevo tipo de dependencia entre las distintas actualizaciones del objeto representado. La propia ordenación de los datos codifica de forma implícita la suficiente información para extraer correctamente una aproximación con nivel de detalle variable. Consecuentemente, el mismo proceso de construcción crea un modelo del cual se pueden extraer aproximaciones de nivel de detalle tanto uniforme como variable, no siendo necesario un proceso adicional que podría incrementar el coste del proceso de construcción así como su complejidad.

Junto a este algoritmo, al que se ha denominado algoritmo de fuerza bruta, se han presentado una serie de mejoras que permiten aumentar la eficiencia y obtener mejores prestaciones que las conseguidas por el modelo multirresolución MT. Los resultados también muestran cómo los algoritmos propuestos incrementan de forma importante la tasa de cuadros por segundo cuando la zona refinada es relativamente pequeña respecto al resto del modelo, pero también cómo caen sus prestaciones a medida que la zona refinada aumenta de tamaño.

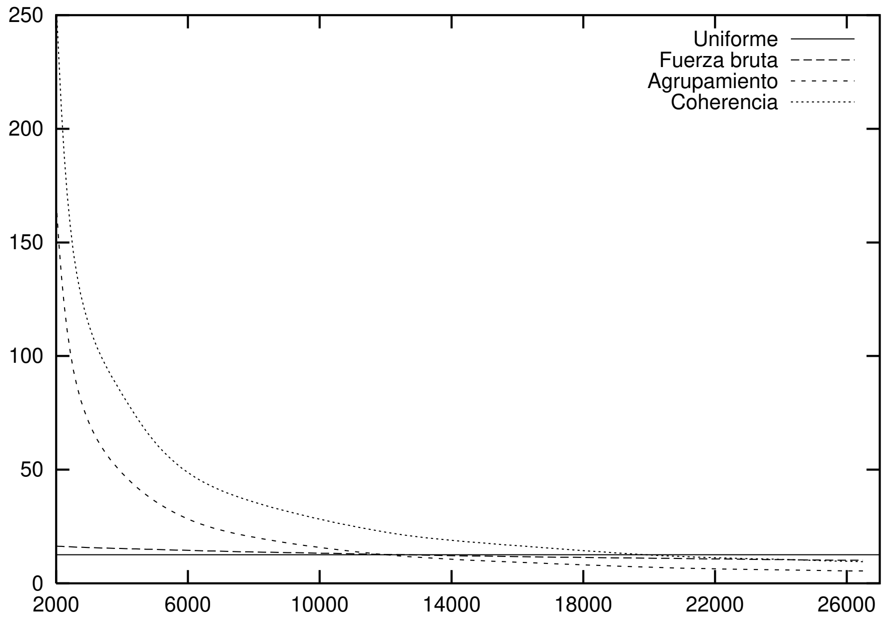


Figura 8.7: Comportamiento de los algoritmos de visualización para el modelo *Bunny*.

