Técnicas de aceleración para el método de radiosidad jerárquica

Emilio José Padrón González

Departamento de Electrónica e Sistemas
Universidade da Coruña

Departamento de Electrónica e Sistemas
Universidade da Coruña

Tesis Doctoral

Técnicas de Aceleración para el Método de Radiosidad Jerárquica

Emilio José Padrón González

Mayo de 2006

Dirigida por:
Margarita Amor López
y Ramón Doallo Biempica

Dra. Margarita Amor López, Profesora Titular de Universidad del área de Arquitectura y Tecnología de Computadores de la Universidade da Coruña.

Dr. Ramón Doallo Biempica, Catedrático de Universidad del área de Arquitectura y Tecnología de Computadores de la Universidade da Coruña.

CERTIFICAN:

Que la memoria titulada "Técnicas de Aceleración para el Método de Radiosidad Jerárquica" ha sido realizada por Emilio José Padrón González bajo nuestra dirección en el Departamento de Electrónica e Sistemas de la Universidade da Coruña y concluye la Tesis que presenta para optar al grado de Doctor en Informática.

A Coruña, 31 de Enero de 2006

Dra. Margarita Amor López.
Codirectora de la tesis.

Dr. Ramón Doallo Biempica.
Codirector de la tesis.

Fdo. Dr. Luis Castedo Ribas.
Director del Departamento de Electrónica e Sistemas.

Agradecimientos

Antes de pasar a los agradecimientos personales, me gustaría mencionar los organismos e instituciones que han proporcionado los aspectos logísticos y la financiación necesaria que me han permitido la realización de este trabajo de Tesis Doctoral. Primero, y especialmente, al Departamento de Electrónica y Sistemas de la Universidade da Coruña, y en particular al Grupo de Arquitectura de Computadores, por acogerme en su programa de doctorado y darme la oportunidad de formarme como investigador, así como por facilitarme el acceso a todos sus recursos siempre que me fueron necesarios. En segundo lugar, agradecer a los siguientes organismos públicos, la Dirección Xeral de Investigación, Desenvolvemento e Innovación de la Xunta de Galicia y los Ministerios de Ciencia y Tecnología y de Educación, el haberme proporcionado el acceso a las becas que me han permitido desenvolver mi etapa como investigador: una beca predoctoral en el primer caso y una beca FPI en el segundo, asociada esta última al proyecto de investigación *TIC2001-3694-C02-02*. Además, también agradezco la financiación de mi labor investigadora a través de los proyectos *TIN2004-07797-C02-02* y *PGIDIT04TIC105004PR* al Ministerio de Educación y a la Xunta de Galicia, respectivamente. También quiero dar las gracias al CESGA (*Centro de Supercomputación de Galicia*) y al EPCC de la Universidad de Edinburgh por el acceso a los diversos sistemas de procesamiento paralelo que se han empleado para la realización de este trabajo. En el caso del EPCC, hacer mención también a la formación recibida durante el periodo de estancia en sus instalaciones a través del programa de visitas TRACS (ahora HPC-Europa), agradeciendo el magnífico trato y apoyo recibido por parte de todo su personal.

En lo personal, quiero agradecer especialmente a mis dos directores de Tesis, Marga y Ramón, todo el trabajo y esfuerzo que han invertido en verme llevar a buen puerto este trabajo. Su constante apoyo y consejo ha sido fundamental para el desarrollo de este trabajo. También me gustaría mencionar a Montserrat Bóo, por su interés y ayuda en la discusión de algunas de las aportaciones que hemos tratado de realizar en este trabajo. Por último, dar las gracias a todos los que me han ayudado y soportado todo este tiempo... y lo que les queda.

Índice general

Índice de tablas

Índice de figuras

Resumen

La búsqueda del realismo visual en imágenes construidas de forma sintética ha aumentado considerablemente en los últimos años, extendiéndose además a multitud de campos, como pueden ser el diseño arquitectónico, la visualización científica, el cine y la televisión, o la medicina, entre otros. Esta demanda cada vez mayor de lograr acabados realistas ha llevado, dentro de la investigación en informática gráfica, al desarrollo de métodos que tratan de simular la interacción de la luz con un entorno virtual.

Uno de los métodos que mejor modelan el comportamiento real de la luz en su interacción con una escena es el método de radiosidad. Este método presenta, sin embargo, el inconveniente de un alto coste computacional, tanto en tiempo de cálculo como en almacenamiento. Entre las numerosas variantes surgidas con el objetivo de rebajar la complejidad del método clásico destaca el método de radiosidad jerárquica, basado en la aplicación de una subdivisión adaptativa de la escena.

El método de radiosidad jerárquica, no obstante, mantiene todavía una elevada complejidad que dificulta su explotación en escenas de gran tamaño y mantiene unos tiempos de ejecución demasiado elevados. En este trabajo hemos tratado de desarrollar nuevas y distintas soluciones para algunos de los diversos problemas que el método jerárquico de radiosidad plantea.

En el Capítulo 1 realizamos una breve introducción a la problemática de la iluminación global, y al método de radiosidad jerárquica en particular, destacando los principales cuellos de botella presentes en la aproximación clásica del método y las soluciones que proponemos en este trabajo para sortearlos. En el resto de esta memoria describimos cada una de nuestras aportaciones para la optimización de las diferentes fases del método.

En el Capítulo 2 nos centramos en el que es uno de los principales cuellos de botella de cualquier algoritmo de iluminación global: la determinación de la visibilidad

entre los distintos objetos. En este capítulo analizamos las principales propuestas existentes, implementando y realizando pruebas de algunas de las más relevantes, y además, proponemos una nueva aproximación al problema, que denominamos *Coherencia Angular de los Rayos* (*ACR*), basada en aprovechar el principio de localidad en el espacio de direcciones de los rayos lanzados durante el proceso de cálculo de la visibilidad.

Un exceso en el detalle geométrico de la escena a procesar no siempre aporta una cantidad significativa de información al cálculo de la iluminación de la misma, aunque sí supone un aumento drástico en el coste computacional de este proceso. El tratamiento de escenas grandes y con objetos complejos requiere de la aplicación de distintos niveles de resolución en la función de radiosidad, independizando en lo posible la correcta simulación de la distribución de la energía en la escena de la complejidad geométrica de los objetos que la componen. Estas cuestiones son discutidas en los capítulos 3 y 4, presentando una propuesta para el cálculo de la radiosidad jerárquica basada en la subdivisión de superficies.

En el Capítulo 3 analizamos las propuestas más interesantes en cuanto a algoritmos de subdivisión de superficies, que serán luego empleadas en el siguiente capítulo para obtener versiones detalladas de objetos simples. Concretamente nos centramos en aquellos métodos que utilizan la información con grado de vecindad uno de la malla de polígonos. Se describen las dos aproximaciones principales al problema, como son los métodos aproximados y los métodos interpolados, escogiendo como representantes de cada uno de ellos los métodos *Loop* y *Modified Butterfly*, respectivamente. En este capítulo proponemos, además, una implementación paralela general válida para cualquiera de estos métodos, basada en la creación, y posterior reparto, de grupos de triángulos, utilizando un paradigma de memoria distribuida con paso de mensajes.

El Capítulo 4 de este trabajo introduce el uso de técnicas de multiresolución como solución para la reducción efectiva de la complejidad del método de radiosidad jerárquica. Se enumeran algunas de las técnicas más utilizadas y se describe nuestra aproximación propia al problema, basada en el uso de los esquemas de subdivisión de superficies descritos en el capítulo anterior. Con las dos soluciones que proponemos en este capítulo, *método de radiosidad basado en subdivisión de superficies* y *método de radiosidad basado en agrupamiento de normales*, se reduce drásticamente la complejidad del método jerárquico de radiosidad, consiguiendo, como se verá, un rendimiento en el cálculo de la iluminación de la escena altamente independiente de la complejidad geométrica de los objetos de su interior.

En el Capítulo 5, y último, proponemos una solución paralela al problema, que permitirá aprovechar el uso de sistemas distribuidos para el cálculo de la iluminación global mediante el método de radiosidad jerárquica. Primero hacemos un repaso exhaustivo a los distintos trabajos existentes en radiosidad paralela, presentando a continuación dos nuevas aproximaciones, ambas dirigidas a sistemas con memoria distribuida. La primera de nuestras implementaciones, *radiosidad paralela con replicación de la escena*, se basa en el mantenimiento de toda la escena de entrada en las distintas memorias del sistema, de forma que se encuentra accesible por completo a todos los procesadores. En el método se implementa un esquema de balanceo dinámico de la carga computacional que permite mantener dicha carga equilibrada entre los distintos procesadores. Nuestra segunda implementación, *radiosidad paralela con distribución de la escena*, elimina la restricción que supone la existencia de réplicas de la escena completa, distribuyéndola por completo entre todas las memorias del sistema. Con la escena repartida entre los distintos procesadores, el cálculo efectivo de la determinación de visibilidad entre procesadores ha necesitado de la implementación de un nuevo esquema de evaluación distribuida del lanzamiento de rayos. Además, el método se ha implementado siguiendo una aproximación multi-hilo que permite un mejor ajuste de la granularidad utilizada.

Por último, se indican las principales conclusiones y aportaciones de este trabajo, mencionando las líneas de investigación abiertas por el mismo.

Capítulo 1

El método de radiosidad jerárquica

Uno de los principales objetivos de la informática gráfica es la síntesis de imágenes de apariencia realista. La correcta iluminación de una escena sintética es un requisito básico para alcanzar esos niveles de realismo. Con este objetivo, se han desarrollado los llamados modelos de iluminación global, que tratan de simular la propagación de la luz en una escena tridimensional.

En este capítulo describimos los principios básicos y el funcionamiento general del método de radiosidad jerárquica, variante del método de radiosidad clásico en la que hemos centrado este trabajo. El método de radiosidad es, dentro de los modelos de iluminación global, el que de una manera más natural permite modelar el equilibrio que se alcanza en la distribución de la luz en un entorno. La variante jerárquica de este método, por su parte, consigue reducir la alta complejidad del algoritmo clásico, al tratar de forma menos detallada los casos de interacción con la luz menos relevantes, refinando solamente donde es necesario.

1.1. Iluminación global: ecuación de *rendering*

Los modelos de iluminación global [41], frente a los habitualmente conocidos como modelos de iluminación local, consideran la interreflexión de la luz entre las distintas superficies de un entorno, simulando la distribución de la energía en el entorno teniendo en cuenta todas las superficies y las fuentes de luz. Todos los modelos de iluminación global tratan de resolver la ecuación conocida como ecuación de *rendering* [74].

La ecuación de *rendering* expresa el transporte o distribución de la energía lumínica en un entorno. Esta ecuación está basada en el comportamiento físico real de la luz, en contraste con otras muchas técnicas, de uso habitual en síntesis de imágenes, que tratan de obtener una apariencia más o menos realista mediante aproximaciones, no excesivamente fieles, del mismo. Su expresión formal clásica es la siguiente:

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_{\Omega_x} f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}_x) d\vec{w}' \tag{1.1}$$

La esencia de la ecuación es la ley de la conservación de la energía: dados un punto, x, y una dirección $\vec{w}$ concretos, la energía [1] que abandona ese punto en esa dirección, L_o, es la suma de la posible energía auto-emitida por ese punto (emitancia), L_e, y la energía reflejada. El valor de la energía reflejada en el punto x hacia la dirección $\vec{w}$ se corresponde con la aportación de toda la energía proveniente del resto de la escena en todas las direcciones, L_i, multiplicada por el ángulo de incidencia [2] y el factor de reflexión del material, f_r. Este último término viene dado por la función de reflexión bidireccional de un material (*Bidirectional Reflectance Distribution Function*, *BRDF*), que modela la interacción de la luz con una superficie, expresando la relación entre la energía incidente y la reflejada, L_r, en un determinado punto y dirección [3] (ver Figura 1.1):

$$f_r(x, \vec{w}', \vec{w}) = \frac{dL_r(x, \vec{w})}{L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}_x) d\vec{w}'} \tag{1.2}$$

El comportamiento de la luz en un medio, representado por la *BRDF*, puede descomponerse en varios términos más sencillos, de forma que cada uno de ellos expresa un determinado aspecto de la interacción de la luz con el material que lo compone, como pueden ser, por ejemplo, la reflexión difusa, la reflexión especular, la refracción o la transmisión de la energía a través del medio, entre otros (ver Figura 1.2). En la práctica, los diferentes modelos de iluminación global suelen centrarse en uno o varios de estos términos más sencillos, simplificando el efecto que

[1] La medida básica para la energía utilizada en radiometría es la radiancia (*radiance*), que se corresponde con el flujo de energía que incide o parte de una superficie por unidad de ángulo sólido y por unidad de área. Se expresa en $\frac{watts}{steradians \cdot m^2}$.

[2] La notación empleada, $\vec{w}' \cdot \vec{n}_x$, indica el producto escalar entre la dirección incidente, $\vec{w}'$, y el vector normal a la superficie incidida, $\vec{n}_x$, que corresponde al coseno del ángulo que ambos forman.

[3] En radiometría, radiancia de entrada (*irradiance*) y de salida (*radiant exitance*), respectivamente. Esta última, correspondiente al flujo de energía por unidad de área (dA) que abandona una superficie, recibe también el nombre de radiosidad, y se expresa en $\frac{W}{m^2}$.

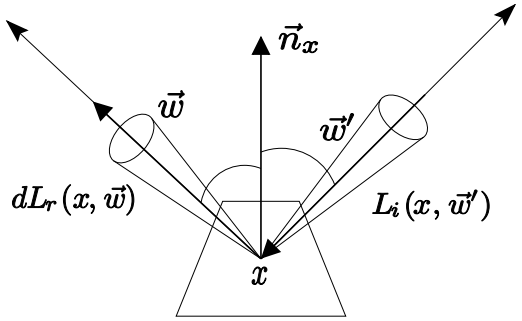


Figura 1.1: Representación geométrica de la *BRDF*.

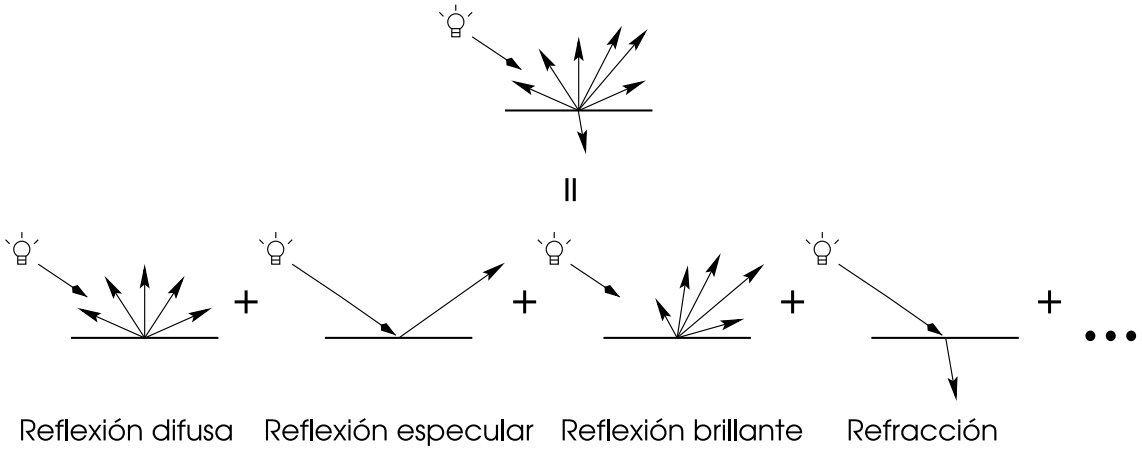


Figura 1.2: Componentes de la *BRDF*.

produce el comportamiento físico real de la luz.

Los modelos de iluminación global tratan de resolver, normalmente de forma aproximada, mediante la introducción de una serie de simplificaciones, la ecuación de *rendering*. Dos son las principales corrientes dentro de este tipo de métodos: los modelos basados en elementos finitos, conocidos como métodos de radiosidad (*radiosity*), y los modelos basados en el lanzamiento o trazado de rayos, métodos de *ray tracing* [52].

Los métodos basados en *ray tracing* generan una serie de rayos o caminos entre la cámara y las distintas fuentes de luz de la escena, mediante los cuales se va a estimar el color de los píxeles en pantalla. La propia naturaleza del lanzamiento de rayos hace que estos métodos sean especialmente útiles a la hora de modelar algunos de los efectos producidos por la interacción de la luz, como la reflexión especular, los brillos o la refracción (sobre todo con la introducción de métodos estocásticos, como Monte

Carlo, como extensión del método de trazado clásico [29, 83, 84]) que dependen de la dirección en la que es reflejada la luz; sin embargo, este tipo de métodos no son los adecuados cuando se trata de simular la componente difusa en el reflejo de la luz, tipo de reflexión muy influyente en el aspecto final de la iluminación de un entorno cerrado. Este inconveniente, unido al hecho de que estos métodos proporcionan un cálculo de la iluminación global de la escena dependiente del punto del vista del observador, y ciertos problemas de ruido y de convergencia, hacen que su uso en solitario para el cálculo de la iluminación de una escena no siempre sea la mejor opción.

Los métodos de radiosidad, por su parte, simulan especialmente bien el comportamiento difuso de la luz en un entorno, como veremos en las siguiente secciones y en el resto de este trabajo, permitiendo obtener resultados extraordinariamente realistas en la iluminación lograda. Además, los resultados proporcionados son totalmente independiente del punto de vista de la cámara. Se puede considerar, por tanto, que ambas aproximaciones, radiosidad y *ray tracing*, son complementarias, por lo que es frecuente encontrarse con métodos híbridos [112, 124, 141] que, una vez obtenida la iluminación difusa de la escena con el método de radiosidad, aplican técnicas de lanzamiento de rayos para añadir efectos especulares, brillos, interacción con medios participativos, etc. al resultado final.

La principal limitación del método de radiosidad consiste en la dependencia que establece entre la iluminación de una escena y su geometría, lo que limita, en principio, la complejidad de las escenas a utilizar. Aunque esta limitación puede superarse mediante el uso de simplificaciones geométricas y jerarquías de objetos, como veremos en el Capítulo 4, una técnica conocida como *Photon mapping* [20, 73] ha surgido en los últimos años para tratar de sustituir la aproximación híbrida y basar la iluminación de una escena únicamente en el lanzamiento de rayos, logrando la independencia de la complejidad de la escena. El *Photon mapping* consiste, básicamente, en la segmentación del proceso de lanzamiento de rayos en dos fases: primero se emiten y propagan a lo largo de la escena los rayos, que ahora se pasan a denominar *fotones*, a partir de las fuentes de luz de la escena, almacenando la información obtenida sobre la iluminación en una estructura independiente de la geometría de la escena, el *mapa de fotones*. El segundo paso consiste en utilizar toda la información disponible para calcular la iluminación final de la escena, sustituyendo a la información que proporcionaría el método de radiosidad en las soluciones híbridas radiosidad-*ray tracing*. El principal inconveniente de esta aproximación radica en la gran cantidad de memoria que requiere el lanzamiento de un número de

fotones lo suficientemente grande como para obtener resultados realistas en la iluminación indirecta en un entorno cerrado [55]. Otro potencial problema al que las implementaciones tienen que hacer frente es a la resolución eficiente de las relaciones de vecindad entre puntos arbitrarios del espacio, necesaria para el aprovechamiento de la estructura del mapa de fotones [88].

1.2. Método de radiosidad

La radiosidad supone una aproximación al problema de la iluminación global basada en elementos finitos. Consiste, primero, en la subdivisión de la escena en un conjunto de elementos pequeños, procediendo a continuación a calcular la energía que se transmite entre ellos, de forma totalmente independiente del punto de vista del observador. La idea subyacente detrás del método de radiosidad es la de lograr un equilibrio en la distribución de la energía de la escena, no en vano la idea del modelo viene de la termodinámica. Con este fin, la radiosidad clásica únicamente tiene en cuenta la componente de reflexión difusa de la luz, necesitando el complemento de alguna técnica de lanzamiento de rayos para la aplicación posterior de efectos especulares o de otro tipo.

La discretización de la escena en elementos finitos, suponiendo además un valor de radiosidad constante a lo largo de la superficie de cada elemento, junto con la simplificación introducida por el modelo, al considerar únicamente la componente difusa de la reflexión de la luz y a todas las superficies de la escena como reflectores difusos ideales [4], permiten transformar la ecuación integral de *rendering* (Ecuación 1.1) en la ecuación clásica de radiosidad:

$$B_i = E_i + \rho_i \sum_{j=1}^{N} B_j F_{ij} \tag{1.3}$$

La interpretación física de la ecuación (representada en la Figura 1.3) nos dice que la radiosidad que abandona la superficie de un elemento por unidad de área, B_i, corresponde a su energía emitida, E_i, en caso de ser una fuente lumínica, más la suma de la energía recibida del resto de elementos de la escena. La fracción de energía

[4]Una superficie con reflexión difusa refleja la luz incidente en todas las direcciones. La reflexión difusa ideal o *Lambertiana* sería el caso particular en el que la energía reflejada es constante en todas las direcciones.

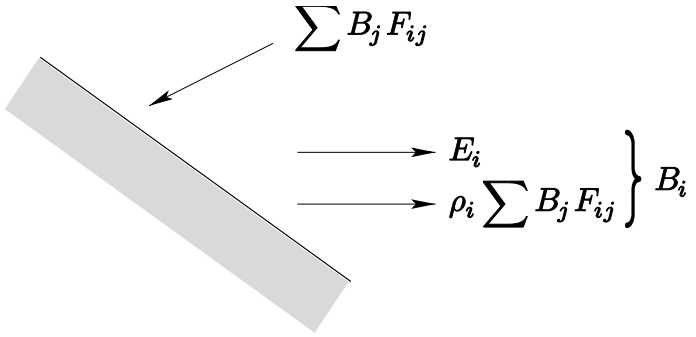


Figura 1.3: Interpretación física de la ecuación clásica de radiosidad.

que se transmite entre cada par de elementos viene determinada por el término F_{ij}, denominado factor de forma (*form factor*), y cuya expresión es la siguiente:

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{(\vec{w} \cdot \vec{n}_x)(\vec{w}' \cdot \vec{n}_{x'})}{|x - x'|^2} dA_j dA_i \; V(i,j) \tag{1.4}$$

En la ecuación se pueden distinguir dos partes: un término de visibilidad y un término geométrico. El término de visibilidad, $V(i,j)$, tiene un valor 0 cuando los elementos no son en absoluto visibles entre sí, y 1 cuando son completamente visibles. El término geométrico, por su parte, indica el porcentaje de energía que se transmite desde el elemento i al elemento j, en función de la distancia a la que se encuentran y la orientación relativa de ambos elementos entre sí. El coste computacional que implica la resolución de la doble integral del factor de forma para cada interacción entre dos elementos de la escena hace que sea habitual, cuando la diferencia entre el tamaño de los objetos y la distancia a la que estos se encuentran es grande, la aproximación de los elementos por discos orientados con la misma área. Se aplica en este caso la expresión más sencilla del factor de forma entre una diferencial de área y un disco (ver Figura 1.4):

$$F_{i\Delta j} = \Delta j \frac{(\vec{w} \cdot \vec{n}_x)(\vec{w}' \cdot \vec{n}_{x'})}{\pi |x - x'|^2 + \Delta j} \; V(i, \Delta j) \tag{1.5}$$

en donde Δj corresponde al área de la superficie que es aproximada con un disco, y los productos escalares $\vec{w} \cdot \vec{n}_x$ y $\vec{w}' \cdot \vec{n}_{x'}$ representan los cosenos de los ángulos entre las normales a cada una de las superficies en los puntos x y x' y los vectores de dirección de la línea que une ambos puntos (Θ_i y Θ_j en la Figura 1.4).

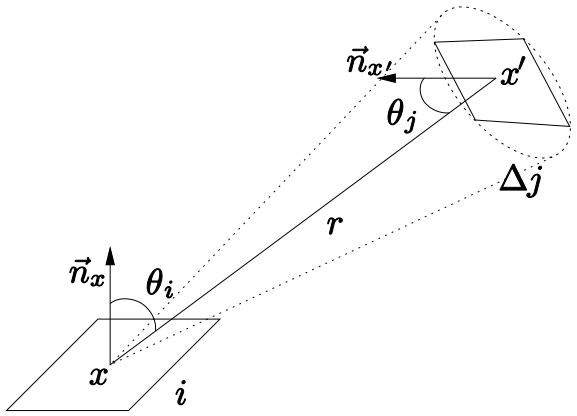


Figura 1.4: Factor de forma entre una diferencial de área y un disco orientado.

El método clásico de radiosidad [56] calcula la iluminación de una escena discretizándola, primero, en un conjunto lo suficientemente fino de elementos, para los que a continuación construye y resuelve un sistema de ecuaciones lineales, con una ecuación como la Ecuación 1.3 para cada elemento de la escena. La representación en forma matricial del sistema de ecuaciones lineales resultante sería la siguiente:

$$\underbrace{\begin{pmatrix} 1-\rho_1 F_{11} & \dots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & \dots & -\rho_2 F_{2N} \\ \vdots & \ddots & \vdots \\ -\rho_{N-1} F_{N-1,1} & \dots & -\rho_{N-1} F_{N-1,N} \\ -\rho_N F_{N1} & \dots & 1-\rho_N F_{NN} \end{pmatrix}}_{\mathbf{K}} \underbrace{\begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_{N-1} \\ B_N \end{pmatrix}}_{\mathbf{B}} = \underbrace{\begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_{N-1} \\ E_N \end{pmatrix}}_{\mathbf{E}} \tag{1.6}$$

donde los coeficientes **B** y **E** son dos vectores con los valores de radiosidad y de emitancia, respectivamente, de todos los elementos de la escena, y **K** corresponde a la matriz de interacciones entre los distintos elementos. La radiosidad de un elemento es descrita, entonces, como una función de la radiosidad de cada uno de los otros elementos de la escena, ponderada con los factores de forma correspondientes y el índice de reflexión difuso del elemento. De esta forma, el sistema clásico de radiosidad tiene una complejidad, tanto computacional como de almacenamiento de $O(N^2)$, que lo hace prohibitivo para escenas grandes.

La resolución del sistema de ecuaciones que propone el método de radiosidad se resuelve, normalmente, mediante un proceso iterativo, que acercará la solución

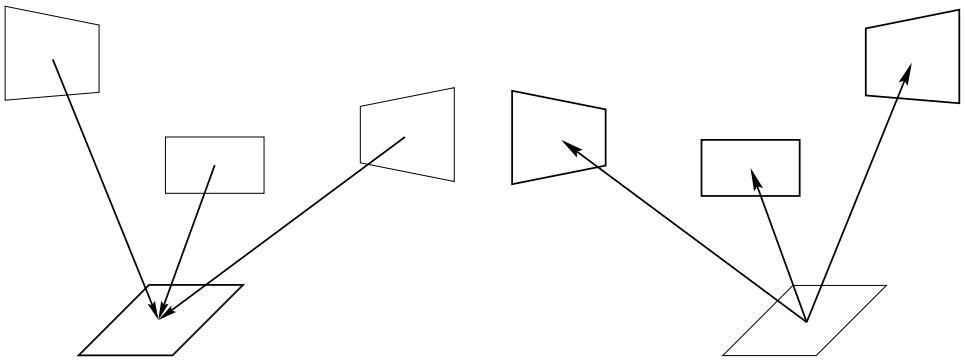


Figura 1.5: (a) *Acumulación* de energía en un elemento (*gathering*) (b) *Disparo* de la energía de un elemento (*shooting*).

obtenida a la solución real tanto como sea necesario. La complejidad del sistema hace inviable, sin embargo, el uso de las técnicas habituales para la resolución iterativa de sistemas de ecuaciones lineales. Entre las diferentes alternativas que se han desarrollado para hacer manejable el cálculo de la radiosidad de una escena, se encuentra el método de radiosidad progresiva [28], que no almacena la matriz **K** de interacciones y, en su lugar, calcula los factores de forma *bajo demanda*. Así, para ello, en lugar de resolver secuencialmente, y de forma completa, las diferentes ecuaciones del sistema planteado, acumulando en cada paso la radiosidad total obtenida por un elemento de la escena, proceso conocido como *gathering* (ver Figura 1.5a); el método progresivo procede *disparando*, en cada paso, la energía de un elemento al resto de elementos de la escena, proceso de *shooting* (ver Figura 1.5b), momento en el que se calculan los factores de forma necesarios, descartándolos una vez usados, sin guardarlos. Si el primero de estos dos casos se corresponde con un proceso iterativo de Gauss-Seidel [57, 133], al realizar el producto de un elemento del vector B, B_j, con toda la columna correspondiente de la matriz **K**, $-\rho_i Fij$, en el refinado progresivo se aplica una variante de la relajación de Southwell [57, 133] para la resolución iterativa de sistemas de ecuaciones lineales.

Para garantizar la convergencia del método, el refinado progresivo escoge, en cada paso, siempre el elemento de la escena con mayor energía todavía *no disparada*, propagando su radiosidad entre el resto de la escena. El resultado del método es, tras cada paso, una escena útil en la que la energía distribuida va aumentando gradualmente. La complejidad computacional, para el disparo de k de los N elementos de la escena sería de $O(kN)$.

Otra manera de reducir la complejidad de la matriz de interacciones, para el cálculo de la radiosidad en una escena, consiste en la aplicación de soluciones jerárquicas análogas a aquellas que han dado buenos resultados en el problema de los N-cuerpos en la física gravitacional (*N-body problem* [129]), y que se basan en que, desde un punto de vista físico, a grandes distancias los pequeños detalles no tienen ninguna influencia. Esto va a permitir, como veremos en el resto del capítulo y a lo largo de todo el trabajo, simplificar los cálculos, haciendo hincapié en las interacciones más relevantes para la iluminación global y simplificando el resto.