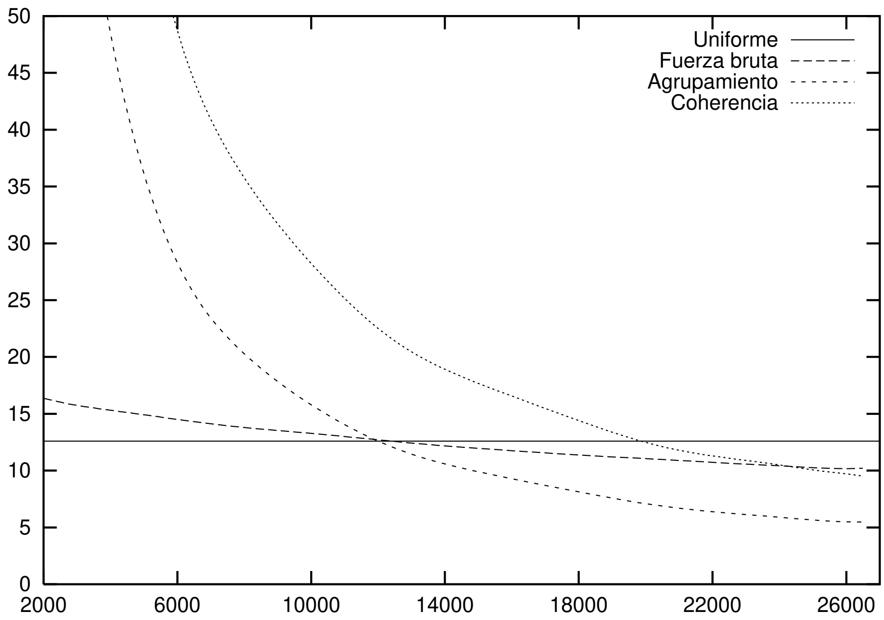
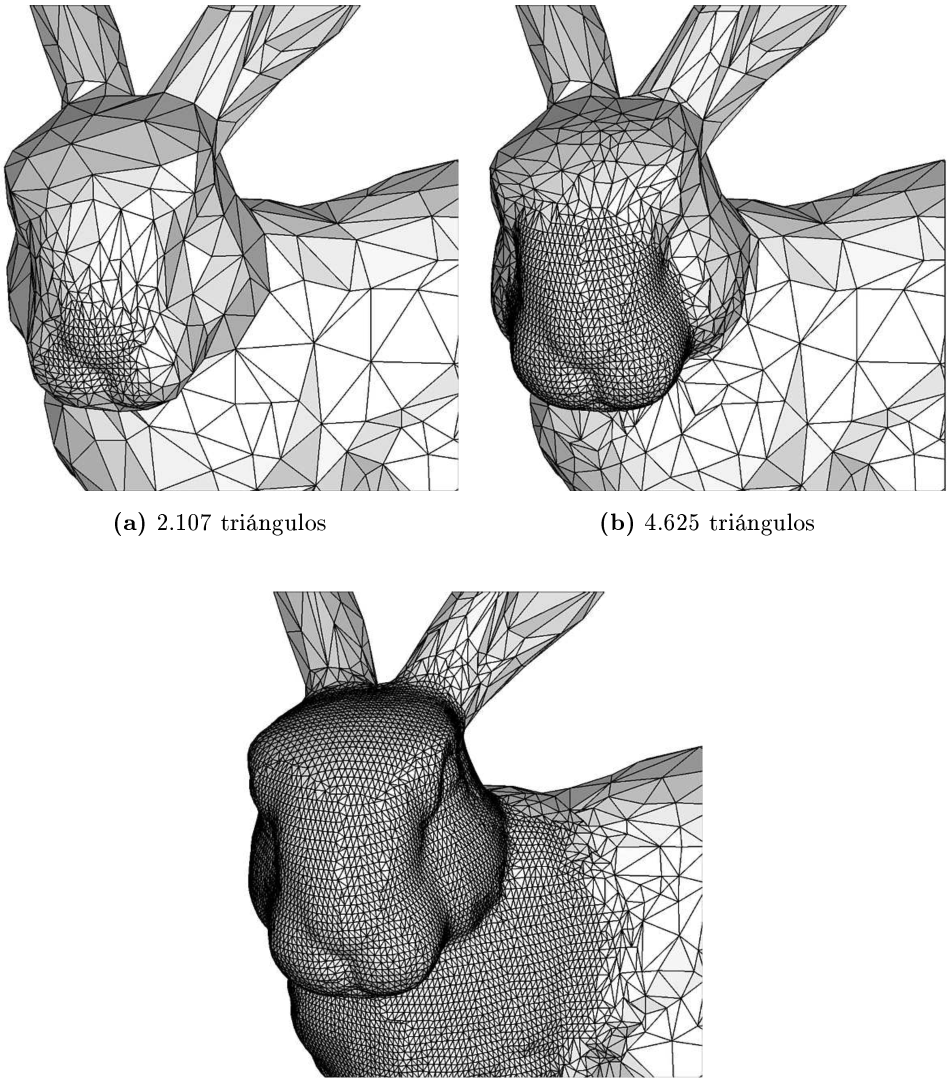


Figura 8.8: Detalle de la gráfica de la figura 8.7.

#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

Tabla 8.1: Cuadros por segundo obtenidos para tres aproximaciones diferentes del modelo *Bunny*.



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

Figura 8.9: Imágenes correspondientes a las tres aproximaciones del modelo *Bunny* de la tabla 8.1

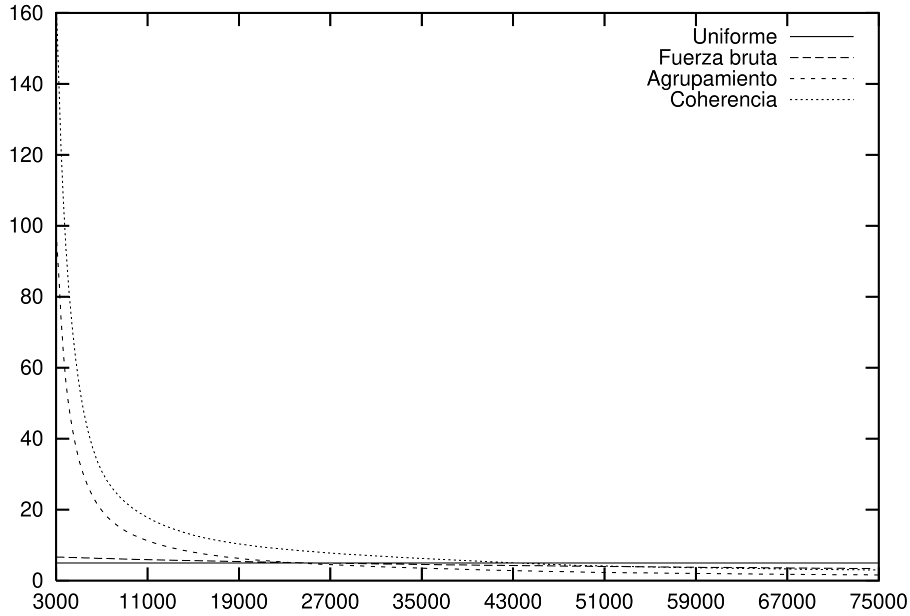


Figura 8.10: Comportamiento de los algoritmos de visualización para el modelo *Phone*.