Matemáticamente, el sistema de ecuaciones que propone el método clásico de radiosidad puede verse como la proyección de la integral de radiosidad sobre un conjunto de funciones base con valor cero en todo su dominio excepto en un determinado intervalo de valor constante (*piecewise constant functions*). Por otro lado, el método jerárquico de radiosidad corresponde a una simplificación de la matriz de interacción $\mathbf{K}$, haciéndola más dispersa mediante la aproximación de conjuntos de coeficientes muy similares por valores constantes. La generalización de ambas ideas lleva a lo que se conoce como *Wavelet radiosity* [58], con el uso de funciones base jerárquicas o *wavelets*, que proporcionan una herramienta de análisis que permite manejar tanto aproximaciones a la radiosidad con funciones base de mayor orden (*Galerkin radiosity* [147]), como distintas métodos jerárquicos, de forma que los casos expuestos de radiosidad clásica y radiosidad jerárquica constituirían el caso particular en el que se usan funciones base constantes para la representación de la función de radiosidad sobre la escena. En la práctica, las soluciones *wavelet* están en desuso, siendo absorbidas por el refinado jerárquico en combinación con técnicas de *clustering*, que veremos más adelante.

Entre las últimas tendencias en modelos de iluminación global se encuentra la utilización de métodos de Monte Carlo para la resolución del método de radiosidad [12, 111, 119]. La base de estos métodos está en la aplicación de un muestreo de Monte Carlo para llevar a cabo el cálculo de la integral del factor de forma. El principal inconveniente de estos métodos, sin embargo, es su alto coste computacional, lo que habitualmente obliga a la búsqueda de técnicas adicionales que permitan rebajar este coste.

Hasta hace relativamente poco era prácticamente impensable encontrarse con implementaciones *hardware* completas de modelos de iluminación global como la radiosidad en las tarjetas gráficas comerciales, aunque sí ha habido trabajos que aprovechan la tarjeta gráfica para acelerar determinadas partes del cálculo [78, 90]. Recientemente, sin embargo, han aparecido los primeros desarrollos [30, 31, 110]

que logran aprovechar las características introducidas en el *hardware* gráfico más reciente [19], permitiendo realizar todos los cálculos concernientes a la iluminación en la propia *GPU* (*Graphics Processing Unit*). Centrándonos en el método de radiosidad, el trabajo presentado en [30, 31] logra aprovechar las *GPUs* programables para implementar una variante del refinado progresivo con un cierto grado de subdivisión adaptativa, aunque todavía presenta grandes limitaciones, como la restricción de trabajar con los polígonos de entrada como el nivel más grueso de subdivisión empleado y el tamaño y complejidad de las escenas a procesar.

En el resto de este capítulo describimos en detalle el método de radiosidad jerárquica, en el que hemos centrado nuestra investigación. Este método ofrece grandes posibilidades en cuanto al tratamiento multiresolución de la distribución de la energía en la escena, siendo la base de la mayoría de los trabajos recientes centrados en radiosidad.

1.3. El método de radiosidad jerárquica

El método de radiosidad jerárquica original [64] parte de un mallado de la escena más grueso que el de los métodos de radiosidad tradicionales, refinando recursivamente los elementos *a posteriori*, durante el cálculo iterativo de la radiosidad, y de forma adaptativa, en función de la energía transportada por las distintas interacciones. Se crea, de esta forma, una jerarquía de subdivisiones de los elementos iniciales de la escena, con la correspondiente jerarquía de interacciones asociada. El uso de esta jerarquía va a permitir resolver el cálculo de la radiosidad de la escena rebajando drásticamente la complejidad $O(N^2)$ de la matriz de interacción clásica, al sustituir interacciones finas innecesarias por otras más gruesas, como se muestra en la Figura 1.6. En este ejemplo, dos jerarquías de elementos interactúan entre sí en diferentes niveles (Figura 1.6a). Mientras que para la transferencia de la energía desde la jerarquía de la izquierda es suficiente con una precisión en la que los elementos refinados a, b, c y d interactúen directamente con el elemento más grueso de la otra jerarquía (elemento 2), la propagación de la energía transmitida desde la jerarquía del elemento 2 necesita de un nivel de detalle más fino, de ahí que, como se muestra en la Figura 1.6a, los elementos e, f, g y h interactúan con los elementos que están en su mismo nivel en el elemento 1, que en este caso actúan como receptores. Esto equivale, como vemos en la Figura 1.6b, a substituir varias interacciones de los niveles más detallados por una única más gruesa equivalente. Por tanto, cada enlace o interacción que relaciona elementos intermedios del árbol jerárquico de refinado

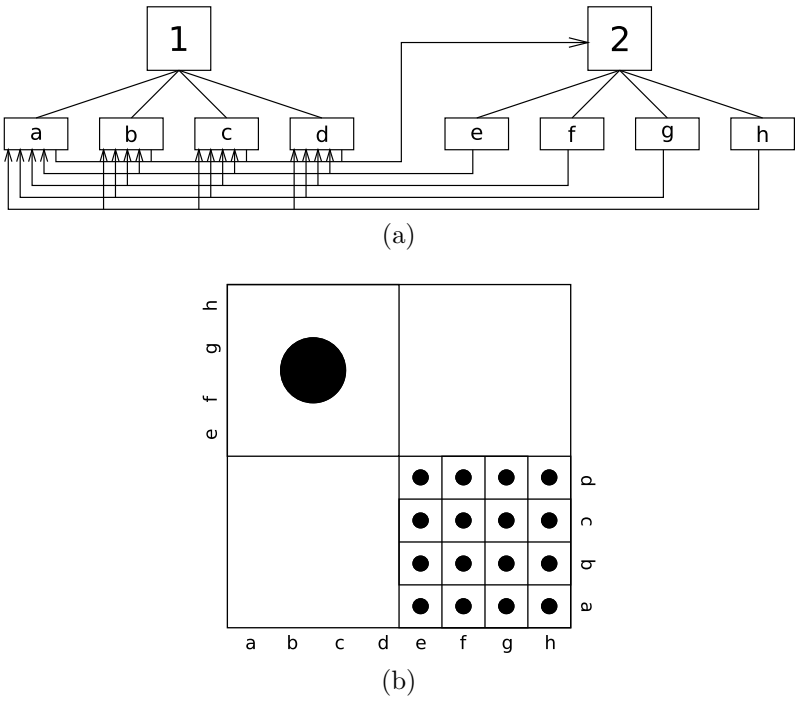


Figura 1.6: Interacciones entre varios niveles de la jerarquía de elementos: (a) Elementos interactuando (b) Matriz de interacción resultante.

representa un subconjunto completo de factores de forma de la matriz **K** original $(N \times N)$.

Otra ventaja de la aproximación jerárquica viene de utilizar un criterio de refinado de los elementos de la escena basado en la distribución de la luz en la misma, de forma que el número final de elementos hoja obtenido con la subdivisión adaptativa va a ser mucho menor que el número de elementos de los que inicialmente se partiría en un esquema de radiosidad clásico para la obtención de un resultado equivalente. La complejidad del método jerárquico de radiosidad, para un conjunto inicial de K elementos gruesos, y alcanzando un número final, tras la iluminación de la escena, de N elementos, con $K \ll N$, es del orden de $O(N + K^2)$, tendiendo a $O(N)$ a medida que aumenta la complejidad de la escena a procesar.

En la Figura 1.7 se presenta un diagrama de flujo que representa el esquema típico del proceso de cálculo de la radiosidad de una escena mediante un refinado jerárquico. La fase principal de cálculo, que se corresponde con el proceso iterativo (FASE 3 en la figura), aparece dividida en tres etapas, a las que preceden una fase inicial de pre-proceso (FASE 1) y la etapa de cálculo de la matriz de interacciones

iniciales (FASE 2). Una vez alcanzada la convergencia, una etapa de post-proceso se encarga de dar los últimos retoques a la escena antes del *rendering*. El esquema presentado en la figura apenas difiere del que correspondería a la resolución del sistema utilizando el método clásico de radiosidad, si exceptuamos los tres pasos que distinguimos en cada iteración del proceso iterativo principal. Es en esta fase, como veremos, en donde se encuentra la esencia de la aproximación jerárquica.

La fase de pre-proceso, que aparece sombreada en el diagrama y etiquetada como *FASE 1*, recibe la escena a procesar y consta, principalmente, de dos pasos. Por un lado, la preparación de la escena de entrada para el cálculo de la iluminación, lo que puede implicar la ejecución de procesos como el mallado de la escena o la triangulización [5] de la misma, entre otros. El siguiente paso, dentro de la fase de pre-proceso, consiste en la construcción de las estructuras auxiliares necesarias para la posterior aceleración del proceso de determinación de la visibilidad que se llevará a cabo a la hora de calcular los factores de forma de las distintas interacciones en la escena. En el Capítulo 2 se trata en profundidad el tema de la optimización en el cálculo de la visibilidad.

El método de radiosidad propiamente dicho comenzaría tras la fase de pre-proceso, y lo componen, por un lado, una fase inicial (*FASE 2* en la Figura 1.7), en la que se determinan las interacciones iniciales entre los objetos de la escena, lo que equivale a la creación de la matriz $\mathbf{K}$ de la Ecuación 1.6; y por otro lado la fase correspondiente al proceso iterativo (*FASE 3* en el diagrama), donde se resuelve el sistema planteado, acercando en cada iteración la solución obtenida a aquella en la que se lograría la distribución total de la energía de la escena.

El método iterativo habitualmente aplicado en radiosidad jerárquica es Gauss-Seidel, lo que equivale a recorrer secuencialmente los elementos de la escena, acumulando la energía recibida en cada uno de ellos por parte de los elementos de la escena con los que interactúa. Los tres pasos típicos que se distinguen en una iteración del método de radiosidad jerárquica sobre cada elemento de la escena son: refinado, propagación de la energía recibida por el elemento y sincronización de su jerarquía (la terminología anglosajona habitual para estos tres pasos de la radiosidad jerárquica es *refinement*, *gathering* y *sweeping*).

La etapa de refinado constituye el núcleo del método jerárquico, al ser la fase a cargo de la decisión de considerar una interacción entre dos elementos como suficientemente precisa o, por el contrario, aconsejar su refinado. La heurística más

[5] Proceso que habitualmente recibe el nombre de *tessellation*

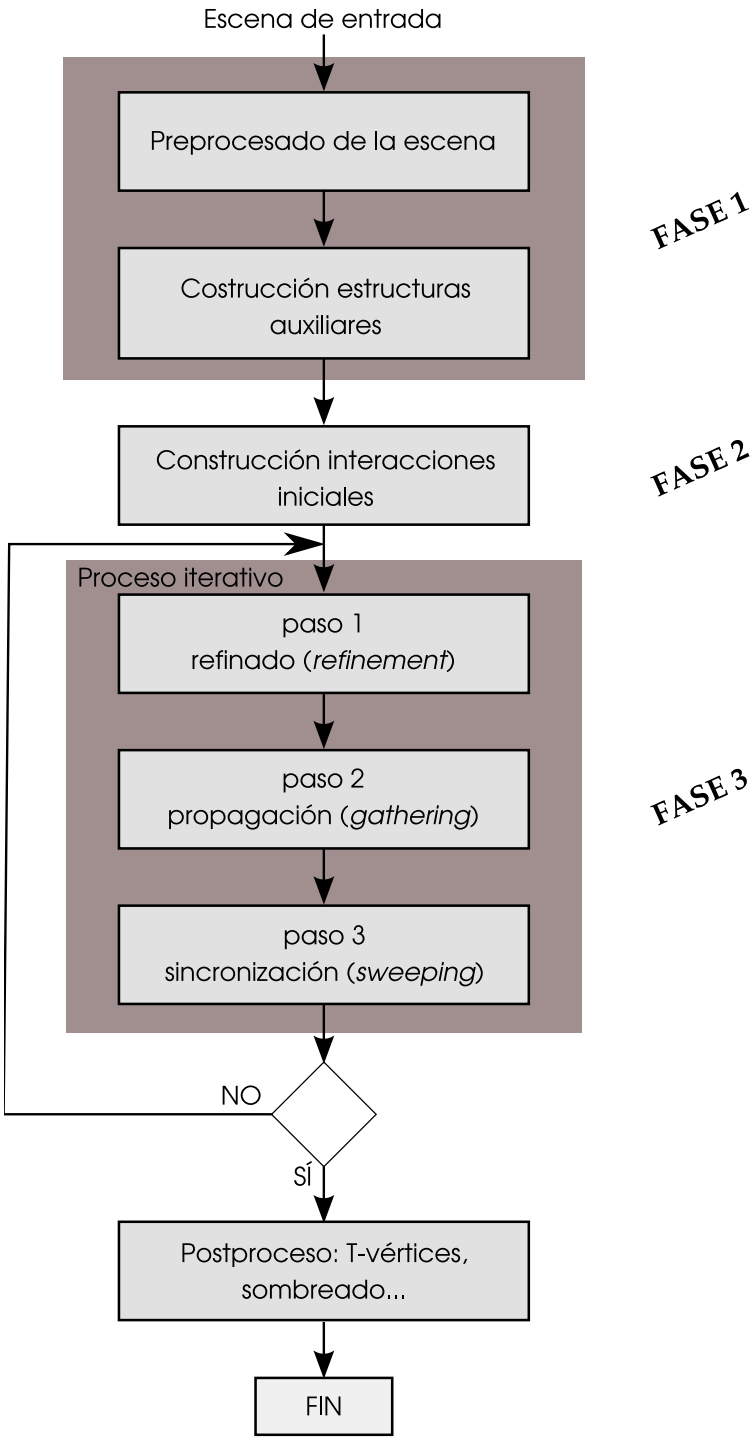


Figura 1.7: Esquema del proceso de refinado jerárquico.

habitual para el refinado jerárquico es la que se conoce como refinado *BF* [64]. Con la aplicación de este método se considera que una interacción entre dos polígonos necesita ser refinada cuando:

1. Esa interacción transporta una cantidad excesiva de energía, de acuerdo a un determinado umbral previamente fijado para la escena. Influyen en el refinado, por tanto, el factor de forma de la interacción, F_{ij}, y la radiosidad que desprende la fuente de la interacción: $B_j \times F_{ij}$.

2. La relación de visibilidad entre los polígonos que están interactuando no está claramente definida, $0 < V_{ij} < 1$. Es decir, las interacciones que no son completamente visibles, ni están totalmente obstaculizadas, tienen una mayor posibilidad de ser refinadas, lo que supondrá un mayor nivel de detalle en la iluminación de las zonas de penumbra.

Cuando una interacción es refinada, uno o los dos elementos involucrados es subdividido, sustituyendo la interacción refinada por interacciones más finas entre los nuevos elementos introducidos. Una vez que todas las interacciones de un elemento han sido evaluadas, el resultado es un árbol jerárquico de elementos surgidos de las sucesivas subdivisiones que recursivamente ha sido necesario realizar sobre el elemento grueso original, que será la raíz de esta jerarquía. A su vez, las interacciones gruesas iniciales, que componían una matriz simétrica, han sido sustituidas por una jerarquía de interacciones entre elementos de distintos niveles de refinado.

Tras el refinado de un elemento, la radiosidad que obtiene cada uno de los elementos de la jerarquía en la que ha sido refinado es calculada y acumulada. Esta energía se obtiene, en cada elemento, a partir de la propagación de la energía transmitida por los elementos con los que interactúa (paso 2 en el diagrama de flujo de la Figura 1.7), siguiendo la siguiente expresión, obtenida de la ecuación clásica de radiosidad (Ecuación 1.3),

$$B_i \mathrel{+}= \sum B_j F_{ij} \tag{1.7}$$

De esta forma, la energía originalmente transportada por las interacciones gruesas del elemento raíz pasa a estar repartida en los distintos elementos de su estructura jerárquica. El tercer paso del proceso iterativo llevado a cabo sobre un elemento, la sincronización de la estructura jerárquica creada, se va a encargar de actualizar los valores de radiosidad en los distintos niveles del árbol, dando coherencia a toda la

estructura. El proceso consiste en recorrer la estructura jerárquica creada y, para cada elemento, por un lado, añadir la energía obtenida en ese nivel a sus descendientes, y por otro lado ponderar la energía recibida en cada rama en la que ha sido refinada para sumarla como propia.

1.3.1. Técnicas de aceleración

El método jerárquico de radiosidad descrito presenta varios puntos críticos y cuellos de botella sobre los que vamos a presentar distintos métodos de actuación a lo largo de este trabajo. Una pequeña relación de los principales problemas que se plantean y las soluciones propuestas para superarlos sería la siguiente:

- La determinación de visibilidad. El cálculo de los factores de forma supone el mayor gasto computacional en el cálculo de la radiosidad de una escena. Concretamente, la obtención del término que se refiere al grado de visibilidad entre dos elementos. El cálculo de los factores de forma se lleva a cabo, primero, en la fase 2 del algoritmo mostrado en la Figura 1.7, en el cálculo de las interacciones iniciales; y después durante la fase 3, cada vez que una interacción debe ser refinada. Para acelerar su cálculo se utilizan habitualmente métodos basados en la creación y explotación de estructuras que indexen el contenido de la escena. Estas estructuras serán precalculadas en la fase 1 del algoritmo, según veremos descrito en el Capítulo 2.

- Demasiados elementos en la escena. Aun con el uso de una jerarquía de elementos, construida a partir del refinado de un conjunto más grueso de elementos de partida que los que se usarían en un método clásico de radiosidad, el número de interacciones entre los elementos iniciales, que sigue siendo de $O(N^2)$ para N elementos de inicio, se vuelve inmanejable a poco que las escenas a procesar crezcan en complejidad. Muchas de esas interacciones iniciales, además, pueden presentar un nivel de detalle excesivo, por lo que la extensión natural al método de radiosidad jerárquica consiste en dejar de considerar los polígonos de entrada como raíces de la jerarquía inicial, ampliando la estructura jerárquica *hacia arriba*.

  La aplicación de técnicas de *clustering* [49, 65, 127, 131] al refinado jerárquico de radiosidad añade la posibilidad de simplificar todavía más la matriz de interacciones, al agrupar objetos cercanos en súper-objetos que puedan actuar como un único elemento cuando esta precisión de interacción sea suficiente.

En el Capítulo 4 proponemos una aproximación a los conceptos de *clustering* basada en la aplicación de técnicas de multiresolución, mediante la obtención de modelos detallados a partir de la subdivisión de superficies gruesas. Los objetos gruesos serán los empleados para el cálculo de la iluminación, mientras que sus versiones suaves proporcionarán un acabado final más realista. En el Capítulo 3 proponemos una solución paralela para los algoritmos de subdivisión recursiva más utilizados.

- Coste computacional y de almacenamiento demasiado elevado. A pesar de la reducción obtenida en los requisitos computacionales y de almacenamiento con las distintas técnicas aplicadas, el procesado de escenas grandes y complejas sigue siendo, en muchas ocasiones, demasiado elevado para máquinas monoprocesador convencionales. En el Capítulo 5 presentamos una solución paralela general al método de radiosidad jerárquica, basada en una aproximación totalmente distribuida.

Capítulo 2

Determinación de la visibilidad

La determinación del grado en el que dos objetos cualesquiera de una escena son visibles (*analytic visibility*) es una de las operaciones más costosas, sino la más, dentro del cálculo de la iluminación global de una escena. Se trata, por tanto, de un cálculo especialmente crítico para el buen rendimiento del modelo de iluminación global aplicado sobre la escena, por lo que su resolución eficiente es de especial importancia, sobre todo en escenas complejas. En cualquier caso, no debe confundirse el cálculo de la visibilidad analítica entre dos objetos, que es el tratado en este capítulo, con la determinación de las superficies visibles a la hora de *renderizar* una escena desde un punto de vista concreto. Este último problema se presenta en una fase posterior del proceso de *rendering*, y utiliza normalmente técnicas derivadas del uso del *Z-Buffer* [22, 134] y tratamiento de transparencias [3, 145].

El cálculo de la visibilidad es de vital importancia en muchos campos de la informática gráfica: realidad virtual, modelado geométrico, visión artificial, etc. En el caso de los métodos de iluminación global, la determinación de la visibilidad entre objetos es habitualmente obtenida mediante alguna técnica derivada del lanzamiento de rayos (*ray-casting* [115]) [11, 66, 15]. Así, consideramos como una estimación de la relación de visibilidad entre dos polígonos la proporción de rayos lanzados entre ambos que no son interceptados por ningún otro polígono de la escena. Esto supone la evaluación, para cada rayo, de un test de intersección rayo–escena, lo que genera el verdadero cuello de botella en el cálculo de la visibilidad (y en general en cualquier método basado en el lanzamiento de rayos [143]).

A lo largo de todo este capítulo se esbozarán algunas de las diferentes técnicas existentes para solucionar el problema que supone tratar de acelerar el proceso de

trazado de los rayos en una escena. Así, la Sección 2.1 sirve como presentación y resumen de las diferentes aproximaciones existentes. Tras una categorización de las mismas, en el resto del capítulo nos vamos a centrar en el conjunto de técnicas de aceleración que denominamos como de *menos intersecciones rayo–objeto*. Así, en las Secciones 2.2, 2.3 y 2.4 se comentan los aspectos más relevantes, y se describe el funcionamiento, de algunas de las principales técnicas de aceleración existentes dentro de esa clase de métodos. El análisis resultante, tras la implementación de varias de estas técnicas, también puede consultarse en [108].

Dado que la aceleración obtenida en la determinación de la visibilidad con los métodos existentes todavía está lejos de ser plenamente satisfactoria en todos los casos, en este capítulo proponemos un nuevo método de aceleración que complementaría a los ya comentados. Se trata del Método de Coherencia Angular de los Rayos (*Angular Coherence of Rays*, ACR) [105], que se enmarcaría dentro de los métodos que explotan la coherencia en el espacio direccional de los rayos. En este caso, se combina la idea de realizar una partición del espacio de direcciones con la creación de clases de equivalencia de rayos, lo que permite reutilizar aquellos rayos que puedan considerarse equivalentes, evitando así la duplicidad de muchos de los cálculos de intersección rayo–escena.

Como en el caso de la subdivisión espacial, distintos tipos de particiones son posibles sobre el hemisferio de direcciones, aunque en el caso de los métodos direccionales las restricciones de memoria cobran especial importancia, como veremos. El método ACR ha sido probado en este trabajo con dos configuraciones diferentes, una configuración de rejilla regular de un único nivel y otra más adaptable de dos niveles, que permite un ajuste más fino con un ahorro considerable de memoria.

La Sección 2.5 presenta el nuevo método propuesto, en las dos variantes comentadas, mientras que en la Sección 2.6 se analizan los resultados obtenidos con su utilización. En esta última sección se muestran, además, los resultados obtenidos con las pruebas de las diferentes técnicas de aceleración descritas a lo largo de todo el capítulo, comentando las ventajas e inconvenientes de las diferentes aproximaciones. Todos los métodos empleados han sido implementados para la resolución de las consultas de visibilidad dentro de un algoritmo de radiosidad jerárquica, objetivo del presente trabajo.

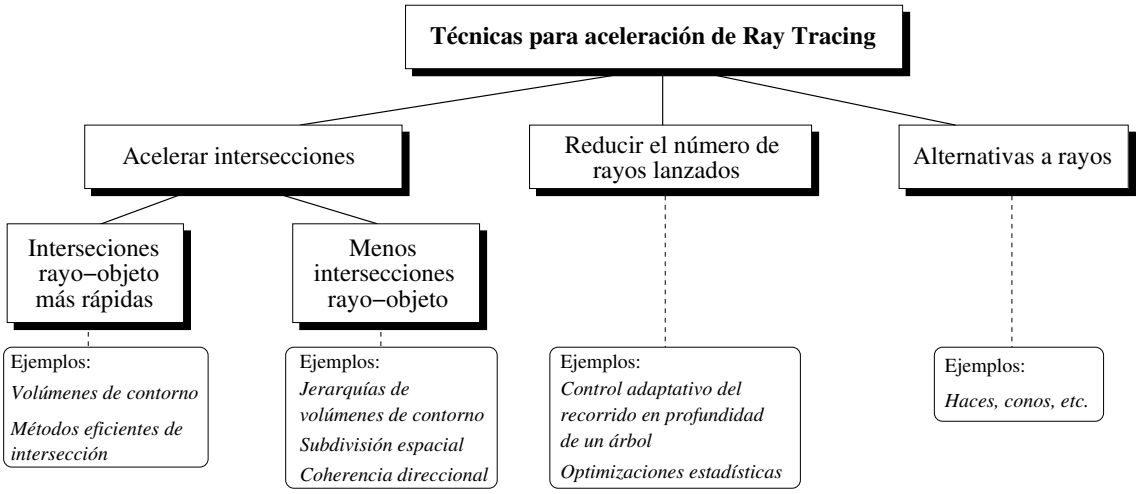


Figura 2.1: Clasificación de Arvo y Kirk de técnicas para la aceleración de *ray tracing*.

2.1. Técnicas para la aceleración del trazado de rayos

Conseguir una reducción efectiva del tiempo empleado en resolver consultas acerca de la visibilidad de los objetos en la escena requiere el tratar de acelerar la fase de evaluación de los rayos lanzados a tal efecto. Atendiendo a la clasificación ofrecida por Arvo y Kirk [11] (ver Figura 2.1), las posibles mejoras a introducir pasan, bien por reducir el tiempo consumido en la evaluación de cada rayo trazado, bien por reducir el número de rayos lanzados, o bien por reemplazar el uso de rayos por el de otras entidades con volumen, más generales, y de las cuales los rayos no sean más que un caso particular.

Dentro del primer grupo encontramos dos subcategorías: *intersecciones rayo–objeto más rápidas* [143, 96] y *menos intersecciones rayo–objeto* [32, 66]. En el primer caso se trata de simplificar, y por lo tanto acelerar, el cálculo de la intersección geométrica entre un rayo y un objeto, lo que engloba la utilización de técnicas eficientes de implementación de las intersecciones rayo–polígono, que habitualmente se ayudan de la utilización de sencillas formas geométricas de contorno que simplifiquen los cálculos. En la segunda categoría agrupamos aquellos métodos que tratan de minimizar el número de objetos a comprobar para cada rayo lanzado, reduciendo la cantidad de intersecciones totales que es necesario calcular.

Respecto a la segunda categoría general (*reducir el número de rayos lanzados*), los

métodos a los que engloba tratan de reducir, normalmente desde una aproximación estadística, el número de rayos que es necesario lanzar en una escena [85, 16, 79]. Se trata de métodos principalmente dirigidos a la resolución completa de la ecuación de *rendering* mediante la aproximación del método de *ray tracing*, y no se adecúan a la simple determinación de la visibilidad entre polígonos necesaria utilizando el modelo puro de radiosidad, por lo que no profundizaremos más en su análisis.

Los métodos del último grupo, que tratan de buscar alternativas al lanzamiento de rayos [67, 69], presentan ciertas propiedades interesantes junto con nuevas complicaciones, como las de imponer ciertas restricciones a los métodos y a las primitivas geométricas a utilizar y producir intersecciones que no están bien definidas o que son difíciles de calcular. Estos métodos tampoco son considerados en este trabajo.

En nuestro análisis nos vamos a centrar en la primera categoría de técnicas de aceleración, concretamente en la subcategoría de *menos intersecciones rayo–objeto*, aunque, por supuesto, el uso de algoritmos de intersección eficientes de la subcategoría *intersecciones rayo–objeto más rápidas* se encuentra implícito.

Para homogeneizar la descripción de las diferentes técnicas de aceleración utilizadas, vamos a considerar el concepto abstracto y formal de un *optimizador* [114] como una función que devuelve la lista de objetos de la escena con los que es necesario realizar el test de intersección rayo–polígono para un determinado rayo, es decir, un método que nos permite descartar comprobaciones de intersección rayo–polígono innecesarias.

La aplicación de un optimizador para la determinación de la visibilidad en una determinada escena requiere, por un lado, de la creación y mantenimiento de una cierta organización o indexación espacial en el dominio de la escena o en el dominio de los rayos, y por otro lado de la existencia de mecanismos que aprovechen toda esa información mantenida. Es decir, se trata de «recorrer» de forma adecuada la estructuras creadas, acelerando así las consultas sobre la visibilidad entre objetos de la escena.

Las distintas funciones de optimización plantean diferentes estrategias para tratar de restringir la búsqueda de posibles intersecciones a aquellos objetos que están en la ruta de cada rayo lanzado, acelerando así el proceso de lanzamiento de rayos. Para ello implementan diferentes organizaciones espaciales a partir de tres conceptos básicos, como son el uso de estructuras espaciales jerárquicas, la subdivisión del espacio y los principios de coherencia y localidad. Así, siguiendo con la clasificación de Arvo y Kirk, distinguimos tres grandes grupos de métodos. Por un lado están los

métodos basados en la construcción de jerarquías de volúmenes de contorno sobre los objetos de la escena: los objetos se agrupan formando objetos más complejos de forma recursiva. Casi en contraposición, pero todavía en el espacio de la escena, tenemos los métodos que aplican algún tipo de división espacial o partición a la escena, división que puede ser o no jerárquica: se obtienen los objetos de cada partición a partir de la selección y división del volumen de la escena. Un tercer grupo lo compondrían una serie de métodos que, en lugar de indexar el dominio de la escena, con el consiguiente aprovechamiento de la coherencia en la localidad espacial de los objetos, trabaja en el dominio de los rayos, utilizando la coherencia en el espacio de direcciones. Una denominación común para este conjunto de métodos es el de métodos direccionales (*directional techniques*).

En las siguientes secciones se describen algunos de los métodos de optimización más representativos de cada uno de los tres grupos comentados, junto con las implementaciones realizadas de ellos.

2.2. Métodos basados en jerarquías de volúmenes de contorno

La utilización de primitivas geométricas simples como volúmenes de contorno que rodeen a objetos más complejos (*Bounding Volumes*, *BVs*), con el propósito de simplificar las intersecciones entre objetos o rayo–objeto [143], es, desde hace tiempo, práctica habitual en informática gráfica. Así, empleando volúmenes de contorno con formas geométricas simples (esferas, cubos...) se logra un cálculo rápido de intersección. La ampliación de esta idea introduce el concepto de jerarquía de volúmenes de contorno (*Bounding Volumes Hierarchies*, *BVHs*).