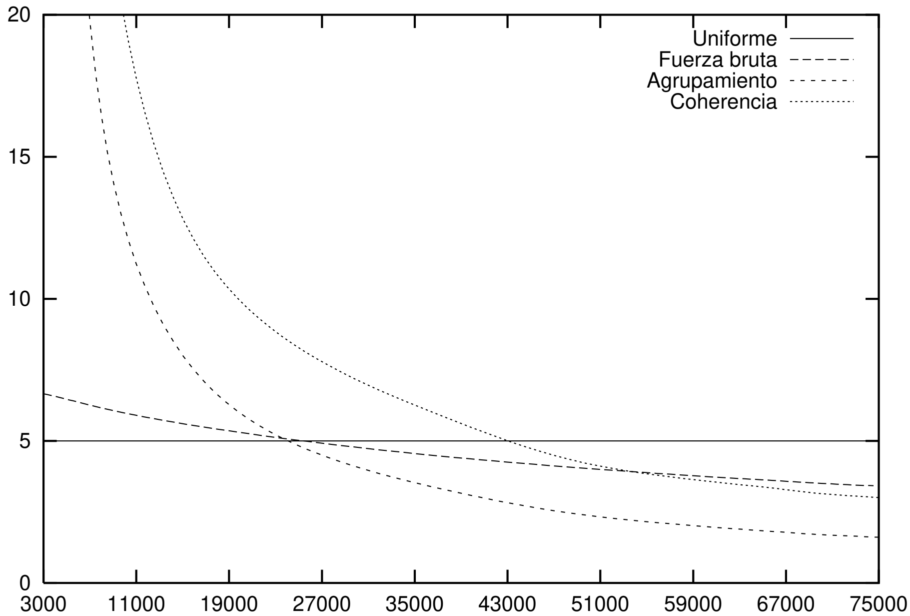
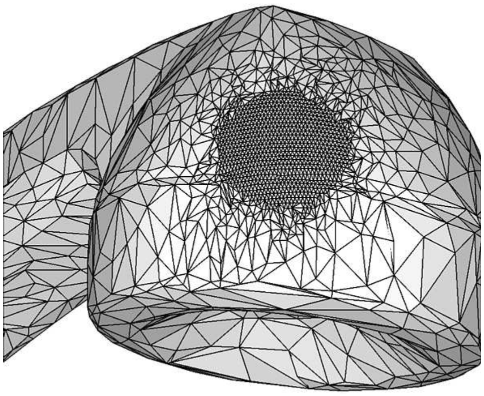


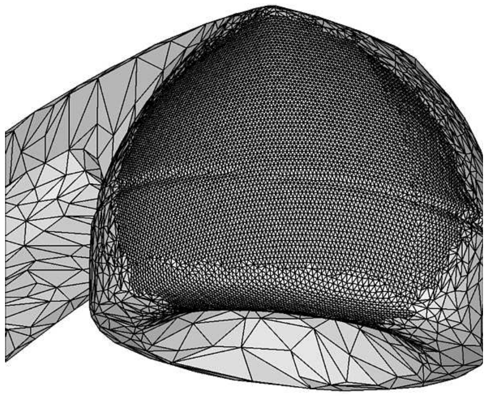
Figura 8.11: Detalle de la gráfica de la figura 8.10.

#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

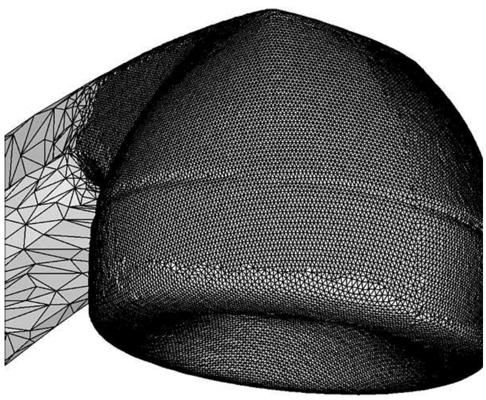
Tabla 8.2: Cuadros por segundo obtenidos para tres aproximaciones diferentes del modelo *Phone*.