Aunque la idea de una organización jerárquica de los objetos de una escena surgió como una optimización del proceso de *clipping* [27], pronto comenzó su aplicación a la aceleración del lanzamiento de rayos [116]. Se logra así, para cada rayo trazado, reducir el número de intersecciones rayo–objeto a comprobar. Los volúmenes inicialmente empleados eran paralelepípedos rectangulares, popularmente denominados cajas de contorno (*Bounding Boxes*, *BBs*), con la orientación adecuada para ajustarse lo más posible a su contenido y minimizar su tamaño. Para optimizar las comprobaciones de intersección rayo–caja, los rayos lanzados eran habitualmente trasladados al espacio de coordenadas de la caja.

Además de las cajas de contorno orientadas con orientación arbitraria (*Oriented Bounding Boxes*, *OBBs*), otros ejemplos de BVs comúnmente empleadas son esferas [68], cajas de contorno alineadas con los ejes (*Axis-Aligned Bounding Boxes*, *AABBs*) [139], que son un caso particular de las orientadas, simplificando el cálculo de intersección pero perdiendo capacidad de ajuste al objeto que encierra (y por lo tanto encerrando más espacio del que sería necesario con una caja orientada), y k-DOPS (*Discrete Oriented Polytope*) [81], que son una extensión de las BBs en la que se define un volumen convexo con el hueco dejado por una serie de parejas de planos enfrentados.

La esencia de una BVH es un árbol jerárquico, que encierra en cada nodo regiones del espacio que rodea a los objetos de la escena. No crea, por lo tanto, una subdivisión del espacio, con lo que no todo el volumen de la escena se encuentra encerrado en el árbol. En el campo de la informática gráfica, las BVHs son utilizadas, además de en la aceleración del test de intersección rayo–escena, para agrupar objetos de la escena jerárquicamente, creando distintos niveles de detalle en función de la precisión necesaria para su representación (conceptos de *clustering* [82, 131, 65] y nivel de detalle, *LOD* [87]) así como en el modelado geométrico [18, 8].

El principal problema de las BVHs no radica tanto en su explotación, que básicamente consiste en el aprovechamiento de la localidad espacial de los objetos de la escena, como en su construcción. Alcanzar de forma automática una organización jerárquica «lógica» de los objetos en la escena no es ni mucho menos una tarea trivial. Además, una organización jerárquica adecuada para, por ejemplo, la aceleración de intersecciones, puede no ser la más idónea para agrupar objetos en *clusters* de cara al cálculo de su factor de forma en el método de radiosidad, pues las métricas a minimizar son distintas en uno y otro caso. Existen, por lo tanto, métodos específicos para la construcción de estructuras jerárquicas orientadas a usos concretos.

Básicamente se puede considerar que son dos las maneras de abordar la construcción de una jerarquías de volúmenes: una aproximación «desde abajo hacia arriba» (*Bottom-up*), que consiste en formar objetos más grandes agrupando objetos pequeños (*clustering*), y una aproximación «desde arriba hacia abajo» (*Top-down*), en la que son los grupos de objetos los que son divididos en subgrupos.

La construcción *bottom-up* de BVHs es, sin duda, inherentemente más compleja, pues requiere examinar las relaciones espaciales entre los diferentes objetos y utilizar algún tipo de heurística para agruparlos de la manera más conveniente. Los dos métodos de referencia para la aproximación «desde abajo hacia arriba» son el de Kay y

Kajiya [77], que mantiene una lista ordenada de polígonos para acelerar la búsqueda de la intersección más cercana, utilizando paralelepípedos de múltiples caras para intentar ajustar los volúmenes al contenido, introduciendo el concepto de «conjunto de planos» (*plane-set*), y el de Goldsmith y Salmon [54], que construye la jerarquía de forma incremental, basándose en una métrica de probabilidad de intersección de la jerarquía con un rayo arbitrario y penalizando los aumentos de tamaño de las BVs a la hora de introducir cada nuevo objeto en la jerarquía. Este último método, creado para acelerar el proceso de intersección rayo–escena, ha servido así mismo como base de otros métodos utilizados en radiosidad para la creación de grupos de objetos (*clusters*) [131].

Por su parte, la aproximación *top-down* utiliza los conceptos y métodos de los algoritmos de subdivisión espacial (descritos en la siguiente sección) para obtener la jerarquía de la escena, normalmente subdividiendo primero el espacio y luego ajustando los *voxeles* obtenidos a la geometría que contienen. Así, los típicos algoritmos de construcción «desde arriba hacia abajo» de BVHs utilizan estructuras puras de subdivisión espacial, como árboles k-D (*k-D trees*) [127] (ver Sección 2.3.1), o variantes como un árbol 8-ario maleable, con las fronteras entre celdas relajadas, que ajusta el espacio de los *voxeles* a su contenido y no a la inversa (*Tight-fitting octree*, *TF-OCT*) [26]. Precisamente esta última, por su simplicidad y rapidez de construcción, es la alternativa que hemos implementado en este trabajo y que describimos a continuación.

2.2.1. Árboles 8-arios ajustados al contenido (TF-OCT)

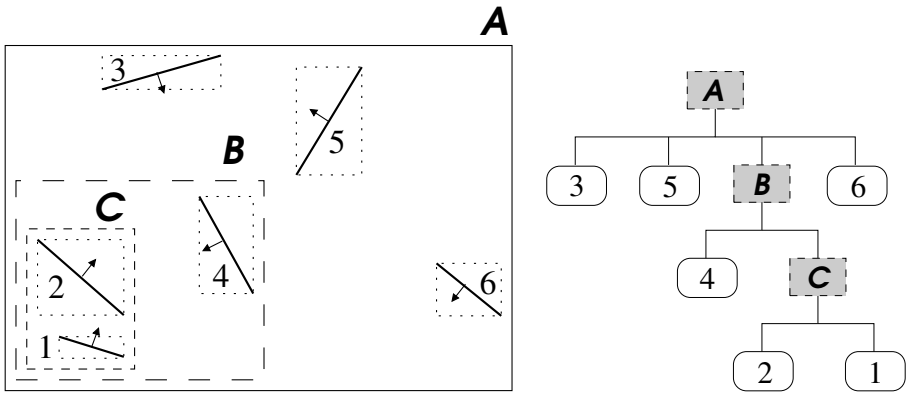
En la Figura 2.2 se muestra el pseudocódigo del algoritmo de construcción de un árbol 8-ario ajustado al contenido (TF-OCT). En cada iteración partimos de un entorno a subdividir en 8 celdas, en principio iguales. Los polígonos del entorno son recorridos (línea 4) y clasificados en función de su tamaño y posición respecto a las celdas hijas en las que ha sido partido el espacio (líneas 5-10). Así, los polígonos cuyo tamaño es excesivo para las celdas hijas son asignados a la celda padre (línea 5). En otro caso los polígonos se asignan a la celda hija que contenga a su centroide, sin tener en cuenta las fronteras de la celda (líneas 6 a 10). Una vez procesados todos los polígonos se actualizan las fronteras de las celdas de acuerdo a su contenido. El proceso se repite recursivamente para cada nueva celda hija (línea 13) hasta alcanzar una cierta profundidad del árbol o un número mínimo de polígonos por nodo (línea 2). El resultado final será una jerarquía de volúmenes/*voxeles* cuyas

```
1  crear_bvh ( *nodo, prof )

2       if (nodo→n_pols ≤ MIN_POLS) or (prof ≥ MAX_PROF)
3          return;

4       foreach p in nodo→lista_pols do
5           if (p→area < nodo→area / 8)
6               octante = 0;
7               if (nodo→centro.x - p→centro.x > 0)   octante + = 1;
8               if (nodo→centro.y - p→centro.y > 0)   octante + = 2;
9               if (nodo→centro.z - p→centro.z > 0)   octante + = 4;
10              añadir_pol (p, nodo→hijos[octante]);

11      for i = 0 to 8 do
12          calcular_contorno (nodo→hijos[i]);
13          crear_bvh (nodo→hijos[i], prof + 1);
```

Figura 2.2: Construcción de una jerarquía de volúmenes de contorno mediante un TF-OCT.

cajas de contorno pueden solaparse.

En la Figura 2.3 se muestra una sencilla escena en 2D (Figura 2.3a) y la jerarquía, en este caso un árbol cuaternario, obtenida con la aplicación del algoritmo de construcción descrito (Figura 2.3b). En la figura, los polígonos se etiquetan con números (del 1 al 6) y las celdas que contienen más de un polígono con letras mayúsculas (A, B y C). Comenzando con un espacio que contiene a un conjunto de polígonos (inicialmente al total de polígonos de la escena), se aplica una subdivisión en cada una de las dimensiones (x e y en este caso) a través del punto central del espacio, obteniendo cuatro celdas hijas de igual tamaño. Los polígonos se asignan a las celdas correspondientes y se actualizan las cajas de contorno de cada nodo. El proceso se aplica de forma recursiva hasta alcanzar alguna condición de finalización, como un mínimo de un elemento por celda en nuestro ejemplo. Así, tras la primera iteración, solo la celda etiquetada como B (correspondiente al cuadrante inferior izquierdo) necesita un nuevo paso de refinado, ya que los polígonos 3, 5 y 6 disponen de su propia celda. Análogamente, dos nuevas celdas son necesarias dentro de la celda B, aunque solamente el subespacio etiquetado como C requiere un último paso en el proceso de partición.



(a) (b)

Figura 2.3: Ejemplo de construcción de una jerarquía de volúmenes de contorno: (a) escena de ejemplo 2D (b) BVH (árbol cuaternario ajustado al contenido).

2.3. Métodos basados en la subdivisión del espacio de la escena

La filosofía de los métodos de subdivisión espacial es la de partir el volumen total de la escena en celdas o *voxeles*, de manera que la suma total de todas las celdas es igual al volumen total de la escena. Estos métodos ponen especial énfasis en el concepto de «espacio» en contraposición al de «objeto», como hacían los métodos basados en BVHs. Además, y también a diferencia de los BVHs, los métodos de subdivisión espacial mantienen de forma implícita, debido a su propia naturaleza, un orden espacial entre los polígonos de la escena.

Existe un buen número de diferentes estructuras espaciales utilizadas para organizar la subdivisión de una escena, pero una primera clasificación puede llevarnos a distinguir entre las indexaciones que producen una subdivisión no uniforme del espacio, como los árboles de partición binaria (*Binary Space Partitioning trees* o *BSP trees*) [43, 100, 93] o los árboles 8-ario (*octrees*) [51, 138]), y las que se basan en una partición del espacio en cuadrículas o rejillas de celdas regulares (*3D Grids*) como la popularizada por Fujimoto *et al.* [44] bajo el nombre de SEADS (*Spatially Enumerated Auxiliary Data Structure*).

En nuestro caso hemos optado por la utilización de árboles BSP, cuyo principal atractivo radica en su simplicidad, tanto de concepto como de implementación. Esta simplicidad y una gran versatilidad han extendido su uso y su estudio, pudiéndose encontrar numerosos trabajos ampliando y mejorando el diseño original de Fuchs *et*

al. [43]. De hecho, esta estructura es probablemente, no solo el método de subdivisión más ampliamente utilizado, sino también una de las estructuras de aceleración más extendidas. Las principales características de esta técnica, así como los diferentes tipos existentes y los detalles de la implementación realizada en este trabajo se muestran a continuación.

2.3.1. Árboles BSP

Un árbol de partición binaria del espacio (*Binary Space Partitioning Tree*, *BSP Tree*) es una estructura de subdivisión espacial utilizada para resolver un amplio número de problemas dentro de la informática gráfica. Se trata de una estructura jerárquica que mantiene una partición binaria recursiva del espacio, de manera que en cada nodo del árbol se guarda un plano que divide el espacio de ese nodo en dos partes. El resultado es una jerarquía de *voxeles* que recorrida de forma adecuada permite un acceso ordenado a su contenido desde cualquier punto de vista (en contraste con las BVHs que no mantienen ese tipo de ordenación espacial).

El problema a resolver, y objetivo inicial, cuando se presentó por primera vez la idea de un árbol BSP [42, 43], era el de ordenar los polígonos de una escena para incrementar así el rendimiento en el proceso de eliminación de las superficies ocultas a la hora de mostrar la escena. El motivo era la todavía nula existencia de implementaciones *hardware* que aceleraran el uso de un *Z-Buffer*, cuyo mantenimiento por *software* consumía demasiado tiempo. Para el desarrollo de esta técnica Fuchs *et al.* se basaron en los trabajos que Schumacker *et al.* habían desarrollado en el campo de la representación de las superficies visibles [122, 136].

Aunque en la actualidad la utilidad para la que los árboles BSP fueron inicialmente pensados ha quedado ampliamente superada por los múltiples desarrollos *hardware* en ese campo, los árboles BSP se han mostrado extraordinariamente útiles y versátiles en diferentes áreas dentro (y fuera) de la informática gráfica, como por ejemplo el caso que nos ocupa de la aceleración en la determinación de la visibilidad entre polígonos por medio del lanzamiento de rayos, la representación de modelos poligonales [137, 100] o la detección de colisiones [93, 7].

Según el tipo de planos utilizados para realizar las particiones binarias se distinguen principalmente dos tipos de árboles BSP: los que subdividen el espacio utilizando planos alineados con los ejes (*Axis-Aligned BSP Trees*, *AA-BSP*) [75, 66] (ver Figura 2.5), también conocidos como árboles BSP rectilíneos u ortogonales,

```
1  crear_bsp ( *nodo, prof, *escoger_divisor() )

2       if (nodo→n_pols == MIN_POLS) or (prof == MAX_PROF)
3          return;

4       divisor = escoger_divisor (nodo);

5       partir_polígonos (divisor, nodo);

6       asignar_polígonos (divisor, nodo, nodo->izq, nodo->dcha);

7       crear_bsp (nodo→izq, prof + 1, escoger_divisor);
8       crear_bsp (nodo→dcha, prof + 1, escoger_divisor);
```

Figura 2.4: Construcción de un árbol BSP.

y los que utilizan planos arbitrariamente orientados, escogiendo como divisores los planos de los propios polígonos de la escena (*Polygon-Aligned BSP Trees*, *PA-BSP*) [42, 43, 1] (ver Figura 2.7).

El procedimiento general para la construcción de un árbol BSP es básicamente el mismo para cualquiera de sus variantes, y se muestra en la Figura 2.4. El espacio de la escena es recursivamente subdividido en dos subespacio (líneas 7-8) hasta satisfacer una determinada condición de finalización (línea 2), que podría ser, por ejemplo, alcanzar una profundidad máxima o un número mínimo de elementos en el subespacio. Cada iteración supone escoger el plano divisor para un *voxel* (línea 4) y asignar los polígonos a cada uno de los dos subespacios creados (línea 6). En cuanto a qué hacer con los polígonos que son atravesados por el plano divisor que ha sido escogido (línea 5), cualquiera de los dos soluciones posibles puede ser buena, dependiendo de si lo que se prima es el conseguir un árbol equilibrado (en cuyo caso los polígonos son partidos y los trozos asignados a los subespacios correspondientes), o evitar un aumento significativo del número de polígonos en la escena (los polígonos podrían ser asignados directamente al nodo padre o bien a ambos nodos hijos en lugar de ser partidos).

En cuanto a la forma de recorrer el árbol a la hora de atender una consulta sobre la visibilidad entre dos polígonos, se propusieron inicialmente métodos basados en un recorrido secuencial de la estructura creada, tanto para árboles 8-arios [51] como BSPs [75]. Frente a este tipo de recorridos, surge posteriormente una alternativa recursiva [72], en la cual se han basado las implementaciones tanto para el AA-BSP como para el PA-BSP de este trabajo. Básicamente, se trata de un recorrido en-

orden, comenzando en el nodo que contiene el origen del rayo lanzado, recorriendo primero el subárbol positivo y luego los subárboles no recorridos de los ancestros, siempre en un orden desde el más cercano al más lejano, y podando recursivamente las ramas ocultas al rayo. El recorrido finaliza cuando se alcanza el polígono destino del rayo. Por supuesto, la búsqueda de un obstáculo para el rayo mostrará una mayor eficiencia cuanto más balanceado sea el árbol de partición (a igual número de nodos, claro).

Los principales inconvenientes de los árboles BSP son su carácter irregular, más acentuado en el caso de los árboles PA-BSP, la obtención de una subdivisión en cierto modo arbitraria del espacio, y el hecho de ser una estructura eminentemente estática y difícil de modificar una vez construida. Como se puede apreciar en la Figura 2.5, la utilización de planos alineados con los ejes como divisores del espacio proporciona una partición del espacio más regular que el uso de los planos de los propios polígonos.

La obtención de un árbol binario que permita resolver las consultas posteriores de forma eficiente depende casi por completo del criterio empleado para escoger el plano divisor en cada *voxel*, teniendo en cuenta que conviene mantener un árbol BSP equilibrado (para recorrer el menor número de nodos posible en cada consulta), minimizar el número de polígonos a partir (para evitar añadir complejidad a la escena inicial) y seleccionar el plano divisor en un tiempo razonable.

Árboles BSP alineados con los ejes (AA-BSP)

En el caso de los árboles AA-BSP, la subdivisión de una celda implica primero escoger la dimensión en la que subdividir (x, y o z en 3D), para luego elegir el punto por el que trazar el plano divisor. Un criterio muy utilizado es el de escoger sucesivamente cada una de las dimensiones de forma cíclica: si el padre es subdividido en la dimensión x, el hijo lo es en la y, el nieto en la z y el ciclo se repite. Los árboles BSP construidos de esta manera se conocen también por el nombre de árboles k-D (*k-D trees*). Otra estrategia posible consiste en partir el espacio siempre por la dimensión más larga de la caja. Respecto a la elección del punto por el cual trazar el plano divisor, la solución ideal es la que trata de obtener un resultado equilibrado entre el número de elementos a ambos lados del plano, minimizando en lo posible el número de polígonos atravesados por el plano divisor. Sin embargo, es común rebajar el coste de esta operación para obtener árboles BSP de forma rápida escogiendo directamente el punto central de las celdas para trazar el plano divisor.

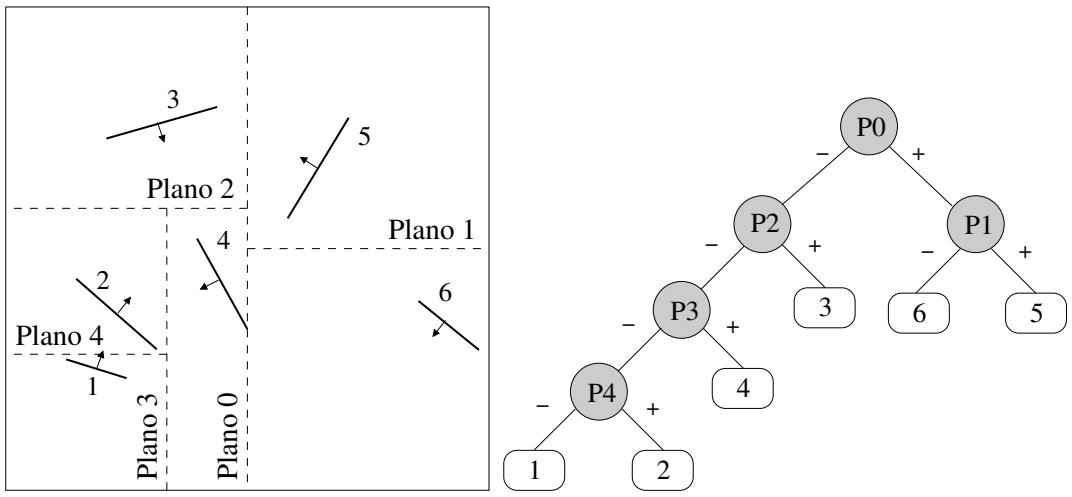


Figura 2.5: Ejemplo de árbol BSP alineado con los ejes: (a) escena de ejemplo 2D (b) árbol AA-BSP.

En el ejemplo en 2D mostrado en la Figura 2.5 los ejes x e y son alternativamente subdivididos, intentando que los planos divisores atraviesen el menor número de polígonos posibles. El criterio de finalización para la subdivisión recursiva es, en este caso, la obtención de un único polígono por celda. El espacio total inicial es primero subdividido a través del eje x por el plano 0, como se puede observar en la Figura 2.5a. A continuación, cada una de las mitades obtenidas es partida en dos por el eje y, cada una en un punto diferente (planos 1 y 2), evitando atravesar ningún polígono. Solo el *voxel* que contiene varios polígonos (polígonos 1, 2 y 4) necesita de una mayor subdivisión. El espacio de esta celda es subdividido en el eje x (plano 3) y, por último, en el subespacio resultante que todavía conserva más de un polígono se traza el plano 4 cortando el eje y. El árbol AA-BSP obtenido se muestra en la Figura 2.5b.

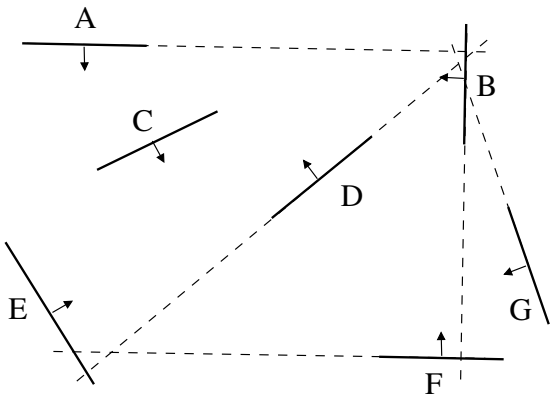
Árboles BSP alineados con los polígonos (PA-BSP)

Respecto a la construcción de un árbol PA-BSP, los planos de los propios polígonos de la escena (por lo tanto con una orientación arbitraria respecto a los ejes) son los utilizados para subdividir el espacio. En este trabajo hemos implementado varias estrategias diferentes de selección del polígono divisor [97, 70, 71], que a continuación pasamos a analizar. La primera y más sencilla consiste en utilizar como criterio de selección el orden de los polígonos de entrada en la descripción de la escena. Esta

estrategia estaría dentro de lo que se conoce como un nivel de profundidad de análisis 0 o trivial, que consiste en una selección arbitraria (y muy rápida) de cada plano a utilizar, con la desventaja de que la partición binaria final obtenida depende en gran medida del azar. Concretamente, siguiendo la estrategia comentada de utilizar el orden de los polígonos de entrada para la creación del árbol binario, se puede obtener una partición final de la escena de bastante calidad si se puede garantizar previamente un cierto orden en los polígonos de entrada, de forma que los objetos más grandes y periféricos se utilicen antes para evitar el mayor número posible de subdivisiones de polígonos (lo que implicaría probablemente un preproceso previo guiado por el usuario).

Un nivel de profundidad de análisis 1 significa que se analizan las posibles consecuencias que en el nivel inmediatamente siguiente tendría la elección de un determinado polígono como divisor. En este nivel de análisis están criterios como la selección del polígono cuyo plano corta al menor número posible de otros polígonos en su subescena, es decir, el que provoca el menor número de cortes posibles (*least-crossed*) [43] o la selección del polígono que es cortado por los planos de los demás el mayor número de veces (*most-crossed*) [97], evitando así que esos potenciales cortes le lleven a cabo posteriormente. Otra heurística con nivel de análisis 1 sería buscar siempre el divisor que proporciona un resultado más balanceado en el siguiente nivel del árbol. En la Figura 2.6 se muestra un ejemplo de estas estrategias con profundidad de análisis 1. Como vemos, en nuestro ejemplo o bien el polígono C o bien el polígono E sería el seleccionado por el método *least-crossed* (*lc* en la figura) para actuar como divisor del subespacio, pues sus planos no producen ningún corte en el resto de la subescena, con lo que no sería necesario partir ningún polígono en este paso de la iteración. Con el método *most-crossed* se escogería el polígono B, que es el más cruzado por los planos del resto de la escena. Escogiéndolo, se garantiza que esos cortes son evitados en la construcción del resto del árbol. Un criterio puro de equilibrio nos llevaría a escoger el polígono D, que deja igual número de elementos a ambos lados de su plano y produciría dos subespacios perfectamente equilibrados.

Es habitual, además, encontrarse con combinaciones de dos o más criterios de nivel 1, sirviendo uno como base para seleccionar un conjunto de polígonos entre los que escoger utilizando algún otro criterio. Así, son frecuentes combinaciones como: *least-crossed with most-crossed tie breaking*, que selecciona los polígonos que producen un menor número de cortes y resuelve los empates seleccionando el más cortado; o *most-crossed with least-crossed tie breaking*, que actúa justo al contrario que el anterior; desempates con criterios de equilibrio (*most-balanced tie-breaking*),



lc	mc	equ
A: 1	A: 0	A: +5
B: 1	B: 3	B: +3
C: 0	C: 0	C: +4
D: 2	D: 0	D: 0
E: 0	E: 2	E: +6
F: 1	F: 1	F: +5
G: 1	G: 0	G: +5

Figura 2.6: Ejemplo de análisis con un nivel de profundidad 1: *least-crossed* (*lc*) *most-crossed* (*mc*) máximo equilibrio (*equ*).

entre otros casos. En la Figura 2.7 se presentan dos árboles PA-BSP diferentes para una escena en 2D sencilla (Figura 2.7a). En el primer caso (Figura 2.7b) se prima la obtención de un árbol lo más equilibrado posible, mientras que en el segundo (Figura 2.7c) se trata de minimizar el número de polígonos a partir como prioridad principal (usando en este caso una estrategia *least-crossed with most-balanced tie breaking*).

Aumentando el nivel de profundidad de análisis a la evaluación de las consecuencias de nuestra elección un nivel más tenemos métodos más sofisticados, como el de minimización de conflictos (*Conflict Minimisation*) [43, 70], que combina el método *least-crossed* con la maximización en la eliminación de conflictos entre polígonos en el siguiente nivel. En [70, 71] se extendió la idea de la minimización de conflictos a la profundidad de análisis deseada, con el método *Conflict Neutralisation*. Se considera que la selección de un polígono como divisor elimina un conflicto cuando su plano deja a uno de sus lados a un polígono cuyo plano corta a otro polígono que se encuentra al otro lado del plano seleccionado. Así, en el método de minimización de conflictos (nivel 2) se selecciona el polígono que produce un menor número de cortes y evita, con la subdivisión del espacio creada, el mayor número de divisiones potenciales en la siguiente iteración. En la Figura 2.8 se muestra un ejemplo de selección del plano divisor mediante minimización de conflictos. En la tabla de la figura aparecen, para cada polígono candidato a ser seleccionado, los polígonos que quedan en el subespacio positivo (*pos*) y en el negativo (*neg*) del plano del polígono divisor, así como los que son cortados por este (*cor*). Así, para cada candidato se obtiene un valor numérico, que representa la bondad de la partición con él obtenida, como la suma de los conflictos que elimina su selección (cortes en polígonos de

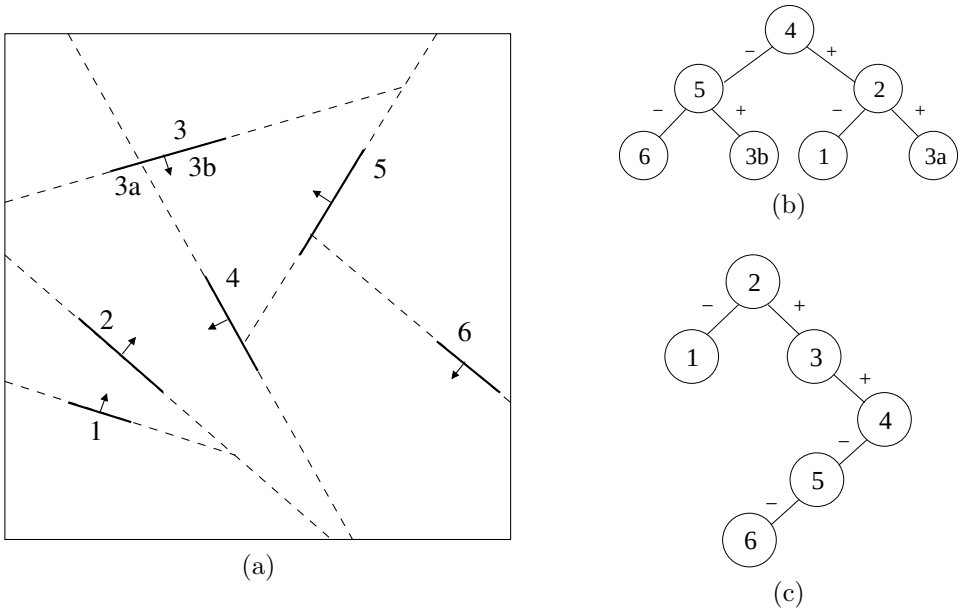
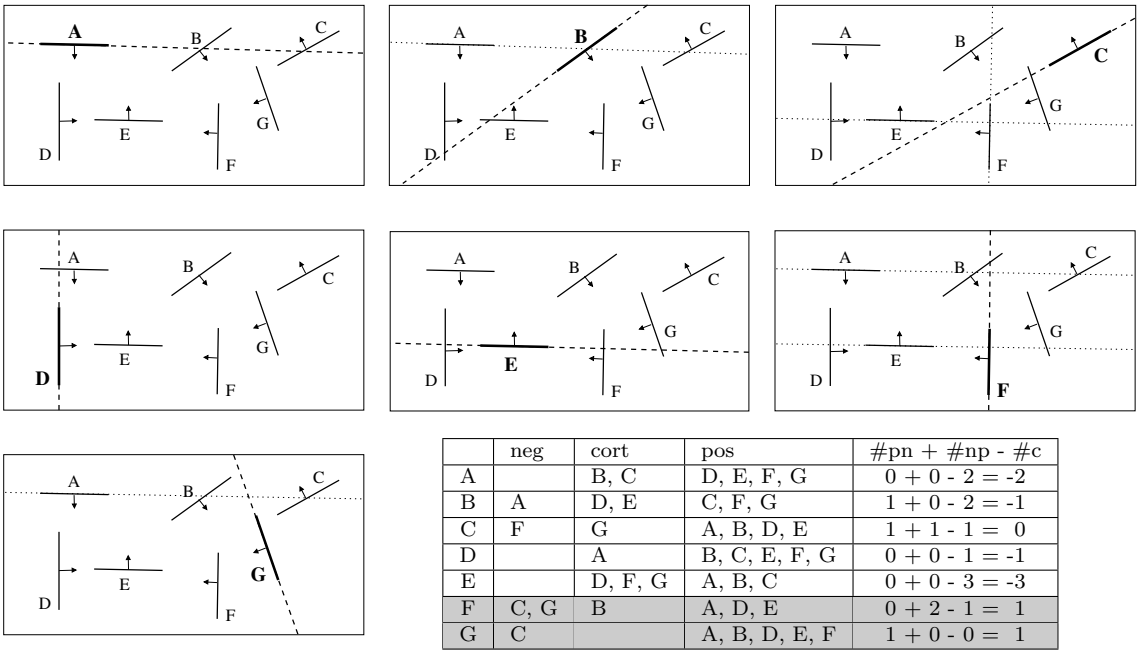


Figura 2.7: Ejemplo de árbol BSP alineado con los polígonos: (a) escena de ejemplo 2D (b) árbol PA-BSP (criterio de maximización del equilibrio) (c) árbol PA-BSP (criterio de minimización del número de polígonos).

su subespacio positivo que serían provocados por los planos de los polígonos en su subespacio negativo, $\#pn$, y viceversa, $\#np$) menos el número de conflictos (cortes) que provoca directamente ($\#c$). En nuestro ejemplo el mejor valor lo obtienen los polígonos F, que provoca el corte del polígono B y elimina los conflictos entre A y C y E y G; y G, que no produce ningún corte adicional y elimina el conflicto posterior entre A y C.