(a) 4.794 triángulos



(b) 17.464 triángulos



(c) 62.132 triángulos

Figura 8.12: Imágenes correspondientes a las tres aproximaciones del modelo *Phone* de la tabla 8.2

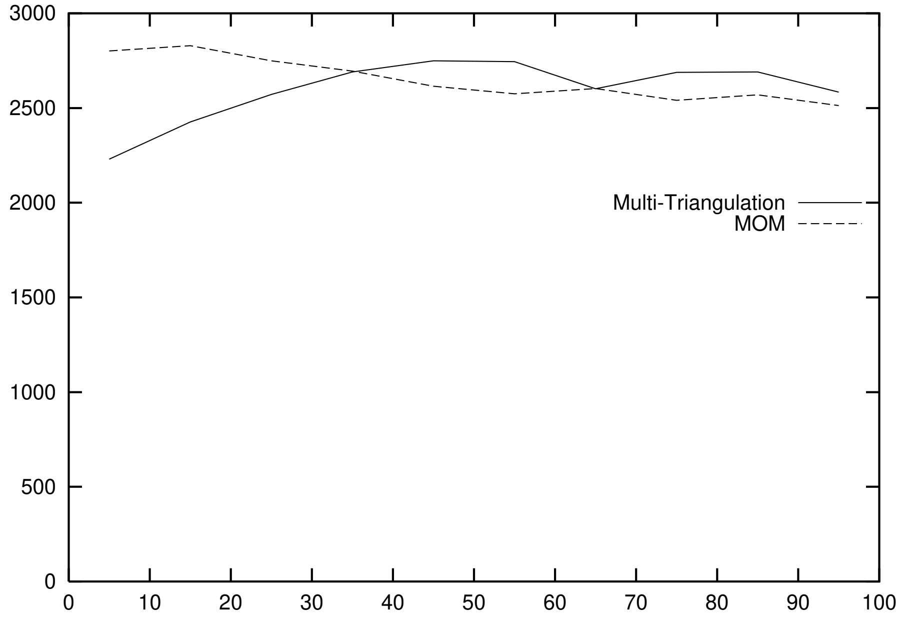


Figura 8.13: Triángulos extraídos para el modelo *Bunny*.

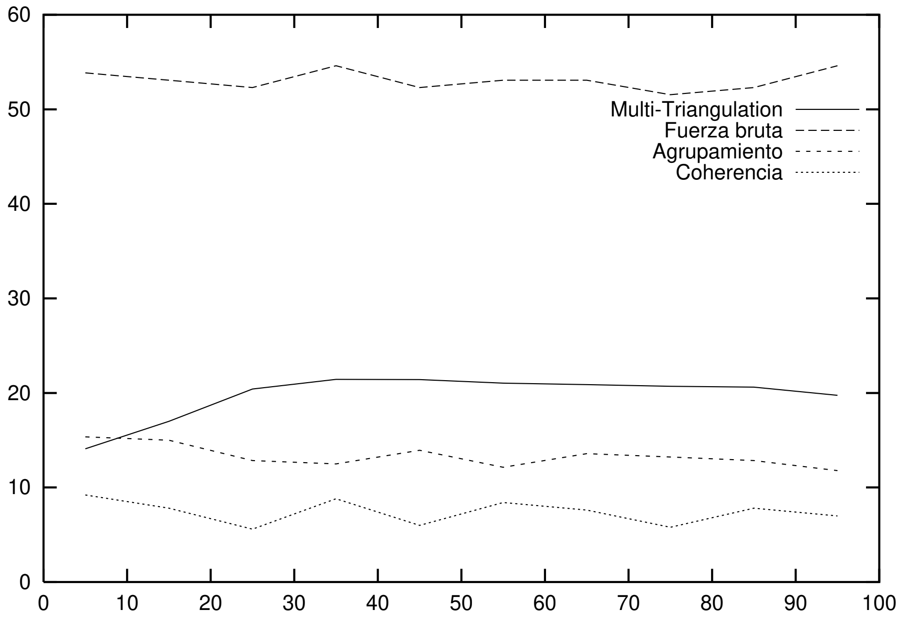


Figura 8.14: Tiempos de extracción (ms.) para el modelo *Bunny*.

Capítulo 9

Conclusiones y Líneas de Trabajo Futuro

En esta tesis, el esfuerzo realizado se ha dirigido hacia resolver el problema de la visualización interactiva de modelos poligonales complejos. El objetivo ha consistido en desarrollar un nuevo modelo 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.

Este nuevo modelo multirresolución permite, ante una petición, recuperar 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.

9.1 Principales aportaciones

Mallas Ordenadas Multirresolución 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. El modelo consta de una estructura de datos básica que permite almacenar las múltiples aproximaciones de un objeto y de la cual recuperar tanto aproximaciones de resolución uniforme como variable. Además, se han realizado una serie de extensiones que permiten explotar la propiedad de la coherencia cuadro a cuadro y el uso de abanicos de triángulos con el fin de aumentar aún más las prestaciones y mejorar el comportamiento general del modelo.