Todas las estrategias descritas tratan de construir un árbol BSP para la escena que cumpla las ya comentadas propiedades de equilibrio y minimización en el número de nuevos polígonos introducidos como resultado de cortes. Experimentalmente se ha comprobado (como se verá en la Sección 2.6) que en muchas ocasiones vale la pena sacrificar la condición de equilibrio si con ello se logra no aumentar significativamente la complejidad de la escena utilizada con la introducción de un número excesivo de nuevos elementos.



	neg	cort	pos	#pn + #np - #c
A		B, C	D, E, F, G	0 + 0 - 2 = -2
B	A	D, E	C, F, G	1 + 0 - 2 = -1
C	F	G	A, B, D, E	1 + 1 - 1 = 0
D		A	B, C, E, F, G	0 + 0 - 1 = -1
E		D, F, G	A, B, C	0 + 0 - 3 = -3
F	C, G	B	A, D, E	0 + 2 - 1 = 1
G	C		A, B, D, E, F	1 + 0 - 0 = 1

Figura 2.8: Ejemplo de selección de un plano para la partición binaria del espacio mediante el método de minimización de conflictos (profundidad 2).

2.4. Métodos basados en la coherencia de direcciones

Todos los optimizadores hasta ahora descritos se aprovechan de la localidad espacial de los polígonos en la escena para limitar el número de cálculos de intersección rayo–polígono. Una aproximación ligeramente diferente, aunque complementaria, consiste en explotar la coherencia en la localización y dirección de los rayos lanzados. Así, la idea básica consiste en que un optimizador basado en esta aproximación eliminaría de la lista de posibles obstáculos entre dos elementos aquellos objetos que no se encuentran en el espacio de direcciones entre ambos elementos, evitando tener que hacerlo rayo-a-rayo. Se trata por lo tanto de actuar en el dominio de los rayos en lugar de en el de la escena.

Estos métodos llevan a cabo una discretización del espacio de direcciones entre polígonos o conjuntos de polígonos en lugar de indexar el espacio de la escena. Como resultado se tiene que uno de los principales inconvenientes de estas técnicas es el alto consumo de memoria que producen.

Un precursor de esta clase de métodos es el algoritmo de clasificación de rayos

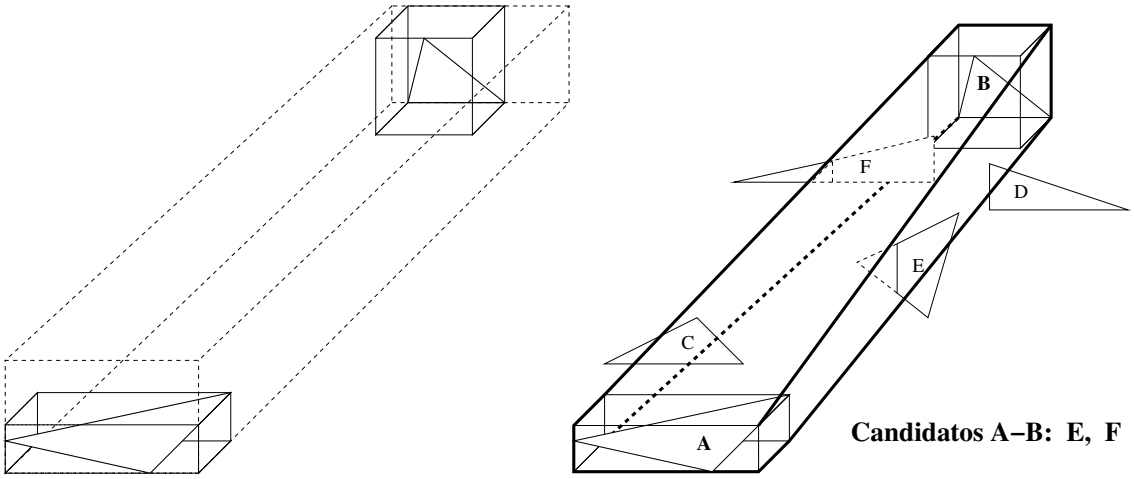
(*ray classification*) de Arvo y Kirk [10], que subdivide mediante volúmenes 5D (hipercubos, al considerar el dominio de los rayos como un espacio pentadimensional: las tres dimensiones espaciales del origen del rayo más las dos dimensiones angulares de su dirección) el espacio de todos los rayos posibles entre dos polígonos y asocia listas de candidatos a obstaculizar cada uno de estos conjuntos de rayos. La construcción de los volúmenes 5D es altamente costosa tanto en tiempo de ejecución como en memoria requerida y consumida, y el procedimiento en general presenta una mayor complejidad y es menos robusto y predecible que los optimizadores que trabajan en el dominio de la escena, por lo que han sido escasamente implementados hasta ahora [128, 32]. Algunos métodos más recientes, como el de Sharpe *et al.* [123], retoman la idea de la clasificación de rayos en 5D, solucionando parte de los problemas que presentaba el algoritmo original, pero todavía sin llegar a alcanzar los resultados en cuanto a rendimiento de otros métodos.

El mismo concepto sería tomado posteriormente por Haines y Wallace [63] pero en un espacio tridimensional, consiguiendo mejoras importantes en el rendimiento frente al método de Arvo y Kirk. En su método Haines y Wallace construyen una estructura 3D denominada «corredor» (*shaft*) que hace las veces de volumen de contorno para el espacio de direcciones entre dos polígonos. Así, todo rayo lanzado entre dos polígonos está contenido en el corredor construido entre ellos, con lo que se puede utilizar dicho corredor para previamente descartar aquellos polígonos que nunca van a estorbar en la relación de visibilidad entre dos objetos. Es el proceso conocido como *shaft culling*, y con él se construyen listas de candidatos a obstaculizar la visibilidad entre dos elementos, como se describe a continuación.

2.4.1. Listas de candidatos a obstáculos

Una manera sencilla de explotar la localidad direccional de los rayos entre dos objetos es delimitar el volumen que contiene a todos los posibles rayos lanzados entre los dos objetos y crear una lista con todos los polígonos que atraviesan o se encuentran por completo dentro de ese volumen (ver Figura 2.9). Esta lista contiene a todos los posibles candidatos a obstaculizar alguno de los rayos lanzados entre ambos polígonos.

La estructura volumétrica 3D utilizada, un corredor (*shaft*), no es más que un versión algo más estilizada de una caja de contorno alineada con los ejes. Se trata de un conjunto de planos que permite ajustar el volumen de la caja de contorno «gruesa» entre dos polígonos al espacio de direcciones entre ellos. Se construye a



(a) (b)

Figura 2.9: Construcción de una lista de candidatos a obstáculos mediante un corredor (*shaft culling*): (a) Caja de contorno gruesa entre los dos polígonos (b) Corredor obtenido y proceso de descarte de polígonos (*shaft culling*).

partir de las cajas de contorno individuales de los dos polígonos, formando primero la caja de contorno gruesa y sustituyendo luego los planos alineados con los ejes (un grado de libertad) donde es posible por planos que unen directamente las aristas de las cajas de contorno individuales (dos grados de libertad). Los planos utilizados permiten una rápida construcción del corredor, aunque no se logra el ajuste perfecto al espacio de direcciones que se conseguiría con planos con los 3 grados de libertad directamente entre los polígonos, además de acelerar la detección de intersección corredor–polígono respecto a una caja de contorno orientada.

En la Figura 2.9 se muestra un ejemplo de la creación de un corredor y la posterior construcción de la lista de candidatos para dos polígonos. Primero se forma la caja de contorno grueso y las cajas individuales de los dos polígonos (Figura 2.9a), usando planos alineados con los ejes. Esta caja gruesa será además útil para un rápido test de intersección preliminar de los polígonos. La caja gruesa es refinada usando planos que unen las aristas apropiadas en las cajas de contorno individuales, obteniendo el corredor (Figura 2.9b). La búsqueda de candidatos se lleva a cabo comparando los demás polígonos primero con la caja gruesa, para un descarte rápido de los polígonos más alejados, y después directamente con el corredor para un ajuste más fino con los polígonos más cercanos. En nuestro ejemplo (Figura 2.9b), la caja gruesa permite que el polígono C sea inmediatamente descartado como candidato a obstaculizar la interacción A-B. El corredor permite luego descartar también el polígono D, mientras

que E y F se confirman como candidatos a obstaculizar los rayos lanzados entre A y B.

2.5. Método basado en la coherencia angular de los rayos (*Angular coherence of rays*, ACR)

A pesar del uso de las diferentes técnicas de aceleración descritas, la determinación de la visibilidad entre objetos continúa todavía consumiendo una parte significativa del tiempo de ejecución en el cálculo de la iluminación global de una escena, por lo que sigue siendo prioritaria su optimización. Con ese objetivo presentamos en esta sección un nuevo método de aceleración basado en la aplicación de la coherencia direccional de los rayos, que puede usarse como complemento a los otros métodos de aceleración ya descritos. Nuestro método [105] consigue un gran ahorro en el tiempo final del cálculo de la iluminación global de una escena.

Como se comentaba en la introducción del capítulo, la determinación de la relación de visibilidad existente entre dos polígonos de una escena, (A, B), se calcula habitualmente como la proporción de los rayos lanzados entre ambos polígonos que no son interceptados por ningún otro polígono de la escena, lo que se traduce en la resolución de lo que vamos a denominar como *Ecuación de visibilidad*:

$$V^{AB} = \frac{1}{n}\sum_{i=1}^{n} V_i^{AB}, \qquad V_i^{AB} = \begin{cases} 0 & \text{si } \vec{r_i} \text{ es interceptado} \\ 1 & \text{en caso contrario} \end{cases} \tag{2.1}$$

donde $\vec{r_i}$ es un rayo que une los polígonos A y B (ver Figura 2.10), y n el número de rayos lanzado para comprobar la visibilidad entre dos polígonos, con un valor típico que suele rondar los $10 - 16$ rayos. El resultado final supone un gran número total de rayos disparados y, por lo tanto, de cálculos de intersección para una escena.

Como se ha comentado en la Sección 2.4, las técnicas direccionales tratan de explotar la coherencia en el dominio de los rayos. Así, nuestro nuevo método trata de reutilizar los rayos que ya han sido lanzados previamente en lugar de trazar nuevos rayos que podrían considerarse equivalentes a estos, combinando para ello la idea de crear una partición en el espacio de direcciones del posible conjunto de rayos lanzados desde un mismo punto origen con la formación de un conjunto de clases de equivalencia. Se trata de evitar la repetición en el cálculo del test de intersección

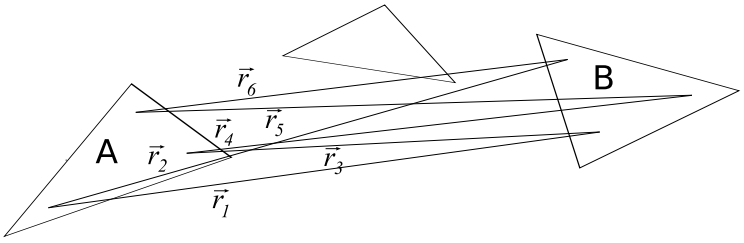


Figura 2.10: Lanzamiento de rayos necesario para resolver la ecuación de visibilidad entre dos polígonos.

rayo–escena para rayos que podríamos considerar equivalentes.

Cada rayo, $\vec{r}$, puede representarse con la siguiente expresión:

$$\vec{r}(\mathbf{p}, \vec{v}) = \mathbf{p} + t\vec{v},$$

donde $\mathbf{p}$ es el punto origen del rayo, $\vec{v}$ es el vector de dirección del mismo y t es el parámetro que indica la distancia a lo largo del rayo.

Los rayos son lanzados entre los distintos objetos usando un conjunto fijo de puntos de muestreo en cada polígono. Esto hace que muchas veces diferentes rayos lanzados desde un mismo punto de un polígono al resto de la escena puedan presentar entre sí sólo ligeras variaciones en cuanto a su dirección. Muchos de estos rayos con origen común siguen, por lo tanto, un recorrido prácticamente idéntico entre sí, interceptando el mismo subconjunto de objetos en su camino. En la Figura 2.11 se representa en 2D una serie de rayos lanzados desde uno de los puntos muestreados en un determinado polígono. Como puede observarse, los rayos $\vec{r_1}$ y $\vec{r_2}$ siguen un camino muy similar a pesar de ser trazados hacia puntos de destino en polígonos distintos. Lo mismo pasa con los rayos $\vec{r_3}$ y $\vec{r_4}$. De hecho, ambos rayos, que tratan de alcanzar el polígono D en la figura, son interceptados en este caso por el polígono de destino (el polígono C) del rayo con cuyo camino coinciden ($\vec{r_1}$ y $\vec{r_3}$, respectivamente). Decimos, entonces, que dos rayos

$$\vec{r_1}(\mathbf{p}, \vec{v_1}) = \mathbf{p} + t\vec{v_1}$$

$$\vec{r_2}(\mathbf{p}, \vec{v_2}) = \mathbf{p} + t\vec{v_2}$$

se consideran equivalentes si sus vectores de dirección correspondientes, $\vec{v_1}$ y $\vec{v_2}$, se intersecan en primer lugar con el mismo obstáculo. En este caso consideramos que ambos rayos caen dentro del mismo ángulo sólido (*solid angle*), para una determinada discretización del espacio de direcciones (2π *steradians*).

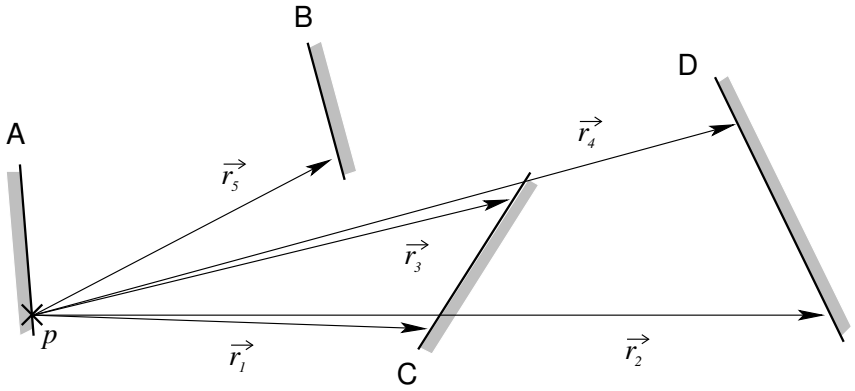


Figura 2.11: Distintos rayos lanzados desde un mismo punto p.

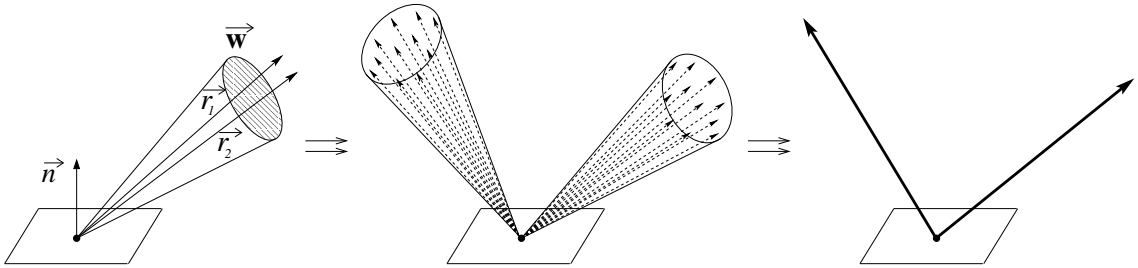


Figura 2.12: Clases de equivalencia de rayos.

La coherencia angular que se deriva de estos hechos puede ser usada para agrupar a aquellos rayos con caminos similares en una misma clase de equivalencia (Figura 2.12). De esta forma, es suficiente con realizar el test de intersección rayo–escena una única vez para cada clase de equivalencia en el cálculo de la visibilidad de un objeto con los diferentes objetos de la escena, escogiendo para ello un representante de la clase de equivalencia.

El hemisferio positivo de direcciones para cada punto utilizado como origen para el lanzamiento rayos se subdivide en un conjunto de ángulos sólidos, de manera lo suficientemente fina para que el error producido en la determinación de visibilidad se mantenga en el umbral deseado. El criterio empleado para obtener el ángulo sólido de referencia utilizado para crear la partición del espacio de direcciones para un punto concreto de un determinado polígono consiste en la creación de un polígono virtual artificial asociado al polígono y tomar el ángulo sólido que forma ese polígono virtual respecto al punto origen de la partición deseada. El polígono virtual asociado a un elemento consiste en un polígono cuya área sería la media de todas las áreas proyectadas de los polígonos que interactúan con el elemento de referencia (A_v) y

posicionado a una distancia que sería la media de sus distancias (D_v). El ángulo sólido,

$$W_r = \frac{A_v}{{D_v}^2},$$

formado por este polígono virtual promedio es la medida de referencia utilizada como base para el cálculo del tamaño (en *steradians*) de las celdas angulares de la rejilla regular que se construye en el espacio de direcciones para un punto origen de rayos. Así, se dividen los 2π *steradians* del hemisferio de direcciones en una rejilla angular regular, obteniendo un conjunto de ángulos sólidos de tamaño W_r, lo que en la práctica equivale a una partición regular del hemisferio en cada una de las dos coordenadas polares (ver Figura 2.13). La utilización de un ángulo sólido medio calculado únicamente a partir de los polígonos con los que realmente se interacciona proporciona una partición del hemisferio de direcciones personalizada para los puntos de cada polígono. Esta partición constituye el conjunto de clases de equivalencia que se mantienen para los puntos de ese polígono. En realidad, y como se describe más adelante, en lugar de utilizar directamente el ángulo sólido de referencia W_r para la construcción de las clases de equivalencia se usa un múltiplo del mismo, lo que permite ajustar el nivel de precisión/optimización deseado de la subdivisión angular.

El conjunto de clases de equivalencia creado puede ser explotado de dos maneras diferentes durante el proceso de lanzamiento de rayos. Una opción consiste en un funcionamiento puramente de caché de rayos, guardando para cada clase de equivalencia el último polígono interceptado por uno de sus miembros, que no tiene que ser necesariamente el obstáculo más cercano. Cuando un nuevo rayo es lanzado se comprueba primero si hay algún obstáculo guardado para su clase de equivalencia, y en caso afirmativo si este se interseca con el nuevo rayo lanzado. Si el obstáculo en la caché de rayos no se interpone a la trayectoria del nuevo rayo se sigue el procedimiento habitual en el disparo de rayos, utilizando el optimizador predeterminado que se estuviera empleando para acelerar el proceso. Utilizando el método ACR de este modo se puede acelerar ligeramente al cálculo de intersección rayo–escena sin pérdida alguna de precisión en el mismo.

El modo de empleo alternativo del método es, sin embargo, el que proporciona una mayor aceleración en la determinación de visibilidad, a costa de una pérdida más o menos importante de precisión. En este caso se escoge un único rayo como representante para cada clase de equivalencia, habitualmente el primer rayo lanzado para esa clase. En cada clase se va a guardar únicamente el polígono que constituye el obstáculo más cercano a la trayectoria del representante de la clase. Ese polígono

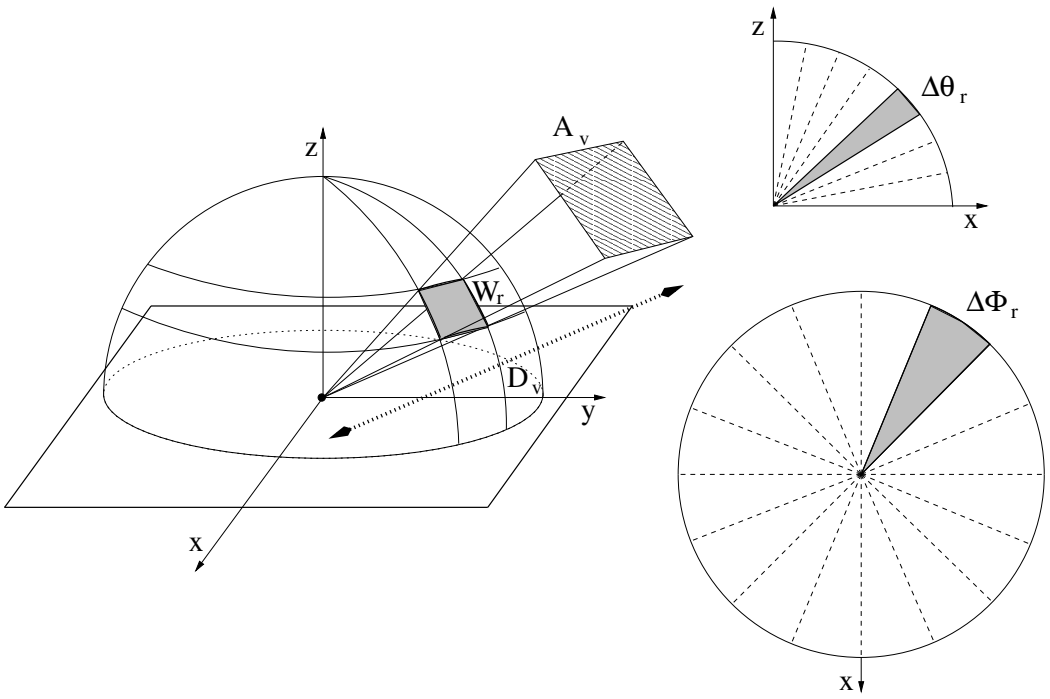


Figura 2.13: Discretización del hemisferio de direcciones en clases de equivalencia regulares.

se va a considerar, por lo tanto, como el obstáculo para cualquier rayo perteneciente a esa clase de equivalencia. Dado que los obstáculos almacenados para las clases son siempre los polígonos más cercanos, lo habitual será utilizar el método ACR en combinación con alguno de los métodos de indexación espacial que permiten obtener de forma rápida esta información, como un árbol PA-BSP, por ejemplo. Con esta aproximación se consigue incrementar significativamente el rendimiento del proceso del lanzamiento de rayos.

En ambas aproximaciones, la comprobación de si un rayo equivalente ha sido lanzado ya previamente se establece como paso previo al procedimiento normal en el cálculo de la intersección rayo–escena.

Una vez calculado el ángulo sólido de referencia (W_r) para un polígono, se pueden aplicar diferentes particiones en el espacio de direcciones de los puntos muestreados de ese polígono. La más simple sería la partición uniforme comentada hasta ahora, con celdas regulares con un tamaño determinado múltiplo de W_r. Esta partición presenta sin embargo el problema de que, usando simplemente W_r como tamaño de celda se obtiene una partición demasiado gruesa del espacio de direcciones, lo que

significa una simplificación demasiado agresiva del número de rayos a ser lanzado y por lo tanto un porcentaje de error cometido en el cálculo bastante grande si estamos empleando la aproximación de un único rayo lanzado por clase de equivalencia (ver Sección 2.6). Discretizaciones más finas (W_r/n, con $n = 2, 3 \ldots$) permiten rebajar sustancialmente este error conservando parte de la aceleración ganada, pero a costa de disparar los requisitos de memoria necesarios. La solución ideal pasa por la utilización de una estructura multinivel, como vemos a continuación.

2.5.1. Método ACR adaptativo

La necesidad de rebajar el error cometido con la simplificación introducida por el método ACR lleva a la utilización de particiones muy finas en el dominio de los rayos, lo que significa unas necesidades de almacenamiento demasiado elevados, sobre todo teniendo en cuenta que el método ACR, como el resto de métodos basados en la coherencia direccional, ya es de por sí bastante costoso en términos de memoria. La respuesta al problema consiste en utilizar organizaciones alternativas a la malla regular, como organizaciones adaptativas no uniformes, aplicando una jerarquía multinivel con varios niveles de clases y subclases de equivalencia, organizada utilizando alguna de las estructuras jerárquicas descritas en las secciones anteriores, de forma análoga a la partición del cubo de direcciones que realizan Arvo y Kirk [11]. De esta forma, refinando de manera adaptativa (podríamos decir que bajo demanda o necesidad) la rejilla angular se reducen los requisitos de memoria conservando las ventajas de utilizar rejillas muy finas.

En nuestro caso hemos empleado una configuración adaptativa de dos niveles, con una rejilla de tamaño fijo en ambos. El número de niveles a utilizar, y la configuración de los mismos podría variarse en función de los requerimientos de tiempo y precisión obtenida, así como en función de los recursos de memoria disponibles.

La operación básica para una configuración en dos niveles es la siguiente: cuando un rayo es lanzado desde un punto, la clase y la subclase de equivalencia a las que pertenece son calculadas. Si es el primer rayo lanzado para la clase de primer nivel, entonces es directamente escogido como representante de la clase de equivalencia (y análogamente es escogido como representante para la subclase de segundo nivel), y la clase no es refinada de momento (esto es, no es necesario reservar memoria para mantener el segundo nivel de la jerarquía para esta clase de momento). Cuando otro rayo perteneciente a la misma clase es lanzado, el segundo nivel de la jerarquía de clases se crea sólo si es necesario diferenciar varias subclases de equivalencia dentro

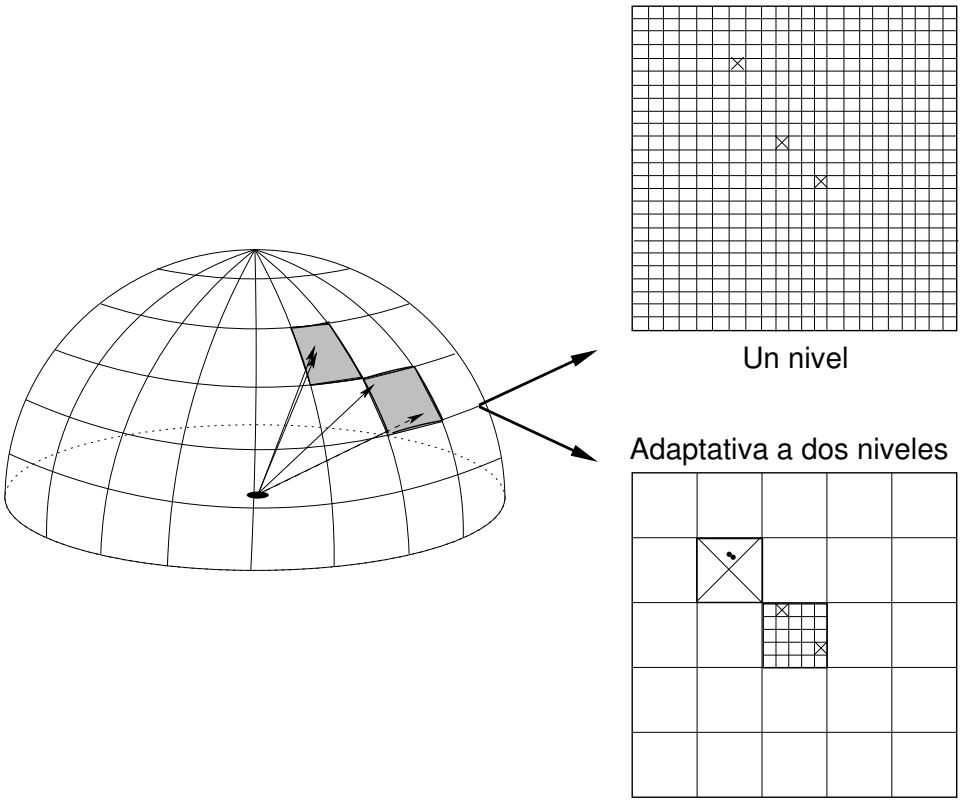


Figura 2.14: Una partición adaptativa de dos niveles del espacio de direcciones frente a una partición de un único nivel.

de la clase. En otro caso, la información guardada para el representante de la clase de equivalencia del primer nivel es utilizada (ver Figura 2.14).