A continuación se resumen las principales aportaciones del trabajo presentado en esta tesis:

- Una estructura de datos que permite un coste de almacenamiento menor que el de los modelos multirresolución de características similares (MT y HDS) y es comparable a modelos que carecen de la característica de la generalidad (PM).
- Un algoritmo de extracción de aproximaciones con nivel de detalle uniforme mucho más eficiente que los modelos multirresolución de características similares. La comparativa 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.
- Una extensión al modelo básico que aprovecha la propiedad de la coherencia cuadro a cuadro. El nuevo algoritmo es más eficiente cuando las aproximaciones requeridas en cuadros consecutivos es pequeña y el nivel de detalle es alto. Como contrapartida, el coste de almacenamiento se incrementa alrededor de un 29%.
- Una extensión que utiliza el abanico de triángulos como primitiva básica de representación. Los resultados experimentales muestran una reducción de un 15% del coste de almacenamiento pero no una mejora en el tiempo de visualización debido a una longitud media de los abanicos corta, un porcentaje alto de triángulos degenerados y el adecuar los abanicos al nivel de detalle requerido en tiempo de ejecución.
- Un algoritmo que permite extraer aproximaciones con nivel de detalle variable, sin necesidad de reorganizar los datos en una nueva estructura, ni de almacenar explícitamente ningún nuevo tipo de dependencia entre las distintas actualizaciones del objeto representado. Junto a este algoritmo, al que se ha denominado algoritmo de fuerza bruta, se han presentado una serie de mejoras que permiten aumentar la eficiencia y obtener mejores prestaciones que las conseguidas por el modelo multirresolución MT.

9.2 Publicaciones directamente derivadas de esta tesis

Las publicaciones originadas hasta el momento por el trabajo que se ha presentado en esta tesis doctoral son las siguientes:

"Un Modelo Multirresolución para Visualización Interactiva"
J. Ribelles, M. Chover, J. Huerta, R. Quirós.

Actas del VIII Congreso Español de Informática Gráfica (CEIG'98), 267-280, Ourense (Spain), Junio 1998.

"Multiresolution Ordered Meshes"
J. Ribelles, M. Chover, J. Huerta, R. Quirós.
Proc. of 1998 IEEE Conference on Information Visualization (IV'98), 198-204, London (UK), Julio 1998.

"Frame-to-frame Coherence of Multiresolution Ordered Meshes"
J. Ribelles, M. Chover, A. López, J. Huerta.
Actas del IX Congreso Español de Informática Gráfica (CEIG'99), 91-104, Jaén (Spain), Junio, 1999.

"Un Servidor de Geometría 3D"
R. Vidal, J. Ribelles.
Actas del IX Congreso Español de Informática Gráfica (CEIG'99), 355-356, Jaén (Spain), Junio, 1999.

"A First Step to Evaluate and Compare Multiresolution Models"
J. Ribelles, M. Chover, A. López, J. Huerta.
Informe técnico, Departamento de Informática, Universitat Jaume I, DI 01-07/99, Castellón (Spain), Julio, 1999
Short Papers and Demos of EUROGRAPHICS'99, 230-232, Milano (Italy), Septiembre, 1999.

"Representación de Modelos Multirresolución con Abanicos de Triángulos"
J. Ribelles, A. López, I. Remolar, O. Belmonte, M. Chover.
Actas del X Congreso Español de Informática Gráfica (CEIG 2000), 37-50, Castellón (Spain), Junio, 2000.

"Multiresolution Modelling of Polygonal Surface Meshes Using Triangle Fans"
J. Ribelles, A. López, I. Remolar, O. Belmonte, M. Chover.
Lecture Notes in Computer Science. Discrete Geometry for Computer Imagery (DGCI 2000), (aceptado), Uppsala (Sweden), Diciembre, 2000.

9.3 Líneas de trabajo futuro

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 [Hoppe, 1999], 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 [El-Sana y Varshney, 1999] 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. [El-Sana et al., 1999] han presentado un pequeño apunte.

Primitivas gráficas. Como se comentó en el capítulo 7, en las librerías gráficas se encuentran primitivas de dibujo que permiten acelerar el proceso de visualización. En dicho capítulo se propuso utilizar la primitiva abanico de triángulos pero no se obtuvo una disminución del tiempo de visualización. Utilizar otras primitivas como la tira de triángulos u otras extensiones que las librerías ofrecen para sacar un mayor partido al sistema gráfico sigue siendo una línea de investigación importante. 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. [El-Sana et al., 1999] 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.

Comparación de modelos multirresolución. Durante la realización de esta tesis, siempre se ha encontrado una gran dificultad a la hora de comparar los resultados obtenidos con los publicados por otros modelos multirresolución. Hasta ahora, cada autor ha comprobado su modelo utilizando criterios propios y, de esta forma, resulta prácticamente imposible comparar los resultados y extraer conclusiones. El diseño de unos tests específicos, bien definidos y reproducibles, es probablemente el único medio para poder comparar las prestaciones de los distintos modelos. En el capítulo 5 se presentó un conjunto de tests que intentan ser un medio para medir las prestaciones de los modelos multirresolución cuando se les solicitan aproximaciones de nivel de detalle uniforme. Diseñar nuevos tests que permitan hacer lo mismo para aproximaciones de nivel de detalle variable, que simulen situaciones que los modelos multirresolución deban resolver, y que los autores puedan reproducir a fin de aplicarlos a sus propios modelos multirresolución, es una línea en la que ya se está trabajando.