Las ventajas en cuanto a ahorro de memoria con la utilización de una jerarquía en dos niveles son claras, e implican un coste computacional añadido muy pequeño. Así, inicialmente se utiliza únicamente el nivel más grueso de la partición, siendo una clase de equivalencia de este nivel refinada sólo cuando varios rayos lanzados para esa clase pertenecen a subclases del segundo nivel diferentes. Por ejemplo, en la Figura 2.14 se trazan cuatro rayos, dos de ellos para una misma clase de equivalencia, pero que corresponden a distintas subclases, lo que hace necesario aplicar, para esta clase, un segundo nivel de refinado. Los otros dos rayos pertenecen a una misma clase de equivalencia y, dentro de esta clase, también a la misma subclase, lo que hace innecesario, en este caso, un mayor refinado. Como se puede ver, la discretización de un único nivel equivalente, con la que se conseguiría el mismo resultado, sería mucho más costosa en términos de almacenamiento, pues necesita almacenar por completo

una rejilla con el tamaño de celda equivalente al más fino de la configuración en dos niveles.

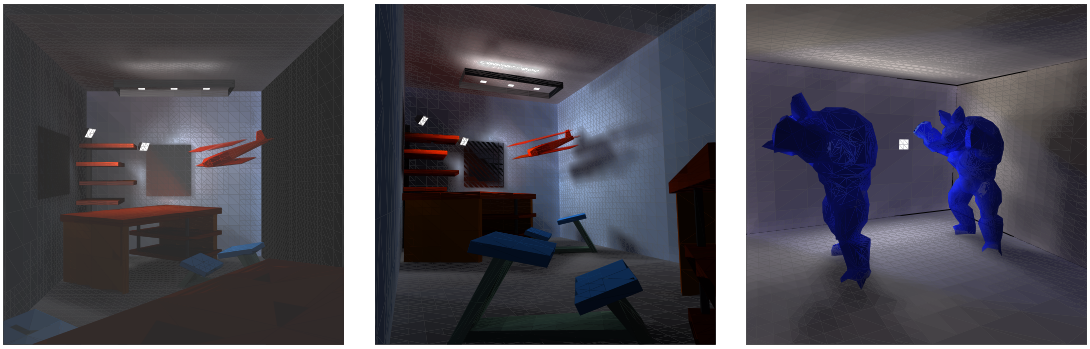
2.6. Resultados experimentales

En esta sección vamos a mostrar un análisis de los resultados obtenidos con la aplicación de los diversos algoritmos descritos en las secciones 2.2 – 2.4. También estudiaremos el rendimiento logrado, con sus ventajas y desventajas, con la aplicación del método ACR propuesto en la Sección 2.5.

Para probar las distintas implementaciones realizadas se ha utilizado un código para el cálculo de la radiosidad jerárquica [64] propio, en el que se han embebido las diferentes técnicas de aceleración. De esta forma, se han obtenido resultados experimentales para una serie de escenas de prueba simplemente cambiando el optimizador utilizado para la determinación de la visibilidad. Las pruebas se han realizado sobre un procesador Intel Xeon a 2.6 MHz sobre un sistema operativo GNU/Linux.

En la Figura 2.15 pueden verse las escenas utilizadas para las pruebas. La escena de la Figura 2.15a es la clásica habitación de Hanrahan, a la que llamaremos simplemente *Habitación*, que resulta en aproximadamente unos 11.000 triángulos finales tras el cálculo de su iluminación con el método de la radiosidad jerárquica. La escena que aparece en la Figura 2.15b se obtiene duplicando parte de la propia escena *Habitación* original en una especie de efecto espejo, al que denominamos *Habitación grande*. Se obtienen así unos 15.000 triángulos finales. Por último, la escena de la Figura 2.15c representa un caso no uniforme, con dos modelos poligonales complejos en una habitación vacía, con 13.000 polígonos finales y que etiquetamos con el nombre de *Armadillos*.

Las escenas utilizadas son simples y no demasiado exigentes en cuanto a cálculo y almacenamiento, lo que permite su ejecución sin problemas en una sola máquina sin consideraciones adicionales. Escenas mayores necesitan de la aplicación de técnicas de *clustering* [65] y/o de procesamiento paralelo [5], como veremos en los siguientes capítulos del trabajo.



(a) (b) (c)

Figura 2.15: Escenas de prueba: (a) *Habitación* (b) *Habitación grande* (c) *Armadillos*.

2.6.1. Estudio comparativo de las diferentes técnicas para la aceleración del trazado de rayos

Para comenzar se presenta una comparativa de la aplicación de los distintos métodos analizados en las secciones 2.2, 2.3 y 2.4. En las gráficas de la Figura 2.16, aparecen los tiempos finales para el cálculo de la radiosidad jerárquica para cada una de las escenas de prueba utilizando los distintos optimizadores para la aceleración del cálculo de la visibilidad. Concretamente, se muestran los resultados de utilizar un árbol PA-BSP (Sección 2.3.1), una jerarquía de volúmenes de contorno con un TF-OCT (Sección 2.2.1) y listas de candidatos a obstáculos construidas mediante un corredor (Sección 2.4.1).

Respecto a la construcción del árbol PA-BSP, las distintas heurísticas descritas en la Sección 2.3.1 han sido probadas. Los mejores resultados fueron los obtenidos con la estrategia *Least crossed with more crossed tie breaking* (con profundidad de análisis 1), por lo que son los tiempos que se han incluido en la gráfica de la Figura 2.16.

Los resultados muestran que el árbol BSP obtiene los mejores resultados para las tres escenas. Su rápida y sencilla construcción, junto con la gran reducción que consigue en la lista de polígonos a comprobar para el test de intersección rayo–escena son la clave del buen resultado conseguido. Además, como era de esperar, la estructura jerárquica de volúmenes de contorno por sí misma consigue los resultados más pobres en las tres escenas. Con estos resultados se puede concluir que la construcción de un árbol BSP adecuado para una escena, entendiendo por adecuado que no introduce un excesivo número de polígonos extra en la escena como resultado de su construcción y que está relativamente equilibrado, es una de las mejores alternativas

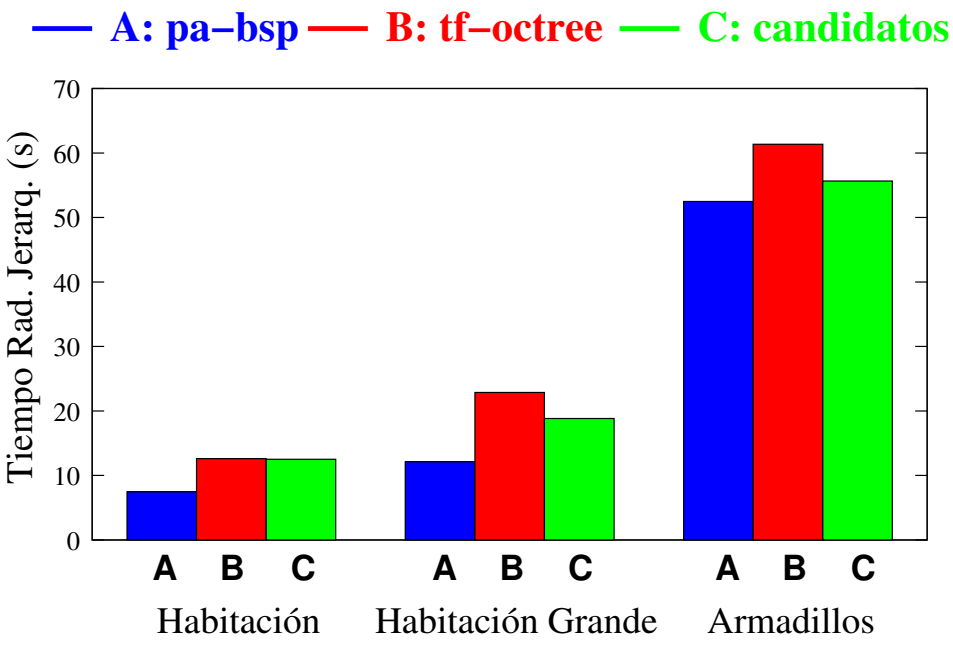


Figura 2.16: Tiempos obtenidos para el cálculo de la radiosidad jerárquica utilizando, para acelerar la determinación de visibilidad, un PA-BSP, un TF-OCTREE y listas de candidatos a obstáculos, respectivamente.

para la aceleración del proceso de lanzamiento de rayos, tanto en rendimiento como en sencillez de implementación y explotación.

El impacto que tanto la heurística empleada para la construcción del árbol BSP como la descripción de los polígonos de la escena de entrada puedan tener en el árbol PA-BSP resultante para la escena se analiza en la Figura 2.17. En este caso se utilizan dos versiones de la escena *Habitación*, etiquetadas en la Figura 2.17 como 1 y 2, respectivamente. La primera de ellas con una disposición adecuada en cuanto al orden en que los polígonos son enumerados en la escena, favoreciendo la construcción de un árbol BSP óptimo sin la necesidad de ninguna heurística adicional, simplemente siguiendo el orden de los polígonos en la escena, con los muros, paredes, techo y polígonos periféricos al comienzo de la descripción de esta. La segunda versión utilizada, por su parte, dispone los polígonos de la escena en un orden totalmente aleatorio en la descripción de la misma. Con esta prueba pretendemos medir la sensibilidad del método utilizado en la construcción del árbol frente a la organización inicial de los polígonos en la escena de entrada. Además del árbol PA-BSP utilizado en la Figura 2.16, un árbol PA-BSP simple (PA-BSP$_{simp}$) construido siguiendo el orden de los polígonos de entrada es comparado. Como se puede observar, el ár-

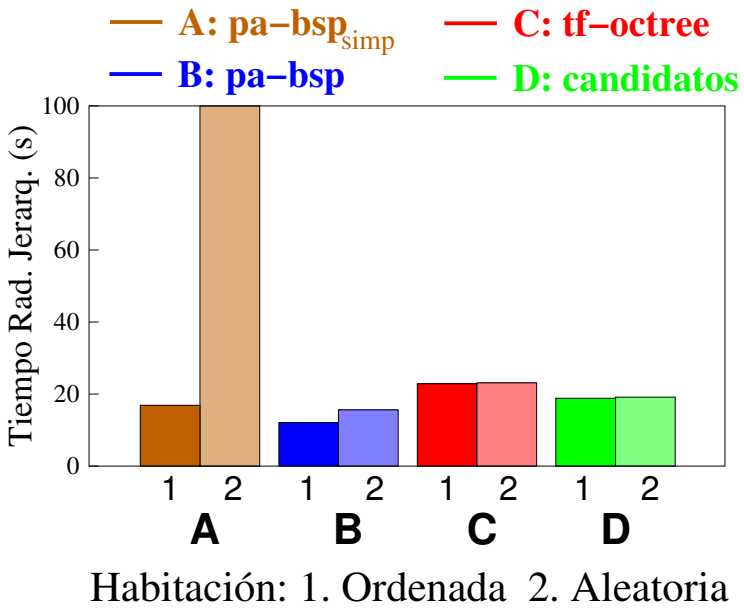


Figura 2.17: Impacto de la organización de los polígonos de la escena en los distintos métodos de aceleración.

bol PA-BSP con nivel de profundidad de análisis 0 tiene un buen comportamiento cuando la escena presenta una organización adecuada para ello, pero es completamente inútil en cualquier otro caso, debido a la gran cantidad de polígonos extra que introduce en la escena como resultado del enorme número de cortes producidos en su creación arbitraria. Una estrategia más sofisticada, como la *Least crossed with more crossed tie breaking* en este caso, añade a la construcción del árbol la robustez necesaria para su aplicabilidad en cualquier tipo de escenas. Este tipo de estrategias parecen por lo tanto de obligatoria utilización al menos que se puedan garantizar unas ciertas condiciones en la escena a procesar. Por supuesto, otros método de aceleración como el TF-OCT o las listas de candidatos tiene un comportamiento completamente robustos y fiable ante estas contingencias.

2.6.2. Resultados experimentales del método ACR

Para mejorar los resultados obtenidos en la Figura 2.16 hemos aplicado el método ACR descrito en la Sección 2.5. Concretamente se ha utilizado en combinación con el árbol PA-BSP, que además de mostrarse por sí solo como un gran optimizador para el lanzamiento de rayos, proporciona de manera sencilla la intersección más próxima para un rayo, cualidad necesaria para trabajar con el método ACR, como

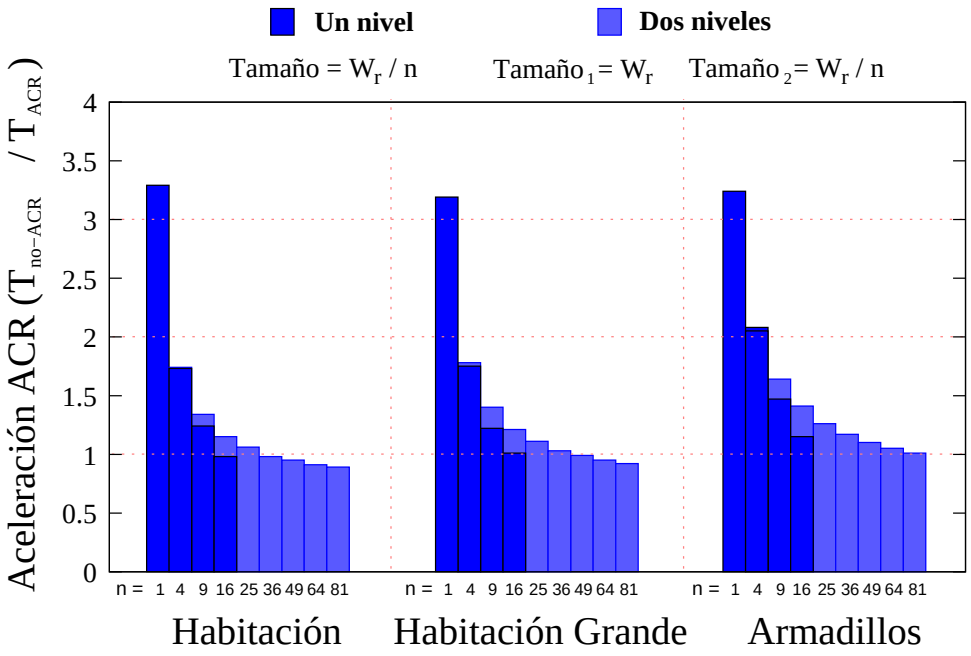


Figura 2.18: Rendimiento obtenido con la aplicación del método ACR, con diversas configuraciones, sobre un árbol PA-BSP.

se ha comentado anteriormente. Los resultados obtenidos, en términos de aceleración (*speed-up*), se muestran en la Figura 2.18, con

$$aceleración = \frac{tiempo\ noACR}{tiempo\ ACR} ,$$

donde el numerador se corresponde con el tiempo de procesamiento necesario para el cálculo de la radiosidad en la escena utilizando un árbol PA-BSP, y el denominador con el tiempo necesario con distintas configuraciones de ACR sobre el mismo árbol BSP. Se han probado varias configuraciones de rejilla angular, de más gruesas a más finas. Así, los primeros resultados corresponden a una configuración de un nivel, con celdas de tamaño W_r/n, con $n = 1, 2, 3 \ldots$ Para la configuración adaptativa de dos niveles (Sección 2.5.1) se han utilizado celdas de tamaño W_r para el nivel más grueso y de tamaño W_r/n, con $n = 2, 3, 4 \ldots$, para el nivel más fino, de manera que ambas configuraciones sean fácilmente comparables.

La gráfica de la Figura 2.19 representa el error introducido por el método ACR respecto a la utilización únicamente del PA-BSP original. Como medida de calidad de la escena resultante hemos utilizado la relación señal-a-ruido cuadrático media

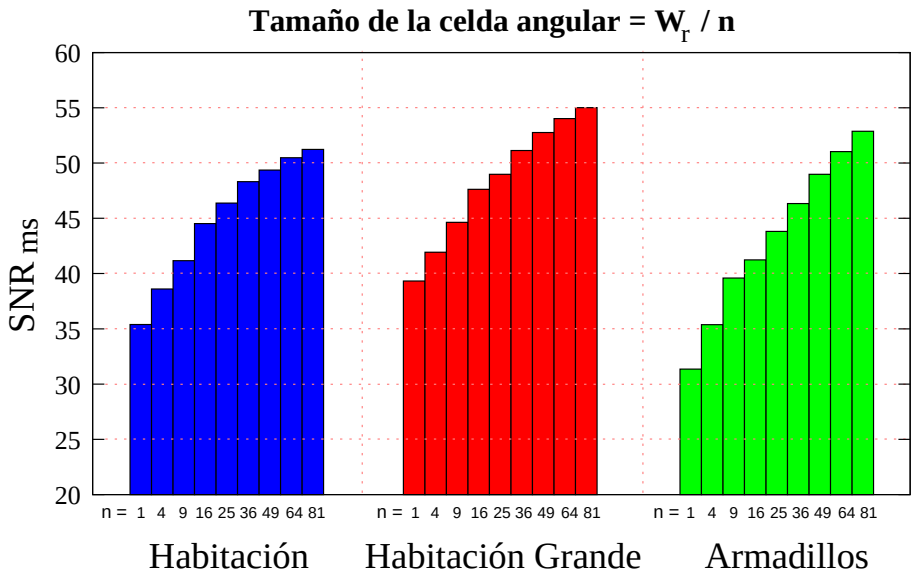


Figura 2.19: Medición del error cometido: relación señal a ruido (SNR_{ms}).

(*mean-square signal-to-noise ratio*, SNR_{ms}) [101], con la que comparamos la escena obtenida con las distintas configuraciones ACR y la obtenida directamente utilizando el PA-BSP:

$$SNR_{ms} = 10 \log_{10} \left(\frac{\sum_u \sum_v [Img_t(u,v)]^2}{\sum_u \sum_v [Img_t(u,v) - Img_r(u,v)]^2} \right),$$

siendo $Img_t(u,v)$ la imagen cuya calidad se desea medir (la escena obtenida mediante la aplicación del método ACR en nuestro caso); y $Img_r(u,v)$ la imagen de referencia utilizada. Para esta comparación, los valores de color utilizados corresponden a los vértices más finos comunes a las jerarquías producto del refinado de ambas escenas.

Como se puede observar, el tiempo necesario para el cálculo completo de la radiosidad de la escena se ve drásticamente reducido con la utilización del método ACR no adaptativo de un único nivel, aunque con bastante pérdida de precisión en la iluminación, como vemos en la Figura 2.19. Los altos requisitos de memoria necesarios para configuraciones más finas de la rejilla angular se solucionan con la utilización del método adaptativo de dos niveles, que permite aumentar el valor de n (ver Figura 2.18). Naturalmente esto se traduce en una disminución del error cometido, como la gráfica de la Figura 2.19 corrobora.

El método con la configuración de un nivel trabaja bastante bien con escenas uniformes como la habitación y la habitación grande. Así, por ejemplo, para $n = 4$ (un valor de $W_r/4$ para el tamaño de la celda) el rendimiento obtenido con la aplicación del método ACR no adaptativo de un nivel para las tres escenas de prueba sobrepasa 1,5 veces el obtenido simplemente con el la aplicación del árbol PA-BSP, manteniendo el nivel de la SNR_{ms} próximo a 40 (ver Figuras 2.18 y 2.19). Sin embargo, con escenas no uniformes como la de los armadillos, con muchos polígonos muy concentrados en una localización específica de la escena, el resultado se resiente y el error introducido por la simplificación que la rejilla ACR introduce se hace algo más palpable. Si en el caso de las dos habitaciones el valor de SNR_{ms} alcanza valores cercanos o superiores a 40 a partir de $n = 4$, y superiores a 35 en cualquier caso, para la escena menos uniforme de los armadillos no se logran resultados similares con rejillas más gruesas de $W_r/4$. Una partición más fina del espacio de direcciones se hace necesaria para este tipo de escenas si se pretende mantener la precisión en niveles altos, elevando así hasta niveles insostenibles las necesidades de memoria si se utiliza la configuración de un único nivel. Como se demuestra de los resultados de las gráficas de las Figuras 2.18 y 2.19, la configuración adaptativa de dos niveles es capaz de manejar ese tipo de situaciones, logrando un rendimiento alrededor de 1,5 veces el resultado de la no aplicación de ACR para $n = 9$, con un valor de SNR de casi 40 para la escena *Armadillos*.

Para mostrar cómo el proceso de lanzamiento de rayos se ve simplificado, la Figura 2.20 representa la proporción de rayos que alcanzan su polígono de destino sin encontrar obstáculo alguno durante la determinación de visibilidad para cada configuración del método ACR respecto al caso de su no utilización. Naturalmente, las configuraciones más gruesas están relativamente lejos de las cifras obtenidas sin la optimización ACR, y la distancia se va reduciendo de forma directamente proporcional a la reducción de la simplificación introducida (es decir, al emplear configuraciones más finas de rejilla). Observando la gráfica se ve que la pendiente es menos marcada en los casos de la habitación y la habitación grande, probablemente debido a la gran uniformidad de ambas escenas, con menos polígonos y más regularmente distribuidos que la escena de los armadillos, lo que produce que los rayos disparados desde un polígono tiendan a distribuirse más, cayendo en un mayor número de clases de equivalencia diferentes, lo que produce que un menor número de rayos sea simplificado.

En la gráfica de la Figura 2.21 se muestra información respecto al número de clases de equivalencia utilizadas en cada caso. De nuevo los datos varían en propor-

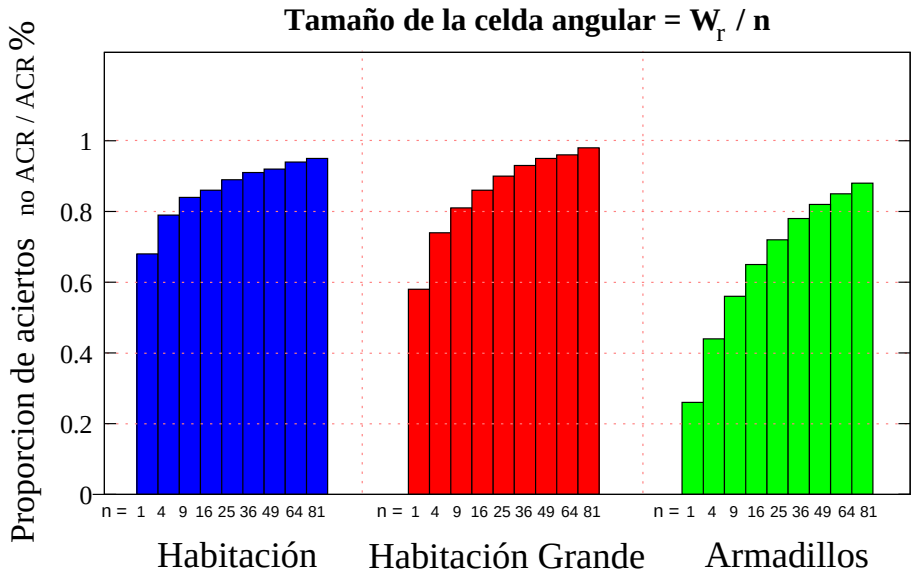


Figura 2.20: Proporción de rayos alcanzando su destino en cada configuración ACR respecto al caso sin ACR.

ción al nivel de simplificación introducido por el método ACR. La primera columna de cada par representa la proporción del número de clases de equivalencia creadas respecto al número total de rayos lanzados, mientras que la segunda columna muestra las veces que el rayo representante de una clase de equivalencia es utilizado. Así, configuraciones más gruesas presentan un menor número de clases de equivalencia que son más intensivamente reutilizadas (con el consiguiente ahorro en el lanzamiento de nuevos rayos), mientras que configuraciones más finas producen exactamente la situación opuesta, un mayor número de clases de equivalencia cuya información se utiliza en menor medida. Por lo tanto, un mayor número de clases de equivalencia se traduce en una menor simplificación introducida en la determinación de la visibilidad, al implicar una partición más fina del espacio de direcciones, y con ello en una mayor fidelidad del resultado, aunque con una menor mejora en el rendimiento, como se observa en la Figura 2.16.

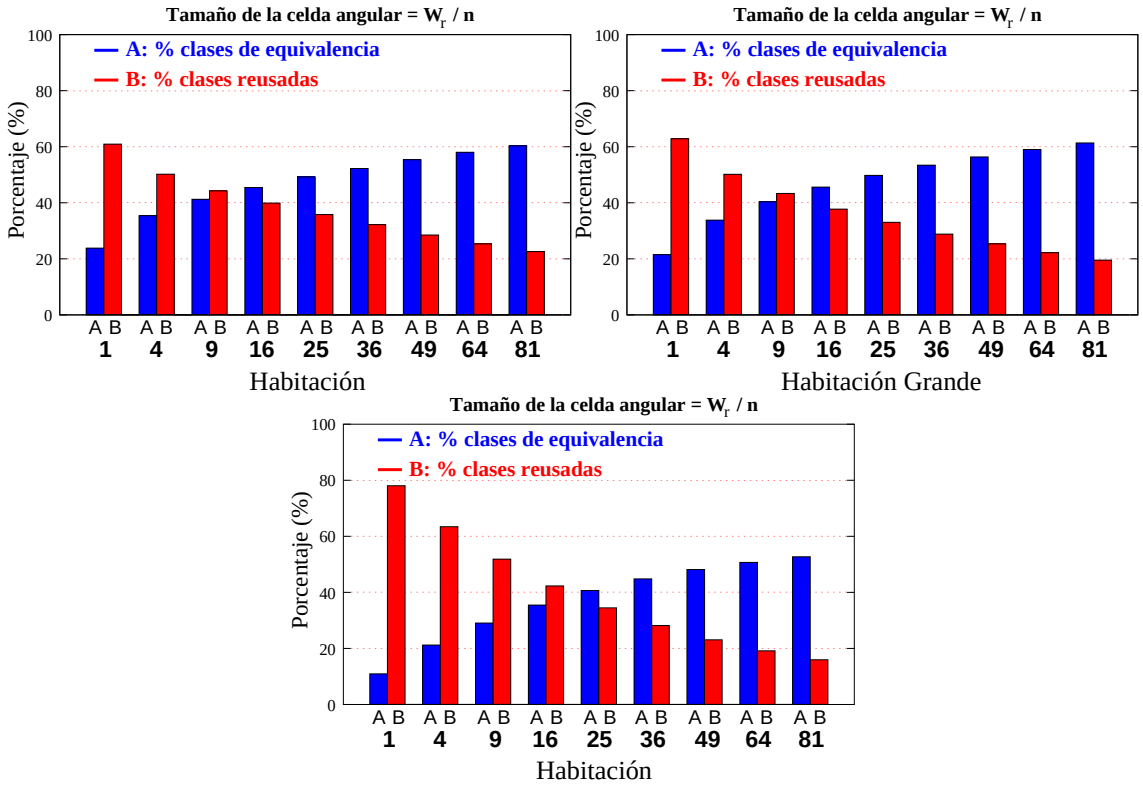


Figura 2.21: Proporción de clases de equivalencia existentes para los rayos lanzados y porcentaje de veces que el representante de una clase de equivalencia es reutilizado.

Capítulo 3

Subdivisión de superficies

Las cada vez más altas expectativas de realismo a alcanzar en muchos de los campos en los que se aplica la informática gráfica generan la necesidad de trabajar con curvas y superficies lo más suaves y detalladas posibles. Dado el uso generalizado de polígonos planos como principal entidad geométrica a la hora de modelar un entorno, la obtención de superficies detalladas que aproximen una superficie curva suave, mediante la subdivisión recursiva de superficies planas gruesas, ha pasado en los últimos años a ser un recurso habitual en informática gráfica, modelado geométrico y visualización científica.

Por otro lado, la utilización de modelos poligonales altamente detallados complica la aplicación de muchas de las técnicas destinadas a lograr una apariencia final realista de los modelos procesados, como es el caso de los modelos de iluminación global. Se trata de métodos con una complejidad lineal, en el mejor de los casos, respecto al número de polígonos de entrada, con lo que su utilización se vuelve prácticamente inviable, en términos de tiempo de cálculo y de consumo de memoria, con la aparición de modelos excesivamente complejos en las escenas de entrada. Además, gran parte del aumento en el coste introducido con el uso de modelos poligonales muy complejos puede no verse en absoluto reflejado en el resultado final alcanzado. Así, por ejemplo, en el caso del método de radiosidad no todas las interacciones entre los diversos objetos de una escena tienen la misma importancia de cara a la obtención de un aspecto final realista en la iluminación de la escena. Lo solución más habitual en estos casos consiste en la introducción de ciertas simplificaciones en los modelos a la hora de efectuar determinados cálculos. Esta simplificación no afectan en exceso al resultado final obtenido, aunque sí al coste computacional necesario

para lograrlo. Con esa idea surge lo que se conocen como técnicas de *clustering*, que consisten en la creación de grupos o conjuntos de polígonos, normalmente con una organización jerárquica, que van a poder ser tratados de forma conjunta en cuanto a su comportamiento con el resto de la escena.

En el caso de la radiosidad, con la aplicación de técnicas de *clustering* se consigue una extensión del método de radiosidad jerárquica, de forma que, en la jerarquía de elementos que se utiliza para la distribución de la energía de la escena, los polígonos iniciales no sean la entidades más gruesas entre las que poder considerar interacciones [131, 65]. La aproximación más frecuente al *clustering* en radiosidad trata de agrupar los objetos de la escena de forma que la modelización de la transferencia de luz desde y hacia los grupos de objetos se encuentre por debajo de un determinado umbral de error [127, 26, 95]. Un enfoque diferente, sin embargo, lo proporcionan una serie de métodos basados en técnicas de multiresolución y simplificación de superficies: *face clustering* [144, 47]. En este caso se trata de calcular la iluminación utilizando versiones simplificadas de los modelos complejos iniciales, proyectando luego el resultado final sobre el modelo sin simplificar.

En este trabajo describimos una aproximación propia a la utilización de *clustering* para acelerar el cálculo de la radiosidad en escenas complejas, basada en las ideas de simplificación y subdivisión de superficies (los detalles se muestran en el Capítulo 4). Así, durante el proceso de cálculo de la iluminación global de la escena, mediante el método de radiosidad jerárquica, se utilizarán modelos simplificados de los objetos complejos. A la hora de *renderizar* la escena, sin embargo, estos modelos son refinados hasta el nivel de calidad requerido mediante un algoritmo de subdivisión recursivo.

Como solución para aplicar el proceso de refinado de los modelos tras el cálculo de la iluminación, hemos desarrollado una versión paralela de dos de los algoritmos de subdivisión de superficies más populares: *Loop* [86] y *Modified Butterfly* [149]. El diseño y la implementación realizadas, así como los resultados obtenidos, son, sin embargo, generalizables a otros esquemas de subdivisión basados en la utilización de la información de vecindad de primer orden, incluidas estrategias de subdivisión adaptativa [17]. Nuestra propuesta paralela está basada en la partición de la malla original en diferentes grupos de triángulos, utilizando las técnicas de agrupamiento de triángulos presentadas en [17, 2] con el nombre de *Grouping*. La partición obtenida cubre la malla inicial de triángulos del modelo por completo, de manera eficiente, equilibrando el número de triángulos por grupo de la partición y con un reducido número de triángulos frontera entre grupos. Una vez creados, los grupos son cla-

sificados y distribuidos entre los distintos procesadores para su refinado. Nuestra aproximación utiliza un balanceo dinámico de la carga computacional entre los distintos procesadores, con el propósito de alcanzar el mayor equilibrio posible en la distribución de dicha carga y acortar los tiempos de espera. Así, tras un reparto inicial de los grupos de triángulos entre los procesadores, fase eminentemente estática y que no requiere comunicaciones, cada procesador se va a encargar de aplicar el esquema de subdivisión sobre los grupos de triángulos preasignados. Posteriormente, un esquema de balanceo dinámico y distribuido de la carga computacional entre los procesadores permite equilibrar en gran medida la carga de trabajo del sistema. Para ello, los grupos de triángulos todavía no procesados pueden pasar de un procesador con mucha carga de trabajo, al que inicialmente habían sido asignados, a otro menos cargado durante el proceso de refinado.

En el resto del capítulo describimos en detalle los esquemas de subdivisión paralela propuestos, dejando para el Capítulo 4 la explicación de su utilización dentro de un método para el cálculo de la radiosidad de una escena. Así, el resto de este capítulo se organiza de la siguiente forma: una primera parte donde se describen, respectivamente, los dos algoritmos de subdivisión utilizados en el trabajo, *Loop* (Sección 3.2) y *Modified Butterfly* (Sección 3.3), tras un pequeño repaso al estado del arte en subdivisión de superficies (Sección 3.1); y una segunda parte con la paralelización propuesta para ambos algoritmos, y aplicable, además, a otros esquemas de subdivisión (Sección 3.4), junto con los resultados con ella obtenidos (Sección 3.5).

3.1. Introducción

Un algoritmo de subdivisión recursivo obtiene una superficie suave como el límite de una secuencia de sucesivos refinados aplicados sobre una superficie gruesa más simple, que hace las veces de malla de control. La superficie suave hacia la que tiende el proceso iterativo se denomina superficie límite. Las reglas de subdivisión utilizadas dependen exclusivamente de la conectividad, a nivel topológico, de la malla gruesa de partida. Existe una amplia variedad de esquemas de subdivisión de superficies, de forma que han llegado a convertirse en herramienta habitual y necesaria dentro de la informática gráfica y del campo del modelado geométrico. En general, las distintas técnicas de subdivisión de superficies existentes se pueden clasificar en dos grupos: técnicas de subdivisión aproximada o técnicas de subdivisión interpolada.

Los esquemas de subdivisión aproximada obtienen una malla fina a partir de la malla gruesa o de control mediante la generalización de los métodos para la obtención y subdivisión de curvas *B-spline*, por lo que uno de sus comportamientos más característicos es que la malla resultante del refinado no conserva los vértices de la malla inicial. Así, las técnicas de Doo-Sabin [35] y Catmull-Clark [21] aproximan respectivamente superficies *B-spline* bicuadráticas y cúbicas a partir de una malla de control rectangular. Uno de los representantes más populares de esta clase de métodos es el algoritmo *Loop* [86, 34], que extiende las ideas de Catmull-Clark [21] para el trabajo con mallas de triángulos. Trabajos recientes [132, 142] han unificando los esquemas de subdivisión Loop y Catmull-Clark, permitiendo trabajar de forma conjunta sobre mallas híbridas con triángulos y rectángulos.

Los esquemas de subdivisión interpolados, por su parte, se caracterizan porque no mueven los puntos de control existentes, sino que simplemente insertan, a partir de la aplicación de una serie de reglas, un nuevo punto por cada dos puntos de la malla original. Por lo tanto, con este tipo de métodos los vértices de la malla gruesa inicial permanecen como vértices en la malla refinada. El método más conocido basado en interpolación es el algoritmo *Butterfly* [37]. Su principal inconveniente radica en la necesidad de partir de una malla inicial de topología regular para ser capaz de obtener una malla refinada lo suficientemente suave (al menos C^1) [1]. Considerando un vértice regular u ordinario aquel que tiene valencia [2] seis, que es el valor que resulta tras subdividir cada triángulo grueso en cuatro nuevos subtriángulos, se entiende por topología regular la ausencia total de vértices extraordinarios. La variante conocida como *Modified Butterfly* [149, 148] soluciona este problema, manteniendo la simplicidad del algoritmo original y permitiendo obtener resultados satisfactorios aun con la presencia de vértices extraordinarios en la malla de partida.

Los dos tipos de esquemas de subdivisión comentados parten de la misma idea básica, en la que cada arista de la malla es subdividida utilizando la información de las aristas vecinas. Sin embargo, mientras que los algoritmos de subdivisión aproximada como *Loop* tienen su punto fuerte en la alta calidad (en cuanto a suavidad) de las mallas refinadas obtenidas, para una velocidad de convergencia relativamente alta, los esquemas de subdivisión interpolada presentan una implementación extraordinariamente simple y permiten una gran cantidad de trabajo *in place*, es decir, en la propia malla a subdividir, presentando por lo tanto unos menores requisitos de memoria. Métodos como *Modified Butterfly* son además muy utilizados en proble-

[1] En matemáticas, a una función con k derivadas continuas se la denota como C^k *continua*.
[2] La valencia de un vértice es el número de aristas que lo contienen.

mas de multiresolución, problemas de *wavelets*, mapas de desplazamiento... debido al mayor control obtenido sobre la malla final resultante.

En los últimos años las diferentes *APIS* gráficas han incorporado herramientas para la subdivisión de superficies (las dos más conocidas, *DirectX* y OpenGL, con los nombres de *NPatches* y *PN Triangles*, respectivamente), funciones que las tarjetas gráficas han empezado a implementar en los últimos años. Así, por ejemplo, una de las primeras soluciones *hardware* para la subdivisión de superficies fue la tecnología *TRUFORM* [99] desarrollada por *ATI Technologies*, que implementa en la GPU una subdivisión de superficies de Bézier basada únicamente en la utilización de la información local de cada arista. Se pueden obtener, sin embargo, superficies finales más suaves utilizando además de la información local de cada arista a subdividir la información de las aristas y vértices vecinos [14, 17, 2]. Por otro lado, las posibilidades que ofrecen las nuevas GPUs programables también permiten una implementación efectiva de un esquema de subdivisión en el propio *hardware* gráfico [125], aunque todavía no hay demasiados desarrollos al respecto.

La cantidad de memoria y el tiempo de computación necesarios para obtener mallas realmente finas son los principales inconveniente de este tipo de esquemas de subdivisión recursivos. El número de vértices de la malla, $\bar{v}_i$, y el número de cálculos a realizar crecen de forma exponencial respecto al número de iteraciones de refinado aplicadas:

$$\bar{v}_i = \bar{v}_0 \times r^i, \tag{3.1}$$

partiendo de una malla con $\bar{v}_0$ elementos, y aplicando un factor de subdivisión no adaptativa de r, tras i iteraciones.

En nuestro trabajo hemos utilizado dos de los esquemas de subdivisión más populares, *Loop* y *Modified Butterfly*, en los cuales todos los cálculos realizados para obtener la posición de los vértices de la malla en cada iteración se basan únicamente en la utilización de la información de vértices con vecindad inmediata antes de la subdivisión, con lo que los puntos obtenidos son combinaciones lineales de estos vértices vecinos. En cualquier caso, nuestro trabajo admite, sin modificaciones excesivas, el uso de cualquier otro esquema de subdivisión no adaptativa de naturaleza similar.

A continuación se describe brevemente el funcionamiento de estos dos esquemas de subdivisión. En ambos casos se trata de una subdivisión no adaptativa de una

malla de triángulos, de manera que en cada iteración se obtiene un nuevo vértice para cada arista de la malla, con lo que, una vez reconectados todos los vértices, cuatro nuevos triángulos sustituyen a cada uno de los triángulos de la malla original. No es el objetivo de la presente memoria el desarrollar la formulación matemática por la que se alcanzan las expresiones utilizadas en los métodos, por lo que para una mayor información pueden consultarse [121, 142].

3.2. Loop

El esquema de subdivisión *Loop* [86] produce una secuencia iterativa de mallas de triángulos que converge en su límite hacia una superficie suave C^2 continua. En cada iteración se obtiene un nuevo punto de la malla para cada dos vértices unidos por una arista en la malla original mediante simple interpolación lineal, de manera que cuatro nuevos subtriángulos sustituyen al original. Tras la subdivisión, la posición de todos los vértices es recalculada en función de las vecindades, haciendo que la malla obtenida esté más cerca de la superficie curva suave, que se alcanzaría en el límite del proceso iterativo.

La subdivisión de una malla mediante el algoritmo *Loop* consiste en la aplicación de forma iterativa de un proceso recursivo de dos pasos. Un primer paso es la propia fase de subdivisión (*splitting step*), en la cual cada triángulo se subdivide en cuatro nuevos triángulos, partiendo cada una de sus aristas por la mitad. El segundo paso es la fase de aproximación o suavizado (*averaging step*), en la cual las posiciones de cada vértice en la nueva malla, tanto de los nuevos vértices creados en la fase previa como de los vértices existentes en la malla original, son recalculadas de acuerdo a una suma ponderada de las posiciones de los vértices vecinos (ver Figura 3.1).

El método de subdivisión distingue cuatro casos diferentes en la fase de aproximación, cada uno de ellos con una configuración diferente de los pesos de los vértices vecinos a utilizar para el cálculo de las nuevas posiciones.

1. Los vértices interiores que no han sido creados en la iteración actual del proceso de subdivisión, y que por lo tanto estaban presentes en la malla previa, utilizan un patrón de pesos con una configuración de $n + 1$ puntos como la de la Figura 3.1a, siendo n la valencia del vértice a aproximar.

2. Los vértices interiores nuevos usan el esquema de cuatro puntos de la Figura 3.1b para obtener su posición definitiva.

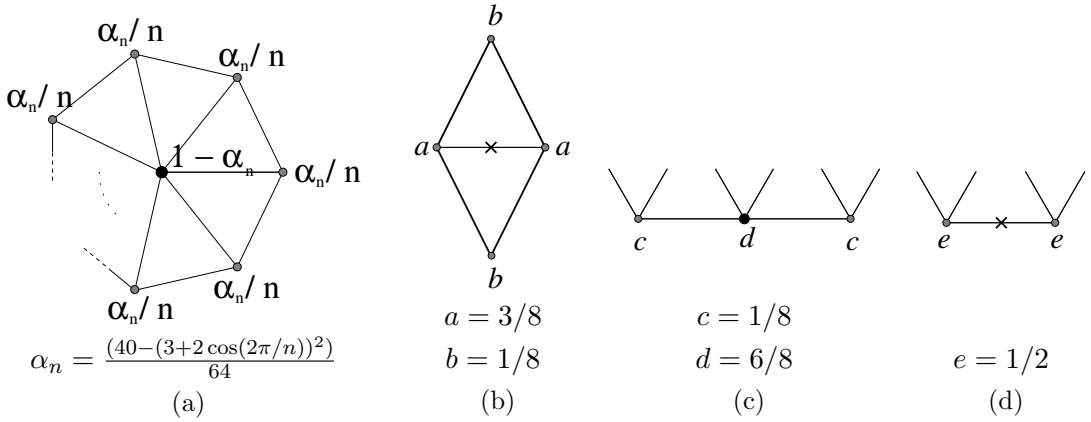


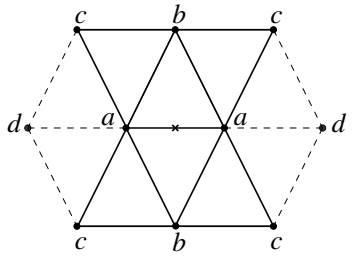
Figura 3.1: Patrones con los pesos de los vértices para la fase de aproximación en el esquema de subdivisión *Loop*: (a) Vértices interiores previos (b) Vértices interiores nuevos (c) Vértices exteriores previos (d) Vértices exteriores nuevos.

3. Para actualizar la posición de los vértices exteriores, que se encuentran en un borde de la malla, que ya se encontraban en la malla previa, se utiliza un esquema de tres puntos, incluyendo al propio vértice (Figura 3.1c), con lo que se evita una dependencia de los vértices interiores de la malla que llevaría a la aparición de huecos y desajustes entre mallas diferentes con frontera común.

4. Por último, los nuevos vértices exteriores, que se introducen con la subdivisión de una arista borde de la malla, solamente tienen en cuenta la posición de los vértices de la arista a la que subdividen, con los pesos que se muestran en la Figura 3.1d.

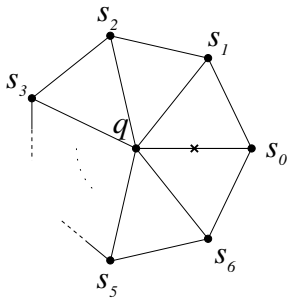
3.3. Modified Butterfly

El algoritmo *Modified Butterfly* [149] procede de manera similar a *Loop*, aplicando iterativamente y de forma sistemática una serie de reglas para la creación de nuevos vértices de subdivisión para cada arista de la malla. Sin embargo, tratándose de un esquema de subdivisión interpolada, tanto los vértices de la malla gruesa inicial como los nuevos vértices, que son sucesivamente obtenidos en cada iteración, mantienen de forma permanente la misma posición en las iteraciones siguientes. La superficie límite hacia la que tiende el algoritmo tiene continuidad C^1, por lo que el resultado es una malla menos suave que la obtenida con los esquemas de subdivisión aproximada como *Loop*. *Butterfly*, sin embargo, presenta ciertas características que lo hacen más



$$a = 1/2 - w$$
$$b = 1/8 + 2w$$
$$c = -1/16 - w$$
$$d = w$$

(a)

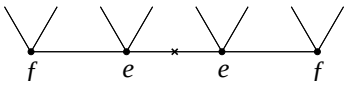


$$w_q = 3/4$$

$$K = 3 \Rightarrow \begin{cases} w_{s_0} = 5/12 \\ w_{s_1} = -1/12 \\ w_{s_2} = -1/12 \end{cases} \qquad K = 4 \Rightarrow \begin{cases} w_{s_0} = 3/8 \\ w_{s_1} = 0 \\ w_{s_2} = -1/8 \\ w_{s_3} = 0 \end{cases}$$

$$K \geq 5 \Rightarrow w_{s_j} = \frac{1/4+\cos(2\pi j/K)+1/2\cdot\cos(4\pi j/K)}{K}, \quad 0 \leq j < K$$

(b)



$$e = 9/16$$
$$f = -1/16$$

(c)

Figura 3.2: Patrones de pesos para el esquema de subdivisión *Modified Butterfly*: (a) Vértices regulares (c) Vértices bordes (b) Vértices K-irregulares.

adecuado en ciertos contextos, como el mantener un mayor control sobre la superficie resultante, que en los esquemas aproximados tiende a volverse cada vez más pequeña en tamaño.

Para la aplicación del algoritmo se consideran fundamentalmente tres tipos de vértices en la malla a refinar: vértices regulares, que son aquellos vértices interiores con valencia 6; vértices K-irregulares, que son vértices interiores con valencia K, siendo K distinto de 6; y vértices bordes, que no se encuentran rodeados por completo de triángulos.

En cada iteración, el algoritmo obtiene un nuevo vértice para cada arista de la malla, mediante la suma ponderada de las posiciones de los vértices vecinos a la arista, ponderando cada vértice con un peso predeterminado (ver Figura 3.2). Se distinguen cuatro casos distintos de subdivisión:

1. La arista a refinar conecta dos vértices regulares. El patrón con los pesos de los vértices vecinos a utilizar es el esquema de diez puntos que se muestra en la Figura 3.2a, extensión del esquema de ocho puntos del algoritmo *Butterfly* original. El nuevo vértice a obtener se calcula como la suma ponderada de las posiciones de los vértices vecinos usando esos pesos. El término w es un parámetro de tensión, que regula la separación que se le permite a la superficie límite hacia la que tiende el algoritmo respecto a la malla de control inicial. Así, por ejemplo, un valor de $-1/16$ produciría un esquema de interpolación lineal simple, con lo que no se alcanzaría una superficie suave en absoluto. Un valor de 0 hace que se utilice el patrón de ocho puntos del algoritmo *Butterfly* original.

2. La arista conecta un vértice K-irregular y un vértice regular. En este caso el algoritmo trata de acercar lo más posible los coeficientes utilizados al caso regular. El método opta, además, por tener en cuenta únicamente los vértices vecinos al vértice irregular, con los pesos indicados en la Figura 3.2b.

3. La arista conecta dos vértices K-irregulares. En este caso, el nuevo vértice se obtiene calculando la media de los resultados producidos por cada uno de los vértices de la arista. Se aplica, por lo tanto, la plantilla del caso anterior a cada uno de los dos vértices irregulares, escogiendo los pesos adecuados para sus valencias, y se toma la media de los valores obtenidos. Este caso tiene relativamente poca relevancia en el resultado final de la subdivisión, al ser sólo posible su aparición en la primera iteración del proceso.

4. Las aristas bordes son subdivididas usando un esquema 1-dimensional de cuatro puntos (ver Figura 3.2c). Con esta disposición, al utilizar únicamente vértices del borde de la malla, se consigue, como en el caso equivalente de *Loop*, evitar la aparición de incoherencias en la subdivisión de dos mallas separadas con un borde común entre ellas.

3.4. Subdivisión de superficies paralela

En esta sección describimos la aproximación seguida para conseguir una ejecución eficiente de un esquema de subdivisión recursivo en un sistema multiprocesador, y que ha sido presentada inicialmente en [102, 104, 103]. Los principales objetivos a cumplir en el diseño de la solución paralela desarrollada son el equilibrio de la carga

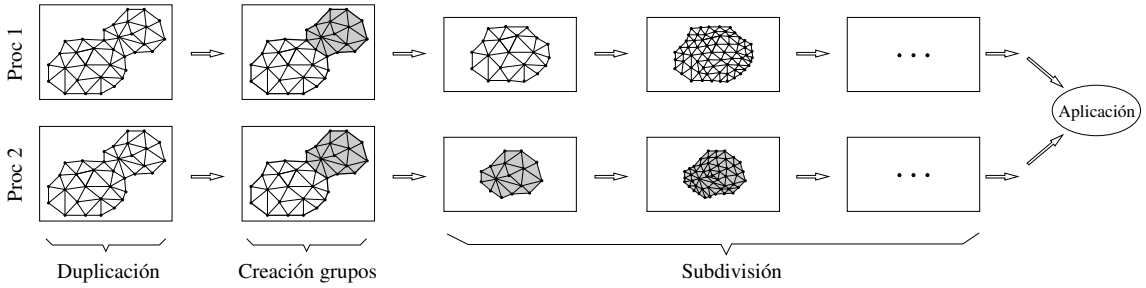


Figura 3.3: Método de subdivisión de superficies paralelo basado en agrupamiento.

computacional entre los procesadores y la minimización del número de comunicaciones necesarias. Además, se ha tratado de realizar una paralelización lo más general posible, razón por la cual se ha optado por un diseño basado en sistemas de memoria distribuida, utilizando un paradigma de paso de mensajes para las comunicaciones entre procesadores. Se trata de una paralelización de grano grueso, en la que cada procesador realiza todo el trabajo de subdivisión para un determinado conjunto de triángulos de la malla inicial. La paralelización se basa en la creación de una partición sobre la malla inicial de triángulos mediante un algoritmo de agrupamiento de triángulos, y es aplicable a cualquier esquema de subdivisión recursivo que utilice información de los vértices con un nivel de vecindad 1.

En la Figura 3.3 se muestra de forma esquemática el algoritmo de la aproximación paralela a la subdivisión de superficies desarrollada en este trabajo. La malla gruesa de partida se almacena inicialmente completa en la memoria de todos los procesadores (primera columna de la Figura 3.3). Este hecho no representa un inconveniente importante, al tratarse de una malla gruesa y por lo tanto con unos requisitos de almacenamiento no demasiado altos. A continuación, cada procesador aplica el algoritmo de agrupamiento de triángulos, con lo que la malla es partida (segunda columna de la Figura 3.3), quedándose cada procesador con el cálculo de la subdivisión para un conjunto diferente de triángulos (columnas 3, 4 y 5 de la Figura 3.3), sobre los que se aplica el pertinente esquema de subdivisión recursivo.

En las siguiente subsecciones se describe, por un lado, el algoritmo de agrupamiento empleado para crear la partición de la malla, pasándose luego a analizar el esquema de balanceo dinámico de la carga implementado.

3.4.1. Agrupamiento de triángulos

Para el procesamiento en paralelo de un conjunto de datos es necesaria la distribución de los mismos entre los distintos procesadores, aplicando algún tipo de partición sobre ellos. En nuestra solución se crean distintos grupos o conjuntos de triángulos dentro de la malla inicial mediante la técnica de *Grouping* [17, 2]. Este método trata de reducir el número de triángulos frontera de las mallas, generando grupos de triángulos equilibrados y cubriendo por completo la malla de entrada de manera eficiente. El reparto de estos grupos entre los distintos procesadores nos va a permitir afinar el equilibrio en la carga computacional entre ellos.

El método utilizado para la creación de grupos de triángulos se basa en la búsqueda de un vértice central para el grupo, a partir del cual se selecciona un determinado número de anillos o tiras de triángulos concéntricas. Las restricciones que se imponen para la creación de los grupos son, principalmente, el que los grupos creados sucesivamente sean contiguos, y que no existan agujeros, es decir, triángulos sin asignar, entre grupos.

Durante todo el proceso de construcción de los grupos de la partición se mantiene una lista de puntos iniciales, L^{start}. Esta lista, que inicialmente se encuentra vacía, almacena una ristra de vértices a partir de los cuales se comienza la búsqueda del vértice central para los siguientes grupos a construir. Adicionalmente, también se mantiene una lista de vértices exteriores o frontera del super-grupo formado por la unión de todos los grupos construidos en un determinado momento, B^{global}.

El algoritmo para la creación de los grupos de triángulos es el siguiente:

1. El algoritmo comienza tomando aleatoriamente un vértice de la malla como vértice central para la construcción del primer grupo (por ejemplo, el vértice v_1 en la Figura 3.4a). La lista L^{start} está vacía.

2. El grupo de triángulos alrededor del vértice central es seleccionado. En nuestro ejemplo de la Figura 3.4 los grupos están formados por tres tiras concéntricas de triángulos alrededor del vértice central, v_1.

3. Cada vez que un grupo es construido se actualiza la lista de puntos iniciales, L^{start}. Se añaden a la lista los vértices exteriores del grupo recién construido que previamente ya estaban en la lista de vértices exteriores globales, B^{global}, y de los que haya triángulos que los contengan que todavía no hayan sido asignados a grupo alguno.

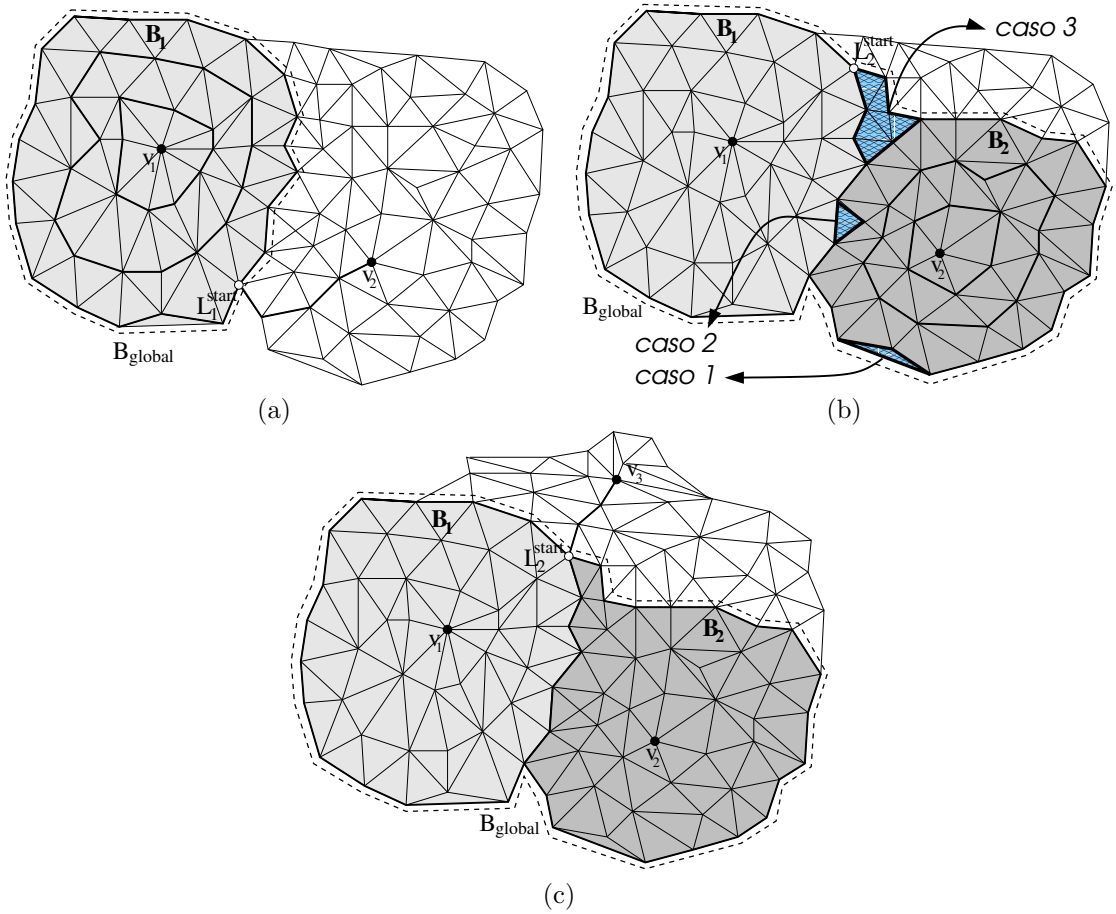


Figura 3.4: Creación de grupos de triángulos: (a) Primer grupo creado (b) Construcción del segundo grupo (c) Selección del vértice central para el siguiente grupo.

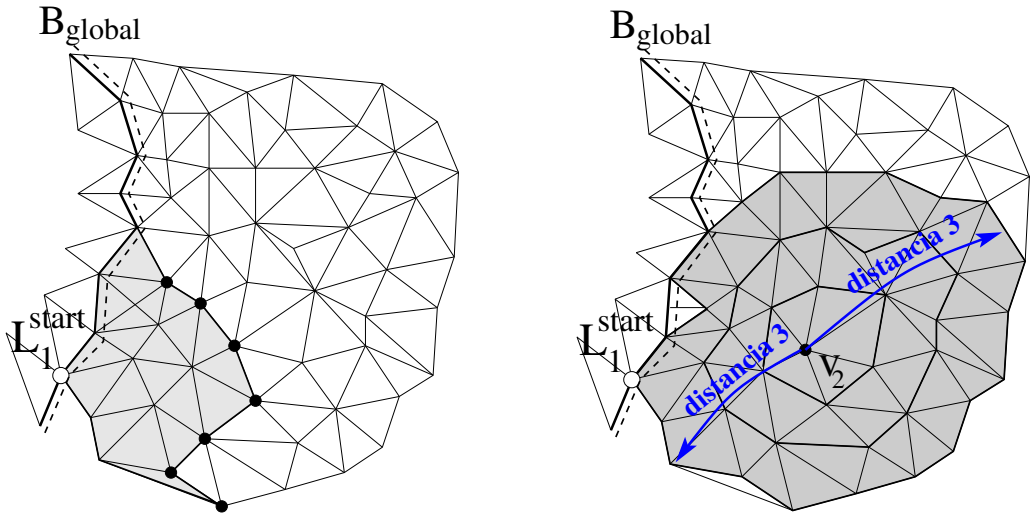


Figura 3.5: Selección del vértice central: (a) Selección de un conjunto de candidatos (b) Construcción del nuevo grupo.

En el ejemplo de la Figura 3.4b, el vértice L_2^{start} se añade a la lista L^{start}, pues es un vértice exterior del grupo B_2 que se encontraba ya previamente en la lista de vértices exteriores del super-grupo B^{global} (marcado en la Figura 3.4a con línea discontinua).

En el caso particular que surge tras la construcción del primer grupo de la malla, al encontrarse la lista B^{global} previamente vacía, simplemente se escoge un vértice arbitrario de B^{global} (el primero, por ejemplo) para la lista L^{start} (el punto L_1^{start} en la Figura 3.4a).

4. Una vez actualizada la lista L^{start} se hace lo propio con la lista B^{global}. Así, los vértices exteriores de cada nuevo grupo construido, B^{local}, son añadidos a la lista de vértices exteriores globales, B^{global}, excepto en el caso de vértices que pertenezcan a la intersección de ambas listas ($B^{local} \cap B^{global}$) y cuyos triángulos hayan sido todos ya previamente asignados a algún grupo. Estos vértices pasan ahora a ser vértices interiores respecto al super-grupo global, por lo que se eliminan de la lista B^{global}. En el ejemplo, vemos la lista B^{global} después de la construcción del primer grupo con línea discontinua en la Figura 3.4a y la lista B^{global} una vez añadido un segundo grupo en la Figura 3.4b.