Modelado multirresolución de especies vegetales. Los modelos poligonales de especies vegetales tienen una serie de particularidades que los hacen distintos de las mallas que continuamente se han utilizado para experimentar con MOM. Esto hace pensar que ni las técnicas de simplificación ni los modelos multirresolución actuales sean adecuados. Se plantean dos posibles soluciones. La primera, diseñar nuevas técnicas de simplificación y modelado multirresolución específicos para modelos poligonales de especies vegetales. La segunda, aplicar el modelado multirresolución a modelos no convencionales (hasta la fecha no se tiene conocimiento de que se hayan aplicado técnicas multirresolución a este tipo de modelos).

Bibliografía

[Abadjev et al., 1999] V. Abdajev, M. del Rosario, A. Lebedev, A. Migdal and V. Paskhaver. Metastream, *Proc. of VRML'99*, 1999

[Algorri y Schmitt, 1996] M.E. Algorri and F. Schmitt. Mesh Simplification, *Computer Graphics Forum*, 15(3), August, 1996

[Andujar, 1998] C. Andujar. Simplificación de Modelos de Poliédricos, *Report LSI-98-1-T*, Departamento LSI, Universitat Politecnica de Catalunya, 1998

[Andujar, 1999] C. Andujar. Octree-based Simplifcation of Polyhedral Solids, *Tesis doctoral*, Departament LSI, Universitat Politècnica de Catalunya, Junio, 1999

[Bell et al., 1995] 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

[Certain et al., 1996] A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin and W. Stuetzle. Interactive Multiresolution Surface Viewing, *Proc. of SIGGRAPH '96*, 91-98, August, 1996.

[Ciampalini et al., 1997] A. Ciampalini, P. Cignoni, C. Montani and R. Scopigno. Multiresolution Decimation Based on Global Error, *The Visual Computer*, 13(5):228-246, 1997

[Cignoni et al, 1997] P. Cignoni, E. Puppo, and R. Scopigno. Representation and visualization of terrain surfaces at variable resolution, *The Visual Computer*, 13:199-217, 1997

[Cignoni et al, 1998a] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms, *Computers & Graphics*, 22(1):37-54, 1998

[Cignoni et al, 1998b] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces, *Computer Graphics Forum*, 17(2):167-174, June, 1998

[Clark, 1976] J.H. Clark. Hierarchical geometric models for visible surface algorithms, *CACM*, 19(10):547-554, October, 1976

[Duchaineau et al., 1997] M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, C. Aldrich and Mineev-Weinstein. ROAMing terrain: real-time optimally adapting meshes, *Proc. of Visualization '97*, 81-88, 1997

[Eck et al., 1995] M. Eck , T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle. Multiresolution Analysis of Arbitrary Meshes, *Proc. of SIGGRAPH '95*, 173-182, August, 1995

[El-Sana y Varshney, 1999] J. El-Sana and A. Varshney. Generalized View-Dependent Simplification, *Proc. of EUROGRAPHICS '99*, 83-94, September, 1999

[El-Sana et al., 1999] J. El-Sana, E. Azanli and A. Varshney. Skip Strips: Maintaining triangle strips for view-dependent rendering, *Proc. of Visualization '99*, 131-137, 1999

[Erikson, 1996] C. Erikson. Polygonal Simplifcation: An Overview, *Technical Report TR96-016*, Department of Computer Science, UNC-Chapel Hill, 1996

[Finkelstein y Salesin, 1994] A. Finkelstein and D. Salesin. Multiresolution curves, *Proc. of Siggraph '94*, 261-268, July, 1994

[Floriani et al., 1996] L. De Floriani, L. Marzanoo and E. Puppo. Multiresolution models for topographic surface description, *The Visual Computer*, 12, 7, 317-345, 1996

[Floriani et al., 1997a] 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

[Floriani et al., 1997b] L. De Floriani, P. Magillo and E. Puppo. Building and Traversing a Surface at Variable Resolution, *Proceedings IEEE Visualization '97*, 103-110, 1997

[Floriani et al., 1998] L. De Floriani, P. Magillo and E. Puppo. Efficient Implementation of Multi-Triangulations, *Proceedings IEEE Visualization '98*, 1998

[Foley et al., 1990] J.D. Foley, A. van Dam, S. Feiner, J. Hughes and R. Phillips Computer Graphics. Principles and Practice, *Addison-Wesley Publishing Company*, 1990