5. La lista de puntos iniciales, L^{start}, se utiliza para obtener el vértice central de cada nuevo grupo a construir (v_2 en la Figura 3.4b y v_3 en la Figura 3.4c,

por ejemplo). La idea es buscar, partiendo de un vértice v de la lista L^{start}, un vértice que esté rodeado por un determinado número de anillos concéntricos de triángulos, de manera que no se solapen con ninguno de los grupos ya construidos. El procedimiento a seguir se representa en la Figura 3.5, considerando grupos de tres anillos de triángulos. Primero, se identifican los vértices candidatos que estén a distancia tres del punto L_1^{start} (Figura 3.5a). De este conjunto de vértices se escogen únicamente los vértices que estén a una distancia mínima de tres de cualquier vértice de la lista B^{global} y del borde de la malla, entre los que se selecciona aleatoriamente el vértice central para el nuevo grupo a construir. En el ejemplo, solo el punto v_2 (Figura 3.5b) cumple estas condiciones.

6. El algoritmo concluye cuando todos los vértices de la malla ya han sido agrupados, obteniéndose una partición que cubre por completo la malla de triángulos.

La ejecución del algoritmo, tal y como ha sido descrita, no garantiza la cobertura completa de la malla con el conjunto de grupos creados. Por ello, una vez construido cada grupo es necesario llevar a cabo una serie de ajustes [17, 2] para evitar dejar agujeros entre grupos. Las tres situaciones especiales a considerar, de las cuales se muestran ejemplos en la Figura 3.4b, son:

1. Triángulo sin asignar con todos sus vértices asignados. Un triángulo cuyos vértices han sido todos incluidos en un grupo debe también ser incluido en ese grupo (Figura 3.6a).

2. Agujeros entre grupos. Los triángulos que hayan quedado sin asignar entre el último grupo construido y el grupo en construcción se añaden al grupo en construcción. En la Figura 3.6b aparece en gris claro el último grupo construido y en gris oscuro el super-grupo global con todos los grupos formados hasta el momento. El grupo de triángulos sin asignar entre estos dos conjuntos tiene que ser identificado y asignado al nuevo grupo en construcción.

3. Identificación de huecos formados por triángulos no asignados con vértices asignados a diferentes grupos. Si los triángulos del hueco son asignados a un nuevo grupo, gran parte de la información de los triángulos vecinos para su subdivisión pertenecería a grupos diferentes, lo que dificultaría su uso en caso de resultar estos grupos posteriormente asignados a diferentes procesadores. Los triángulos de estos huecos son asignados al grupo recién construido una

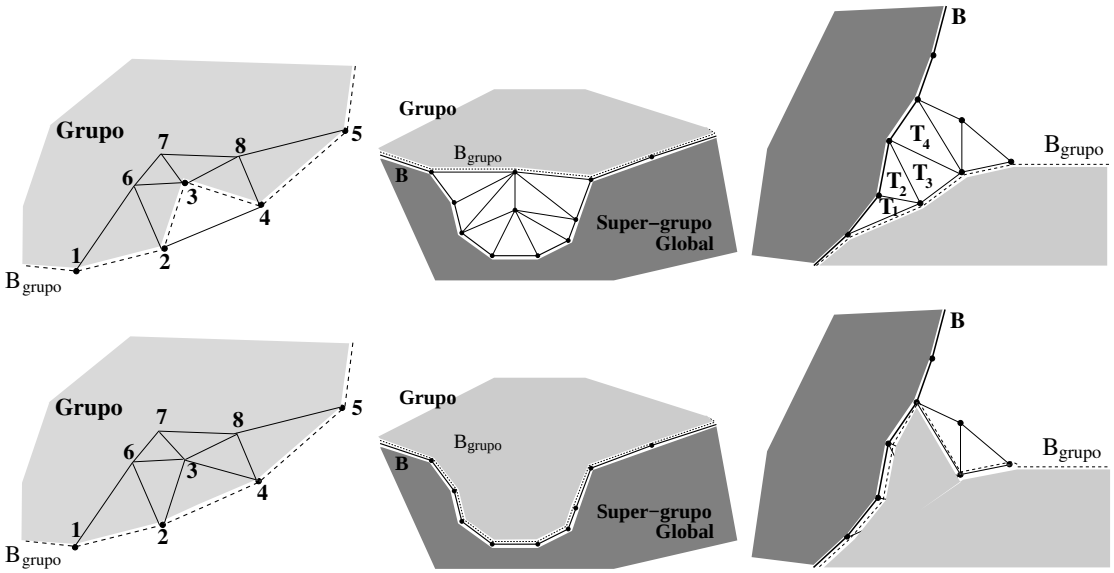


Figura 3.6: Casos especiales a corregir: (a) caso 1: triángulo no asignado con vértices asignados (b) caso 2: agujeros entre grupos (c) caso 3: huecos formados por triángulos con vértices asignados a grupos diferentes.

vez identificados. Un ejemplo de la aparición de estos huecos se muestra en la Figura 3.6c, donde la ristra de triángulos $\{T_1, T_2, T_3, T_4\}$ está delimitada por los grupos previamente calculados (frontera B) y el grupo actual (frontera B_{grupo}). La situación debe ser detectada para incluir estos triángulos en el grupo en construcción.

3.4.2. Algoritmos de subdivisión en un sistema distribuido

Una vez realizada en cada procesador la partición en grupos de la malla gruesa a procesar, los grupos obtenidos se reparten entre los procesadores para la ejecución paralela del esquema de subdivisión recursivo. Este reparto inicial, totalmente estático y que no requiere comunicación alguna entre los procesadores, se realiza mediante una distribución cíclica de los grupos de triángulos entre los procesadores, previa ordenación decreciente de los mismos de acuerdo al número de triángulos que contienen. Tratamos así de equilibrar de la forma más sencilla y rápida posible la carga computacional inicial asignada a cada procesador.

Tras el reparto inicial, cada procesador se encargará, en principio, únicamente

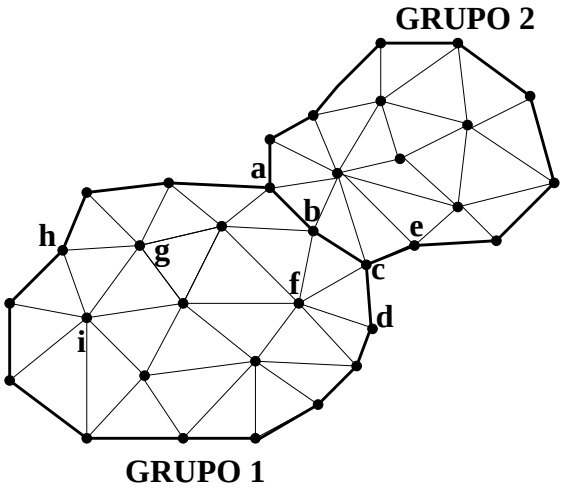


Figura 3.7: Dos grupos adyacentes.

del procesado de los triángulos asignados estáticamente. Así, para p procesadores y una malla cuya partición resulta en g grupos, el número de grupos asignados al procesador i sería de $\lfloor g/p \rfloor + (i < g \mod p)$ grupos de triángulos.

Con el reparto de los grupos entre los procesadores, y la subdivisión de los triángulos de estos en paralelo, las aristas comunes que se encuentran en la frontera entre dos grupos asignados a distintos procesadores son procesadas dos veces, una en cada uno de ellos. Para garantizar la coherencia de la malla, los vértices obtenidos en ambos casos deben ser idénticos. La alternativa utilizada para lograrlo consiste en la aplicación de un patrón de pesos sencillo, que realice simplemente la interpolación lineal de los dos vértices de la arista (ver Figura 3.1d). En la Figura 3.7 se muestra un ejemplo con dos grupos adyacentes asignados a diferentes procesadores. Teniendo en cuenta la información disponible en cada procesador para el proceso de subdivisión para estos dos grupos, podemos clasificar las aristas en tres clases:

- Aristas interiores. Los dos vértices son internos al grupo (por ejemplo, la arista $g - i$ de la Figura 3.7). En este caso toda la información relativa a los vecinos está disponible, por lo que se pueden usar los patrones normales para vértices regulares (Figuras 3.1a y 3.2a) e irregulares (Figuras 3.1a y 3.2b).

- Aristas fronteras. Los dos vértices se encuentran en la frontera entre dos grupos asignados a distintos procesadores. No se dispone, por lo tanto, de la información vecinal suficiente para la aplicación del patrón de pesos normal. Además, el uso del patrón para aristas bordes (Figuras 3.1c y 3.2c) puede pro-

ducir divergencias en la posición de los nuevos puntos introducidos, generando incoherencias en la malla. Así, por ejemplo, en la Figura 3.7, el empleo del patrón de cuatro puntos para la subdivisión de la arista entre los vértices b y c implicaría en uno de los grupos a los vértices a, b, c y d, mientras que en el otro serían a, b, c y e los vértices usados. La solución pasa, para este tipo de casos, por el uso de una simple interpolación lineal de los vértices de la arista.

- Aristas exteriores. Uno de los vértices es interno, mientras que el otro se encuentra en la frontera entre dos grupos asignados a distintos procesadores, con lo que solo se dispone de la información necesaria para la aplicación del patrón de pesos normal para la subdivisión de la arista relativa a uno de sus dos vértices. En este caso, en el que se encuentran aristas como la $h-g$ o la $b-f$ en el ejemplo de la Figura 3.7, también se aplica una simple interpolación lineal entre los vértices de la arista a subdividir.

Esta distinción nos permite trabajar con aristas de las que no se dispone de toda la información vecinal. Para mantener la suavidad en el resultado obtenido en estos casos se pueden utilizar las normales de los vértices de la arista [140]. Cabe destacar que, cuando dos grupos vecinos son asignados a un mismo procesador, toda la información de conectividad entre los grupos se encuentra disponible, con lo que se puede aplicar el esquema de subdivisión original sin aplicar la simplificación comentada. Así, en el ejemplo de la Figura 3.7, si los grupos uno y dos se asignan a un mismo procesador la arista $b-f$ pasa a ser una arista interior más en ese procesador, reduciendo el error cometido.

3.4.3. Distribución dinámica

Para equilibrar la carga de trabajo asignada a cada procesador se ha diseñado un esquema de distribución dinámica de los grupos entre los procesadores, con el que se consigue balancear, durante el proceso de subdivisión, la carga computacional entre los distintos procesadores. Este esquema permite a los procesadores más ocupados descargar parte del trabajo que estáticamente les había sido preasignado sobre procesadores más ociosos, que ya hayan terminado la subdivisión sobre sus grupos. De esta manera se reduce el desequilibrio, y con ello los tiempos de espera, entre procesadores; todo ello, además, de una forma completamente distribuida entre los propios procesadores inmiscuidos en el proceso de subdivisión, sin la necesidad de que uno de ellos actúe como árbitro o maestro de este proceso.

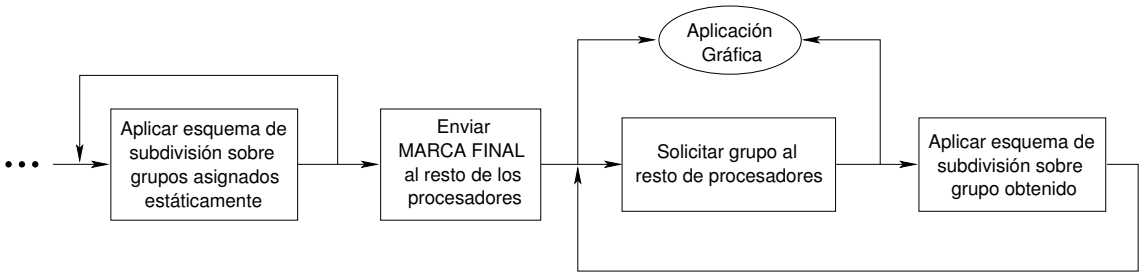


Figura 3.8: Esquema distribuido para el reparto dinámico de la carga.

El esquema seguido para la redistribución dinámica de la carga se muestra en la Figura 3.8:

1. Los grupos creados con la partición efectuada sobre la malla inicial son distribuidos entre los procesadores. Cuando un procesador termina con el trabajo preasignado, envía un mensaje al resto de procesadores anunciando este hecho. Este procesador pasará ahora a encargarse de grupos de triángulos pertenecientes a procesadores que todavía no hayan acabado el proceso de subdivisión asignado estáticamente.

2. Para hacerse con grupos asignados a otros procesadores que todavía no han sido procesados, se envía una petición al siguiente procesador (según el número de identificación del procesador) que tenga todavía grupos sin procesar. En todo momento cada procesador lleva contabilidad de los procesadores que van terminando con los grupos asignados, con lo que las peticiones se van redirigiendo a los procesadores adecuados.

3. Si el procesador que recibe el mensaje de petición dispone todavía de algún grupo sin procesar en el momento de contestar, le traspasa la propiedad de uno de sus grupos al procesador que realizó la petición (ACK). Por el contrario, si el procesador ya no dispone de grupos sin procesar en el momento de recibir la petición, bien porque los haya terminado de procesar o bien porque hayan sido asignados como respuesta a las peticiones ya atendidas a otros procesadores, la petición es rechazada (NACK).

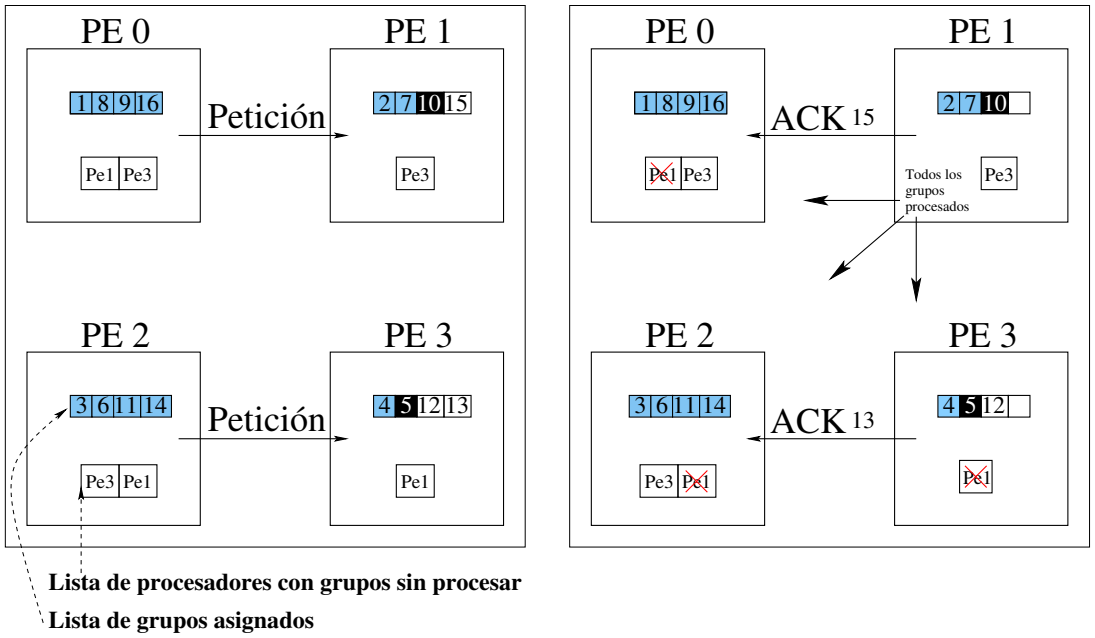
4. Cada nuevo grupo obtenido es procesado, volviendo a continuación al paso 2. Cuando no quedan grupos sin procesar se da por finalizado el proceso de subdivisión y se comunican los grupos subdivididos a la aplicación que los requiera.

Tabla 3.1: Resumen de mensajes intercambiados.

	Envíos		Recepciones	
Qué?	Cuándo?	A quién?	Cuándo?	De quién?
MARCA FINAL	tras procesar grupos preasignados	a todos	en cualquier momento	de proc con grupos ya procesados
PETICIÓN	ocioso y grupos disponibles en otro proc	a algún proc con grupos sin procesar	en cualquier momento	de un proc ocioso
CESIÓN GRUPO	petición recibida y grupos locales sin procesar	al remitente de la petición	tras haber enviado una PETICIÓN	de proc al que se envió PETICIÓN

Con el esquema de planificación empleado se trata de minimizar en lo posible el número de comunicaciones entre los procesadores, evitando además múltiples asignaciones de un grupo a dos o más procesadores. En la Tabla 3.1 se resumen los mensajes de comunicación posibles entre los procesadores. Como se muestra, los tres tipos principales de mensajes que se intercambian los procesadores son: la señalización de haber terminado el procesado estático asignado, la petición de cesión de grupos si procesar y la cesión o traspaso de un grupo a otro procesador. No es necesario el uso de mensajes explícitos de rechazo de las peticiones (NACK), ya que la propia marca de final del procesado local sirve como mensaje de denegación de cualquier petición dirigida a un procesador. El esquema de comunicaciones utilizado es, además, *no bloqueante*, tanto en emisión como en recepción, con lo que se solapan en lo posible comunicaciones y cálculos.

En la Figura 3.9 se presenta un ejemplo del protocolo descrito. Se muestran cuatro procesadores, a los que le son asignados cuatro grupos de triángulos de la malla a cada uno. En la lista de grupos asignados a cada procesador se muestran en gris los grupos ya procesados, y en negro el grupo sobre el que se está trabajando en ese momento. Cada procesador mantiene una lista con los procesadores con grupos disponibles todavía sin procesar. Así, cuando los procesadores 0 y 2 terminan con sus grupos, buscan nuevos grupos a procesar de los procesadores siguientes: el procesador 0 envía una petición al procesador 1, mientras que el procesador 2 hace lo propio con el procesador 3. A ambos procesadores les son asignados, respectivamente, los grupos 15 y 13. Cuando el procesador 1 termina con sus grupos, envía un mensaje al resto, pasando también al modo de búsqueda de grupos ajenos sin procesar.



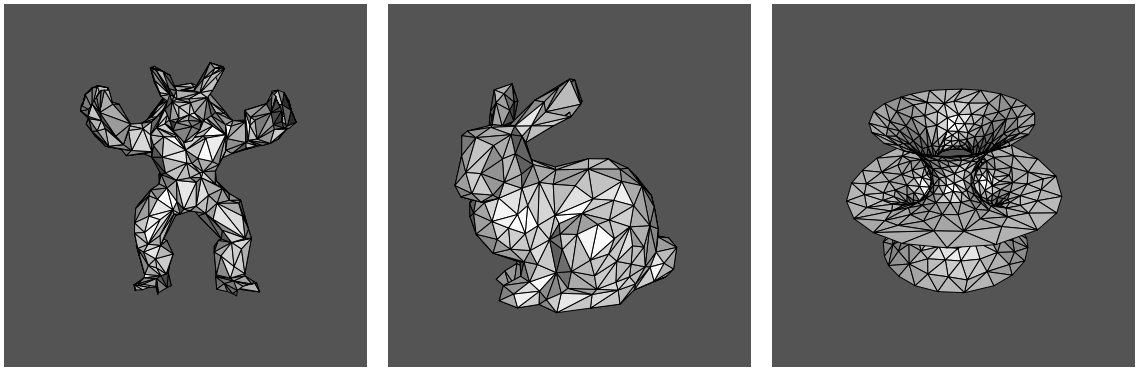
(a) Petición (b) Concesión

Figura 3.9: Ejemplo del esquema de distribución dinámica de la carga.

3.5. Resultados experimentales

Con el algoritmo paralelo para la subdivisión de superficies descrito en los apartados anteriores se han desarrollado implementaciones paralelas de los esquemas de subdivisión *Loop* y *Modified Butterfly*. Para ello se ha empleado el lenguaje de programación C y la librería de paso de mensajes MPI, para la comunicación entre los procesadores, resultando una implementación totalmente multiplataforma y portable a diferentes arquitecturas. Las plataformas sobre las que se han realizado las pruebas descritas en esta sección son: un *cluster* SunFire 6800 con procesadores UltraSPARC-III a 750 MHz; una SGI Origin 2000, sistema de memoria compartida-distribuida con procesadores R10000 a 250 MHz y un Fujitsu Ap3000 con procesadores UltraSparc-2 a 300 MHz.

En la Figura 3.10 se muestran los modelos utilizados para probar el método implementado: *Armadillo*, con 799 triángulos, *Conejo*, con 499, y *Hypersheet*, con 917. Las mallas resultantes tras la aplicación de tres iteraciones de los métodos *Modified Butterfly* y *Loop* aparecen en la Figura 3.11. El número de triángulos obtenido tras la subdivisión asciende a, respectivamente, 51.136, 31.936 y 58.688 triángulos. Debido a la gran semejanza en los resultados obtenidos en nuestras pruebas para los dos



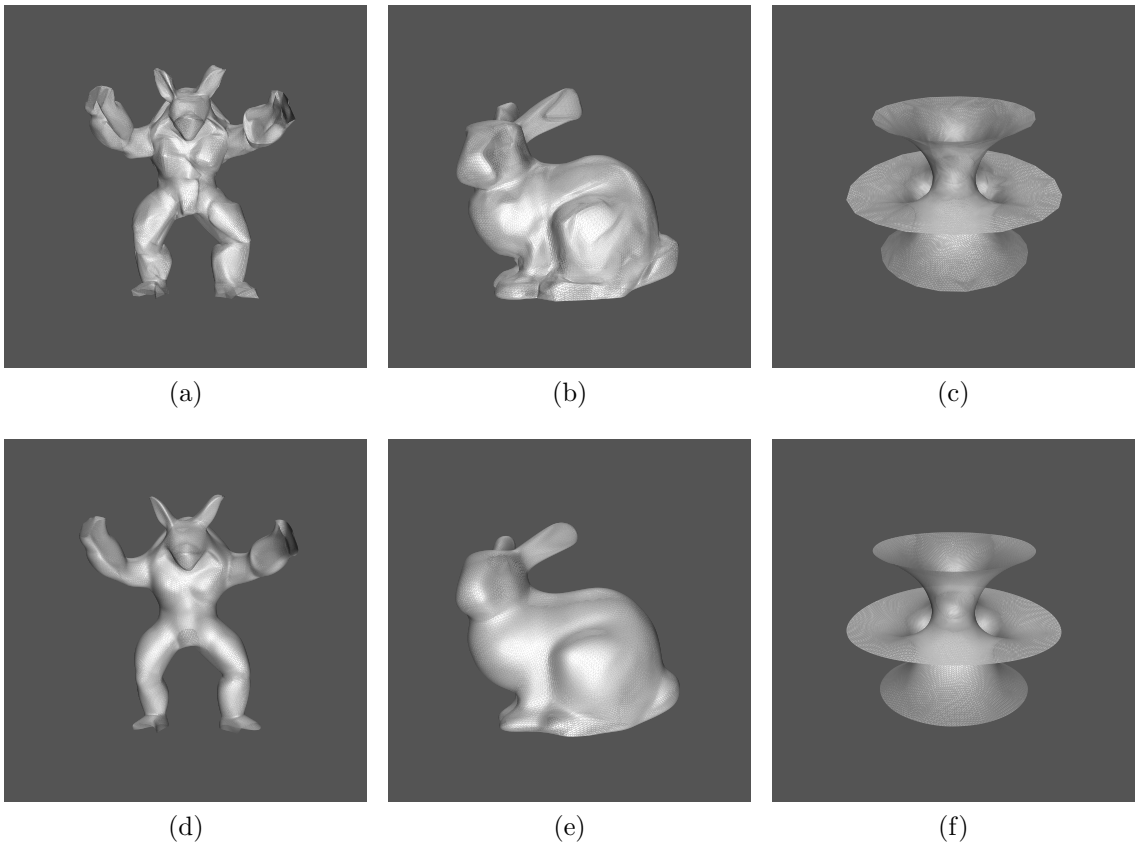
(a) (b) (c)

Figura 3.10: Mallas de prueba antes de la subdivisión: (a) *Armadillo* con 799 triángulos, (b) *Conejo* con 499 triángulos y (c) *Hypersheet* con 917 triángulos.

esquemas de subdivisión empleados, se incluyen en adelante únicamente los valores para el algoritmo *Modified Butterfly*, siendo análogos los logrados con *Loop*.

Los grupos creados en cada caso, para configuraciones de grupos de 2, 3 y 4 anillos de triángulos, se muestran en las gráficas de la Figura 3.12. Como puede apreciarse, la configuración con grupos de 2 anillos crea un mayor número de grupos, pero más equilibrados en cuanto a número de triángulos por grupo que los casos con grupos más grandes. Esto proporcionará un mayor juego para el ajuste dinámico de la carga. Como contrapartida, un mayor número de grupos supone también un número de aristas frontera más elevado, con la consiguiente pérdida de precisión en el cálculo de la subdivisión, al emplear un mayor número de máscaras simples en el proceso (según lo visto en la Subsección 3.4.2).

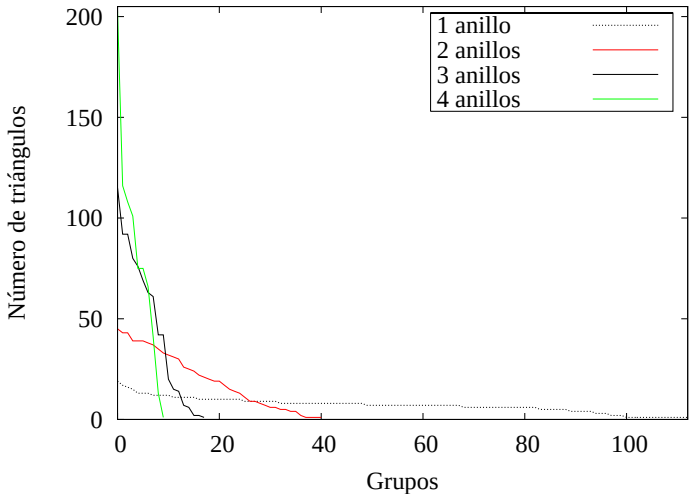
El error cometido con la simplificación introducida se muestra, para el modelo *Armadillo*, en la Tabla 3.2. La medida escogida ha sido el error medio en la subdivisión de la malla mediante el algoritmo *Modified Butterfly* paralelo respecto al resultado de la subdivisión secuencial, tras cuatro iteraciones. Los valores mostrados corresponden a ejecuciones paralelas con grupos de 2, 3 y 4 anillos (columnas de la tabla) para configuraciones de 2, 4 y 16 procesadores (las tres primeras filas de la tabla). La última fila, por su parte, representa el peor caso posible en cada columna; es decir, una configuración de procesadores en la que todas las aristas borde de todos los grupos son aristas frontera, lo que maximiza la simplificación introducida con la versión paralela. Para calcularlo se han utilizado tantos procesadores como grupos resultan en la partición de la malla de triángulos en cada caso: 41, 18 y 10,



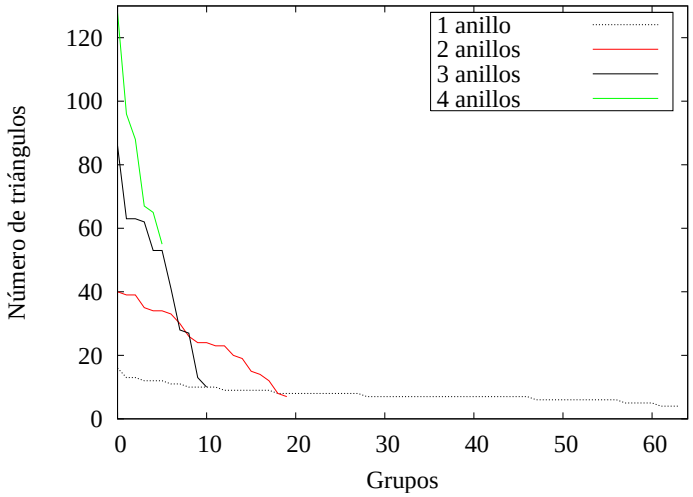
(a) (b) (c)

(d) (e) (f)

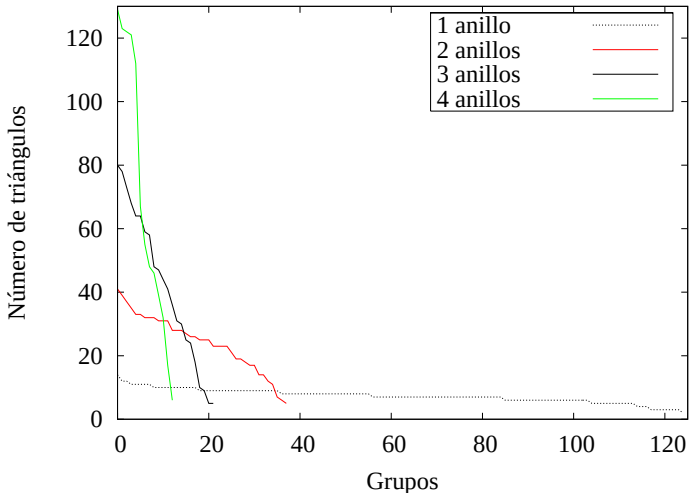
Figura 3.11: Mallas subdivididas tras 3 iteraciones con *Modified Butterfly*: (a) *Armadillo* con 51.136 triángulos, (b) *Conejo* con 31.936 triángulos y (c) *Hypersheet* con 58.688 triángulos; y con *Loop*: (d), (e) y (f).



(a)



(b)



(c)

Figura 3.12: Número de triángulos por grupo (configuraciones de 1, 2, 3 y 4 anillos concéntricos de triángulos por grupo): (a) *Armadillo*, (b) *Conejo* y (c) *Hypersheet*.

Tabla 3.2: Error medio introducido por la simplificación en los patrones utilizados en la implementación paralela del algoritmo *Modified Butterfly*: modelo *Armadillo*.

N. Procs.	2 anillos	3 anillos	4 anillos
2	0.0041	0.0034	0.0013
4	0.0057	0.0038	0.0024
16	0.0064	0.0044	–
N. grupos	0.0065	0.0044	0.0037

Tabla 3.3: Tiempos de ejecución (segundos).