[Fowler, 1979] R.J. Fowler and J.J. Little. Automatic extraction of irregular network digital terrain models, *Proc. of Siggraph'79*, 13(3), 199-207, August, 1979

[Funkhouser et al., 1992] T. Funkhouser, C. Séquin and S. Teller. Management of large amounts of data in interactive building walkthroughs, *Symposium on Interactive 3D Graphics*, 11-20, 1992

[Funkhouser y Séquin, 1992] T. Funkhouser and C. Séquin. Adaptive display algorithm for interactive frames rates during visualization of complex virtual environments, *Proc. of Siggraph'93*, 247-254, 1993

[Garland y Heckbert, 1997] M. Garland and P. Heckbert. Surface Simplification Using Quadric Error Metrics, *Proc. of SIGGRAPH '97*, 209-216, August, 1997

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

[Gieng et al., 1998] T.S. Gieng, B. Hamann, K.I. Joy, G.L. Schussman and I.J. Trotts. Constructing hierarchies for triangle meshes, *IEEE Transactions on Visualization and Computer Graphics*, 4(2):145-161, April-June, 1998

[Guéziec, 1996] A. Guéziec. Surface simplification inside a tolerance volume, *Technical Report RC 20440*, IBM Research Division, T. J. Watson Research Center, 1996

[Guéziec et al., 1998a] A. Guéziec, G. Taubin, F. Lazarus and W. Horn. Simplicial maps for progressive transmission of polygonal surfaces, *Proc. of VRML'98*, ACM, February, 1998.

[Guéziec et al., 1998b] A. Gueziec, G. Taubin, F. Lazarus and W. Horn. Converting sets of polygons to manifold surfaces by cutting and stitching, *Proc. of IEEE Visualization '98*, 383-390, 553, October, 1998

[Guéziec et al., 1999] A. Guéziec, G. Taubin, B. Horn and F. Lazarus. A Framework for Streaming Geometry in VRML, *IEEE Computer Graphics and Applications*, 19(2):68-78,1999

[Heckbert, 1994] P.S. Heckbert. Graphics Gems IV, AP Professional, 1994.

[Heckbert y Garland, 1994] P.S. Heckbert and M. Garland. Multiresolution modeling for fast rendering, *Proc. of Graphics Interface '94*, 43-50, May, 1994 1994.

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

[Hinker y Hansen, 1993] P. Hinker and C. Hansen. Geometric optimization, *Visualization '93*, 189-195, October, 1993

[Hoppe, 1996] H. Hoppe. Progressive Meshes, *Proc. of SIGGRAPH '96*, 99-108, August, 1996

[Hoppe, 1997] H. Hoppe. View-Dependent Refinement of Progresive Meshes, *Proc. of Siggraph '97*, 189-198, August, 1997

[Hoppe, 1998a] H. Hoppe. Efficient Implementation of Progressive Meshes, *Computer & Graphics*, 22(1):27-36, 1998

[Hoppe, 1998b] H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering, *Proc. of Visualization '98*, 35-42, October, 1998.

[Hoppe, 1999] H. Hoppe. Optimization of mesh locality for transparent vertex caching, *Proc. of Siggraph '99*, 369-276, August, 1999

[Huerta, 1998] J. Huerta. Aproximación multirresolución de los árboles binarios de particionamiento espacial: Aplicación a la generación de entornos virtuales, *Tesis doctoral*, Departamento de Informática, Universitat Jaume I, Febrero, 1998

[Kalvin y Taylor, 1996] A.D. Kalvin and R.H. Taylor. Superfaces: Polygonal Mesh Simplification with bounded error, *IEEE Computer Graphics and Applications*, 16(3), May, 1996

[Kobbelt et al., 1998a] L. Kobbelt, S. Campagna and H.P. Seidel. A general framework for mesh decimation, *Graphics Interface '98*, 43-50, June, 1998

[Kobbelt et al., 1998b] L. Kobbelt, S. Campagna, J. Vorsatz and H.P. Seidel. Interactive multi-resolution modeling on arbitrary meshes, *Proc. of Siggraph'98*, 105-114, 1998

[Klein y Strasser, 1997] 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

[Krus, 1997] M. Krus. Maillages polygonaux et niveaux de détails: étude bibliographique, *Technical report 97-10*, LIMSI/CNRS, Orsay, May, 1997

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

[Lindstrom et al., 1996] 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*, 109-118, August, 1996

[Lindstrom y Turk, 1998] P. Lindstrom and G. Turk. Fast and Memory Efficient Polygonal Simplification, *Proc. of IEEE Visualization 98*, 279-286, 544, October, 1998

[Lindstrom y Turk, 1999] P. Lindstrom and G. Turk. Evaluation of Memoryless Simplification, *IEEE Transactions on Visualization and Computer Graphics*, 5(2), April-June, 1999

[Lounsbery et al., 1997] M. Lounsbery, T. DeRose and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type, *ACM Transactions on Graphics*, 16(1):34-73, 1997

[Low y Tan, 1997] K.-L. Low and T.-S. Tan. Model simplification using vertex-clustering, *1997 Symposium on Interactive 3D Graphics*, ACM SIGGRAPH, 1997

[Luebke y Erikson, 1997] D. Luebke and C. Erikson. View-Dependent Simplification of Arbitrary Polygonal Environments, *Proc. of Siggraph '97*, 199-208, August, 1997

[Maciel, 1995] P. Maciel and P. Shirley. Visual navigation of large environments using textured clusters, *Symposium on Interactive 3D Graphics*, 95-102, April, 1995

[Popovic y Hoppe, 1997] J. Popovic and H. Hoppe. Progressive simplicial complexes, *Proc. of SIGGRAPH'97*, 217-224, 1997

[Puppo, 1996] E. Puppo. Variable resolution terrain surfaces, *Proc. of Eight Canadian Conference on Computational Geometry*, 202-210, August, 1996

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

[Puppo, 1998] E, Puppo. Variable resolution triangulations, *Computational Geometry*, 11(3-4):219-238, December, 1998

[Reddy, 1994] M. Reddy. Reducing lags in virtual reality systems using motion-sensitive level of detail, *Proc. of 2nd UK VR-SIG*, 25-31, 1994

[Reddy, 1995] M. Reddy. A survey of level of detail support in current virtual reality solutions, *Virtual Reality: Research, Development and Applications*, 1(2):85-88, 1995

[Rohlf y Helman, 1994] J. Rohlf and J. Helman. IRIS Performer: A high performance multiprocessing toolkit for real-time 3D graphics, *Proc. of Siggraph '94*, 381-394, July, 1994

[Ronfard y Rossignac, 1996] R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra, *Computer Graphics Forum*, 15(3), August, 1996

[Rossignac y Borrel, 1996] J. Rossignac and P. Borrel. Multi-resolution 3D approximations for rendering complex scenes, *Modeling in Computer Graphics: Methods andApplications*, 455-465, 1993

[Scarlatos, 1990] L.L. Scarlatos. A refined triangulation hierarchy for multiples level of terrain detail, *Proc. of Image V Conference*, 115-122, June, 1990

[Schroeder et al., 1992] W.J. Schroeder, J.A. Zarge and W. Du. Decimation of triangle meshes, *Computer Graphics*, 26(2):65-70, July, 1992

[Schroeder, 1997] W.J. Schroeder. A topology modifying progressive decimation algorithm, *IEEE Visualization '97*, 205-212, 545, 1997

[Stollnitz et al., 1996] E.J. Stollnitz, T.D. DeRose and D.H. Salesin. Wavelets for Computer Graphics: Theory and Applications, *Morgann Kaufmann*, San Francisco (CA), 1996

[Taubin et al., 1998] G. Taubin, A. Guéziec, W. Horn and F. Lazarus. Progressive forest split compression, *Proc. of Siggraph'98*, 123-132, July, 1998

[Taubin y Rossignac, 1998] G. Taubin and J. Rossignac. 3D Geometry compression, *Siggraph'98 Course Notes 21*, July, 1998

[Taubin, 1999] G. Taubin. 3D Geometric compression and progressive transmission, *State of the art reports of Eurographics'99*, September, 1999

[Taubin et al., 1999] G. Taubin, W. Horn and P. Borrel. Compression and transmission of multi-resolution clustered meshes, *Technical report RC-21398*, IBM Research, February, 1999

[Taubin y Rossignac, 1999] G. Taubin and J. Rossignac. 3D Geometry compression, *Siggraph'99 Course Notes 22*, August, 1999

[Taylor y Barret, 1994] D.D. Taylor and W.A. Barret. An algorithm for continuous resolution polygonalizations of a discrete surface, *Proc. of Graphics Interface '94*, 33-42, 1994

[Teller et al., 1994] S. Teller, C. Fowler, T. Funkhouser and P. Hanrahan. Partitioning and ordering large radiosity computations, *Proc. of the 21st Annual Conference on Computer Graphics*, 443-450, 1994

[Vidal y Ribelles, 1999] R. Vidal y J. Ribelles. Un Servidor de Geometría 3D, *Actas del IX Congreso Español de Informática Gráfica (CEIG'99)*, 355-356, June, 1999

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

[Willmott et al., 1999] A. Willmott, P. Heckbert and M. Garland. Face cluster radiosity, *Eurographics Workshop on Rendering*, June, 1999

[Xia y Varshney, 1996] J. Xia and A. Varshney. Dynamic view-dependent simplification for polygonal models, *Proc. of IEEE Visualization '96*, 327-334, October, 1996

[Xia et al., 1997] 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):171-183, 1997

[Zorin et al., 1997] D. Zorin, P. Schröder and W. Sweldens. Interactive Multiresolution Editing, *Proc. of Siggraph '97*, 259-268, 1997