	Armadillo		Conejo		Hypersheet	
	Sec	Par	Sec	Par	Sec	Par
Sunfire 6800	7.46	0.42(24p)	4.48	0.36(16p)	8.56	0.43(24p)
Origin 2000	7.28	0.68(20p)	4.47	0.48(16p)	8.29	0.82(20p)
Fujitsu AP3000	16.85	1.43(12p)	10.24	0.9(12p)	19.71	1.74(12p)

respectivamente.

Como puede apreciarse en la Tabla 3.2, utilizando grupos con más triángulos obtenemos un menor error final en la subdivisión, al reducir el número de vértices frontera, aunque a costa de una menor posibilidad de equilibrar la carga entre los procesadores, ya que nuestro método de balanceo trabaja a nivel de grupo, no de triángulo. El error cometido aumenta con el número de grupos (es decir, reduciendo el número de anillos por grupo) y de procesadores, aunque, como se muestra en la tabla, siempre dentro de unos límites bastante razonables, aun en el peor caso posible.

En la Tabla 3.3 se muestran los tiempos de ejecución para la subdivisión en paralelo de los modelos de la Figura 3.10 tras 5 iteraciones sobre las tres plataformas utilizadas. El tiempo mostrado en la segunda columna, etiquetada como *Par*, corresponde al número de procesadores que aparece entre paréntesis a su lado. Obviamente, el número de grupos obtenido tras la partición limita las configuraciones de procesadores a utilizar. Así, para el modelo del *Conejo* se trabaja con 19 grupos tras la partición, con lo que la configuración máxima de procesadores utilizada en este caso es de 16.

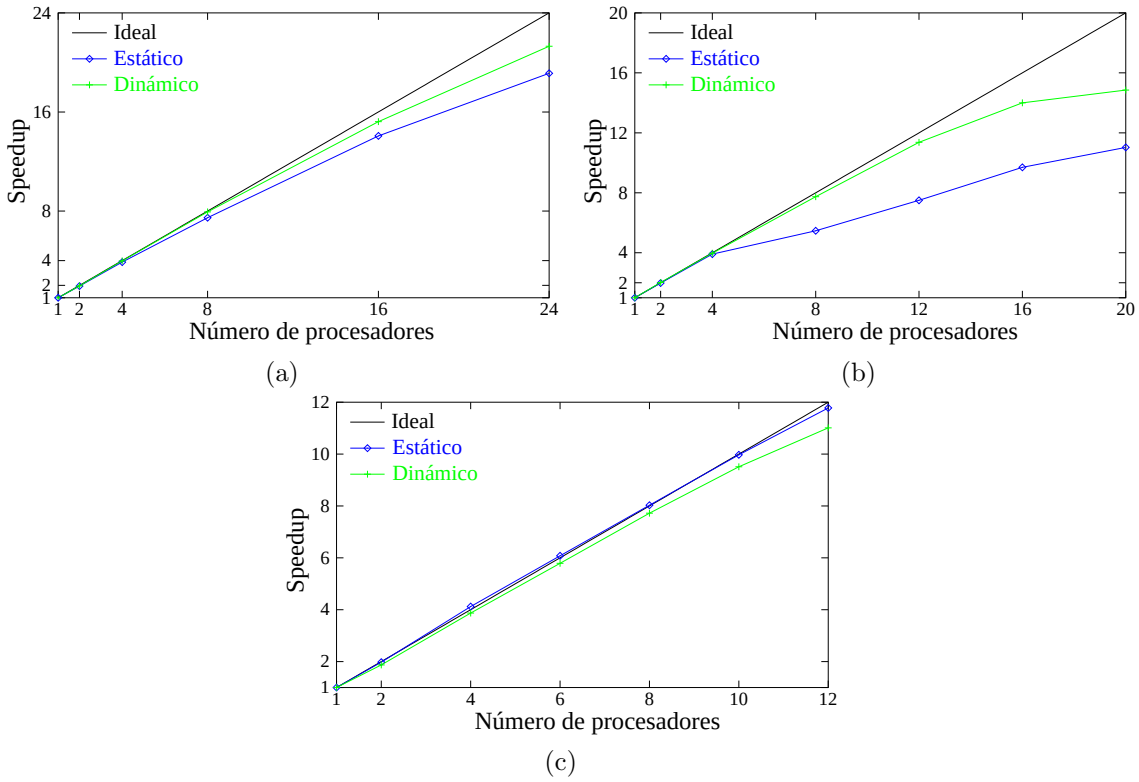


Figura 3.13: Aceleración para para el modelo *Armadillo*: (a) SunFire cluster (b) Origin 2000 y (c) Fujitsu Ap3000.

La aceleración (*speed-up*) obtenida para los tres modelos se muestra en las gráficas de la Figura 3.13, para el *Armadillo*, la Figura 3.14, para el *Conejo*, y la Figura 3.15 para el *Hypersheet*. Estos valores se han obtenido tras la ejecución de cinco iteraciones del método de subdivisión *Modified Butterfly* con grupos de un anillo, lo que proporciona una mayor cantidad de grupos que permiten un mayor equilibrio de la carga; y el tiempo de referencia utilizado para calcularlos es el de una ejecución secuencial pura del esquema de subdivisión, sin algoritmo de agrupamiento alguno. En las gráficas mostradas aparece la aceleración del algoritmo paralelo propuesto junto con el valor logrado desactivando el esquema de balanceo dinámico de la carga implementado, pudiéndose así comparar la bondad de nuestra aproximación frente al simple reparto estático de los cálculos.

Como se puede observar, las gráficas muestran unos resultados muy próximos a la aceleración ideal en los tres sistemas de prueba para nuestro método con el balanceado de la carga computacional activado. Así, se logra el objetivo de equilibrar y

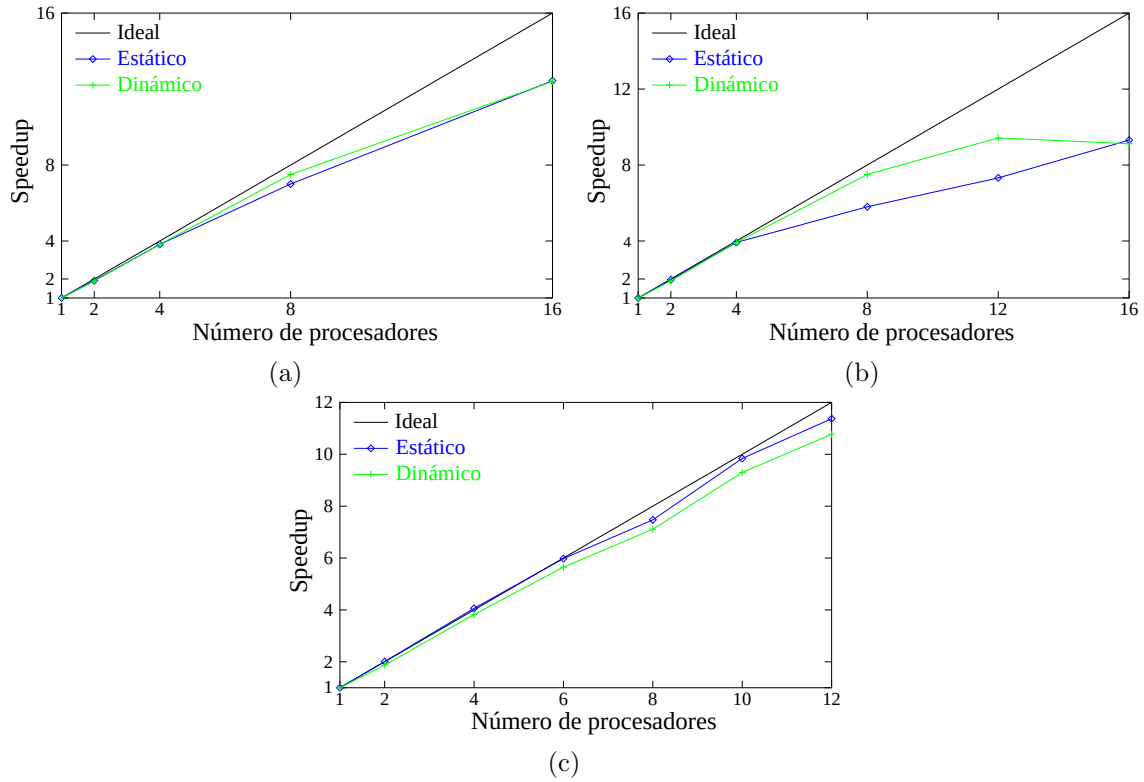


Figura 3.14: Aceleración para para el modelo *Conejo*: (a) SunFire cluster (b) Origin 2000 y (c) Fujitsu Ap3000.

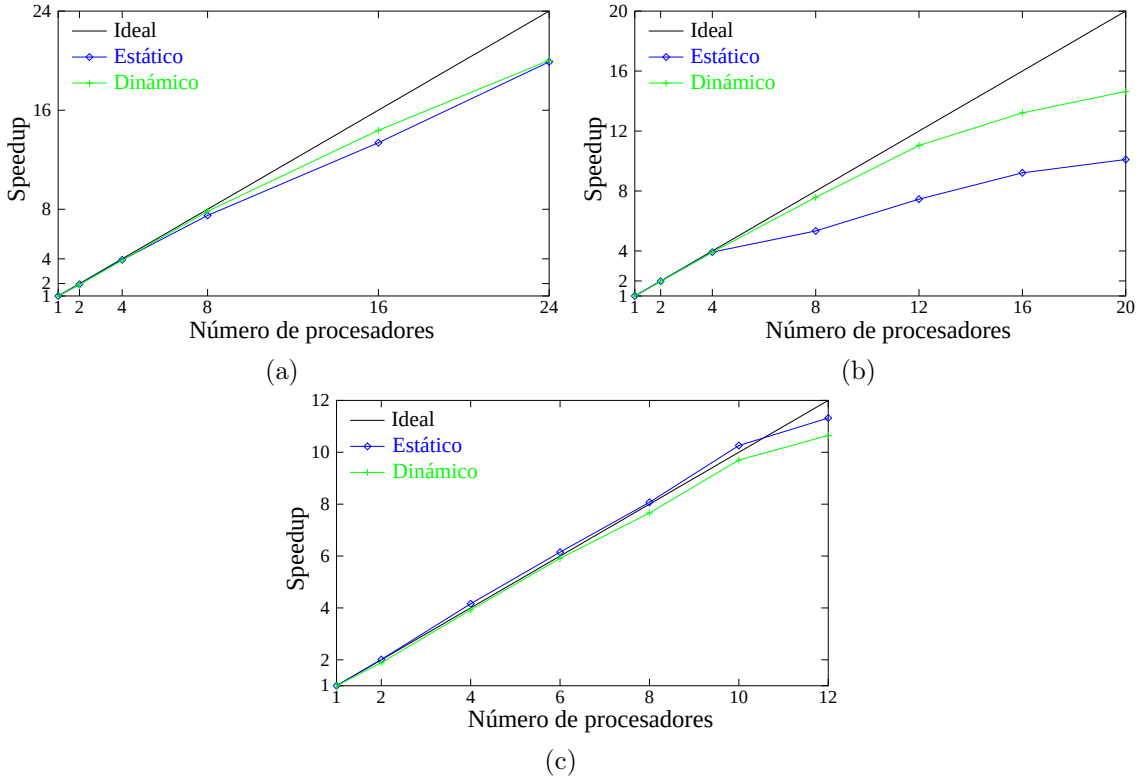


Figura 3.15: Aceleración para para el modelo *Hypersheet*: (a) SunFire cluster (b) Origin 2000 y (c) Fujitsu Ap3000.

maximizar el uso de los procesadores respecto a la distribución puramente estática de los cálculos, como se puede ver por la apreciable diferencia entre los valores de aceleración obtenidos en ambos casos. El margen de mejora logrado con la planificación dinámica es especialmente amplio en el caso del sistema Origin 2000 (Figuras 3.13.b, 3.14.b y 3.15.b), máquina de la que no se disponía en acceso exclusivo durante las pruebas realizadas. En casos como este, en los que uno o varios de los nodos de la máquina de ejecución presentan una mayor carga computacional ajena a nuestro proceso de subdivisión que el resto, el uso de un esquema para el equilibrado de la carga en nuestro proceso tiene especial trascendencia.

En todos los casos mostrados, los resultados comienzan a empeorar cuando se utiliza para la ejecución un número de procesadores próximo al total de grupos obtenidos con la partición de la malla. Lógicamente, al realizarse el equilibrio de la carga con una granularidad a nivel de grupo, un número de grupos demasiado bajo (o demasiado cercano al número de procesadores que se van a utilizar) limita en gran medida la capacidad del balanceo de la carga que se puede conseguir. Así, por ejemplo, en el caso del *Conejo*, que es el modelo más pequeño de los tres empleados, y del que resultan un menor número de grupos, los resultados del esquema de balanceo dinámico de la carga a partir de 8 procesadores son bastante más pobres que los obtenidos con los otros dos modelos.

Capítulo 4

Multiresolución en radiosidad

El realismo que puede alcanzarse en la modelización de la iluminación global en un entorno mediante el método de radiosidad jerárquica se ve en gran parte limitado por la dependencia que este método presenta respecto a la geometría de la escena de entrada. A pesar de la reducción drástica que se logra con el refinado jerárquico respecto al método de radiosidad clásico, para el cálculo de la matriz de interacciones iniciales la complejidad sigue siendo cuadrática sobre el número de polígonos de entrada. Si se pretende utilizar escenas complejas con objetos muy detallados este coste computacional no se puede asumir. Así, la aplicación de técnicas de *clustering* va a permitir procesar escenas de gran tamaño con un coste computacional razonable [95].

La idea fundamental detrás del concepto de *clustering* consiste en la extensión de la jerarquía de superficies utilizada en la aproximación jerárquica a la radiosidad. El método de radiosidad jerárquica construye una jerarquía de polígonos, mediante la sucesiva subdivisión de los polígonos de entrada, que constituyen el nivel más grueso de la jerarquía, que será utilizada para aproximar la distribución de energía en la escena con distintos niveles de precisión. Agrupando los diferentes polígonos gruesos de la escena de entrada, formando conjuntos de polígonos o *clusters*, es posible ampliar *hacia arriba* la jerarquía de elementos de la escena, simplificando interacciones iniciales cuyo detalle es innecesario para el resultado final de la iluminación [26, 95, 127]. El resultado es una jerarquía de *clusters* y superficies con la que se aproxima la distribución de la energía en la escena. Se trata, por tanto, al igual que en el caso del método de radiosidad jerárquica original, de aplicar las soluciones que tradicionalmente han dado buen resultado en el problema de los N-cuerpos en

física gravitacional [129].

Una aproximación ligeramente diferente consiste en la aplicación de técnicas de simplificación de superficies y la utilización de modelos de multiresolución en el cálculo de la iluminación de la escena [36, 47, 80, 144]. Se trata de utilizar versiones de diferente complejidad de los modelos involucrados en la escena, obtenidas mediante la aplicación de métodos de simplificación de superficies [46, 48]. De esta forma, cuando el nivel de detalle geométrico utilizado en una parte de una escena es mucho mayor que la complejidad necesaria para su correcta iluminación, el uso de modelos geométricos más simples independiza de una forma natural el cálculo de la radiosidad de esa innecesaria complejidad geométrica.

En este capítulo presentamos (ver Secciones 4.3 y 4.4) dos novedosas técnicas para el aprovechamiento de las ideas de multiresolución en el cálculo de la radiosidad de una escena. Ambos métodos están basados en la utilización de modelos geométricos sencillos para el cálculo de la radiosidad de la escena; modelos que posteriormente, mediante la aplicación de un esquema de subdivisión de superficies, son reemplazados por objetos más detallados para obtener un acabado final más suave.

4.1. Técnicas clásicas de *clustering* en radiosidad

Una de las características fundamentales que hacen que el método de radiosidad proporcione la mejor simulación de la reflexión difusa de la luz en el entorno es la estrecha relación que el método establece entre la geometría de la escena y la distribución de la energía en la misma. Este comportamiento intrínseco del método constituye también, sin embargo, uno de sus principales inconvenientes, al depender la complejidad del cálculo y del almacenamiento del número de superficies iniciales en la escena.

La extensión del método de radiosidad jerárquico para su uso con una jerarquía con objetos (*clusters*) y superficies permite aumentar drásticamente el tamaño y complejidad de las escenas que es posible procesar mediante dicho método. Así, en lugar de considerar a los polígonos de entrada como el nivel más grueso en el que modelar la interreflexión de la luz, pasamos a tratar interacciones a nivel de grupos de polígonos [33, 65, 127, 131]. La complejidad del cálculo de la radiosidad en la escena sería, ahora, de $O(N + K^2)$, para K polígonos gruesos de entrada con los que se obtiene un total de N elementos finos tras el cálculo de la iluminación.

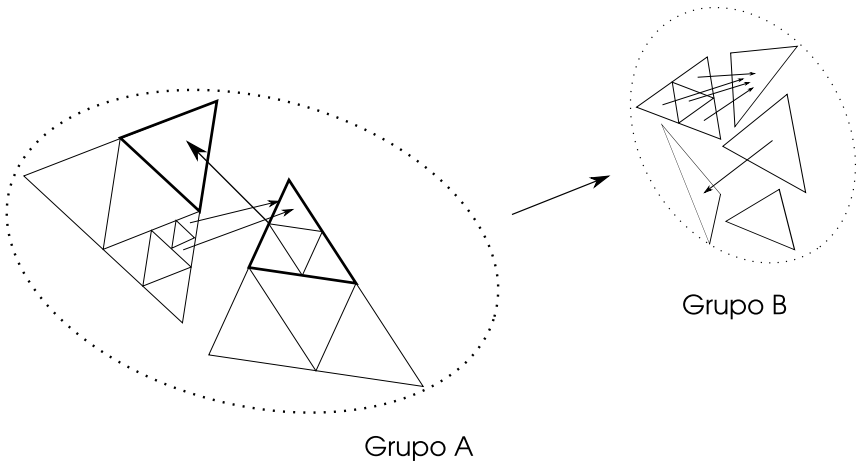


Figura 4.1: Estructura jerárquica de interacciones, combinando el agrupado (*clustering*) y refinado de elementos.

Los métodos de *clustering* clásicos o volumétricos consisten en la construcción de una jerarquía de volúmenes de contorno (*BVH*, como las vistas en la Sección 2.2) que agrupen los polígonos gruesos iniciales de la escena a procesar. La jerarquía de volúmenes de contorno construida va a complementar a las jerarquías de polígonos obtenidas con el proceso de refinado jerárquico, permitiendo insistir en la idea de reducir el tamaño de la matriz de interacción mediante la sustitución de interacciones innecesariamente finas por otras más gruesas (ver Sección 1.3).

En la Figura 4.1 se representa cómo elementos en diferentes niveles de la jerarquía, construida mediante la agrupación y el refinado de los elementos gruesos iniciales, interactúan según sea necesario. En el ejemplo se presentan dos grupos de polígonos gruesos (indicados con puntos: *Grupo A* y *Grupo B*), que están lo suficientemente lejos para que no sea necesario considerar las interacciones individuales entre los polígonos que los forman, sino que utilizando la transferencia de energía entre los conjuntos de polígonos la precisión obtenida en la iluminación es suficiente. En el interior de los conjuntos, sin embargo, las interacciones entre los diferentes polígonos sí son consideradas, y se refinan tanto como sea necesario en cada caso. La Figura 4.2 muestra un fragmento de una escena iluminada en la que los diferentes *clusters* formados con los polígonos de entrada se representan rodeados por una caja que corresponde a su volumen de contorno. La figura muestra en detalle el refinado adaptativo que el método de radiosidad jerárquica aplica sobre los polígonos que forman los objetos *tetera*, *taza* y *silla*.

Como se exponía en el Capítulo 2, dos son principalmente las estrategias en las

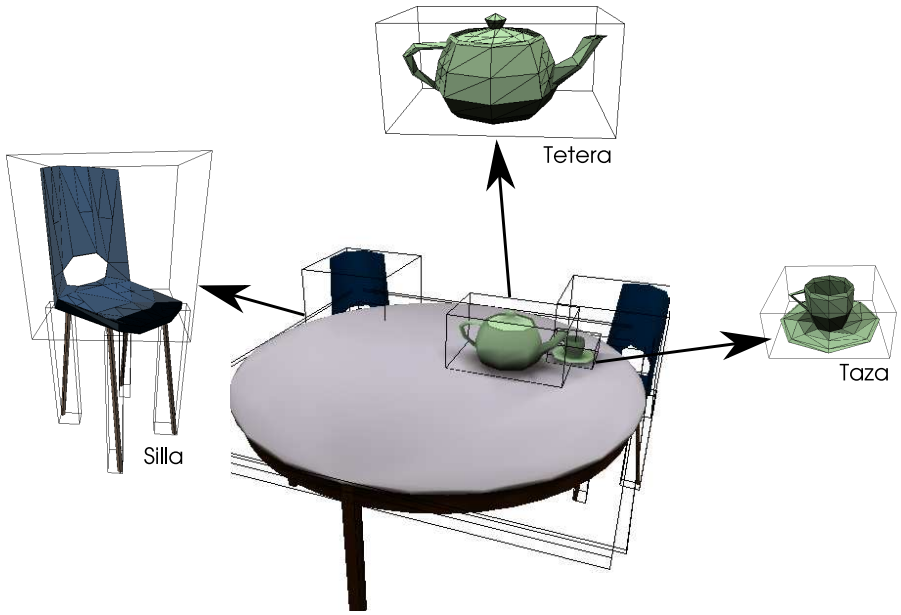


Figura 4.2: *Clusters* de polígonos en una escena iluminada.

que podemos clasificar las distintas técnicas de creación de jerarquías de volúmenes de contorno. Por un lado, la aproximación más sencilla, «desde arriba hacia abajo» (*Top-down*), que consiste en la creación de una estructura de subdivisión espacial en la que se van a introducir los diferentes objetos de la escena [26, 127]. Este tipo de métodos son computacionalmente menos costosos, pero producen en ocasiones resultados que ignoran o desvirtúan gran parte de la estructura de la escena. La aproximación más compleja la constituyen las técnicas «desde abajo hacia arriba» (*Bottom-up*), que de forma incremental construyen objetos más grandes agrupando objetos pequeños [49, 131]. Aunque más costosos, estos métodos pueden ajustarse mejor a la composición real de los objetos que forman la escena, aunque esa mayor sensibilidad a la disposición de los polígonos de entrada también puede tornarse un inconveniente.

A pesar de que el algoritmo original de radiosidad jerárquica es fácilmente extensible para permitir la aplicación de técnicas de *clustering*, la simulación de la interacción de la luz con un conjunto de polígonos no es, sin embargo, una tarea fácil de resolver de forma efectiva [53, 65, 98]. De hecho, las distintas soluciones basadas en técnicas de *clustering* volumétrico se adecúan más al manejo de grandes conjuntos, más o menos organizados, de polígonos que al de modelos poligonales complejos [53, 65]. Además, la calidad de la escena iluminada obtenida depende en

exceso de la capacidad de la estructura jerárquica creada para simular de forma adecuada el flujo de la luz en la escena, lo que provoca una excesiva sensibilidad a cualquier variación en la construcción de la jerarquía de objetos y, por tanto, falta de robustez. Además de todos estos inconvenientes, este tipo de métodos todavía presenta una complejidad demasiado elevada para el tratamiento de escenas demasiado complejas ($K \gg N$), ya que todos los polígonos de entrada y sus jerarquías de descendientes tienen que ser accedidos en cada iteración durante la fase de sincronización del refinado jerárquico, además de tener que mantener almacenada la radiosidad de todos ellos.

4.2. Métodos de multiresolución para radiosidad jerárquica

El uso de técnicas de *clustering* basadas en jerarquías de volúmenes proporciona una simulación razonablemente buena de la distribución de la luz en escenas con una gran cantidad de objetos no excesivamente detallados. Sin embargo, las escenas que incluyen objetos con un elevado nivel de detalle geométrico plantean problemas al tratar de ser procesadas con este tipo de técnicas, tanto de complejidad como para la obtención de una correcta iluminación en la malla de entrada. Teniendo en cuenta que el uso de modelos geométricos altamente detallados proporcionan un acabado final más suave pero que apenas influyen en el cálculo de la iluminación global de la escena, la solución para este tipo de casos pasa por el uso de alternativas basadas en la simplificación de superficies y la utilización de modelos de multiresolución [59, 144].

Los primeros resultados obtenidos con la utilización de este tipo de técnicas para el manejo de mallas muy detalladas en radiosidad se esbozan en [117], aunque todavía de forma totalmente manual. En [59] se desarrolla un método para la aplicación sobre una escena con un alto nivel de detalle de los resultados obtenidos a nivel de iluminación para una versión más simplificada de la misma, basándose en el uso de lo que denominan volúmenes de irradiancia (*irradiance volume*). La interacción del usuario se hace todavía imprescindible para decidir el nivel de simplificación a emplear, y además se trata de un método bastante costoso. En [36] se presenta un algoritmo de radiosidad jerárquica que trabaja sobre mallas con multiresolución y que es capaz de escoger automáticamente el nivel de simplificación apropiado para cada transferencia de energía. Las versiones multiresolución de los distintos objetos

son obtenidas en la fase de modelado de la escena y proporcionadas al algoritmo, constituyendo jerarquías de superficies con distintos niveles de detalle que pasan a formar parte de la jerarquía en la distribución de la energía de la escena propia de la radiosidad jerárquica. El punto débil de esta solución viene dado por el uso de los polígonos de entrada durante la fase de sincronización. Este acceso a los niveles más finos de la jerarquía en cada iteración (como veíamos que también sucedía con los métodos basados en *clustering* de la sección anterior) supone un coste excesivo para el tratamiento de objetos muy detallados en la escena.

La alternativa más utilizada hoy en día, y que soluciona los principales problemas de las soluciones anteriores, es la que se conoce como *Face Clustering* [47, 144]. Este método se basa también en la incorporación de versiones multiresolución de los modelos complejos de entrada, con los que se va a crear una jerarquía de grupos de superficies (*face clusters*). Sin embargo, en lugar de trabajar con un valor escalar de la radiosidad sobre una superficie como en los métodos anteriores y la radiosidad clásica, en este caso se utiliza un vector de radiancia de entrada [1], conservando así la información direccional necesaria que evita la necesidad de ampliar la fase de sincronización de cada iteración jerárquica hasta llegar al nivel de los polígonos de entrada, como sucedía en [36]. De esta forma se consigue, por tanto, una independencia prácticamente completa del cálculo de la iluminación, tanto en tiempo como en almacenamiento en memoria, de la complejidad de la malla de entrada. En cada momento se mantienen en memoria principal únicamente los nodos de la jerarquía de superficies que están interactuando, permaneciendo el resto en memoria secundaria. El trasiego de los elementos de la jerarquía entre memoria principal y secundaria presenta el mayor cuello de botella de esta aproximación.

Para la representación multiresolución de cada objeto complejo de la escena el método utiliza una jerarquía de *clusters* de superficies (*face cluster hierarchy*), construida con una variante del método de simplificación de superficies basado en la contracción de aristas presentado en [46]. Las superficies agrupadas en cada nodo de la jerarquía serán aproximadas, para la aplicación de la ecuación de radiosidad, por una única superficie con su normal ponderada con las áreas que la forman. Las jerarquías de *clusters* de superficies de los objetos de la escena son precalculadas antes del proceso, almacenándose junto con la información geométrica de los mismos.

La Figura 4.3 (tomada de [144]) representa la manera en la que se aplicaría el cálculo de la radiosidad para un conjunto de polígonos, $A - J$, con la aproximación

[1] Como veíamos en el Capítulo 1, denominamos radiancia de entrada (*irradiance*) a la energía incidente en un determinado punto y dirección.

del método de *face clustering* frente a las soluciones más tradicionales. Así, con el método jerárquico clásico (Figura 4.3a) cada uno de los diez polígonos de entrada constituye el nodo más grueso de su correspondiente árbol jerárquico, de forma que si los polígonos A y B son de mayor tamaño que el resto y además están más cerca entre sí, serán subdivididos varias veces, refinando su interacción con interacciones más finas entre elementos de sus árboles jerárquicos. Los polígonos $C - J$, por el contrario, son muy pequeños en relación a los dos primeros y se encuentran a bastante distancia de estos, sin embargo obligan a mantener y tener en consideración un gran número de interacciones en el método clásico de radiosidad, clara fuente de ineficiencia. En la Figura 4.3b se muestra cómo se maneja la situación añadiendo un esquema de *clustering* clásico basado en volúmenes de contorno. Si el *cluster* Q es lo suficientemente pequeño y está lo suficientemente alejado del *cluster* P una única interacción puede ser suficiente para modelar el intercambio de energía entre ambos conjuntos de polígonos. La principal fuente de ineficiencia viene dada, en este caso, por la necesidad de mantener los valores de radiosidad de toda la jerarquía y de actualizarlos en la fase de sincronización de cada iteración. El método de *face clustering* propone sustituir los *clusters* volumétricos por modelos multiresolución para los grupos de polígonos de entrada que representen o pertenezcan a una misma superficie (Figura 4.3c). Con esto se consigue, por un lado, un mejor ajuste a la geometría inicial que con las jerarquías de volúmenes, y por otro lado eliminar por completo toda consideración en el cálculo de la radiosidad de los polígonos iniciales excesivamente finos. Así, en la Figura 4.3c, una representación más gruesa, T, de la malla que forman los polígonos $C - J$ permite independizar totalmente el cálculo de la radiosidad de los detalles innecesarios de los polígonos $C - J$.

El método de *face clustering* toma como punto de partida que la escena ha sido modelada utilizando objetos muy detallados, de los cuales se calculan jerarquías de grupos de polígonos (superficies). Por tanto, es necesario, disponer de un modelado detallado previo de los objetos de la escena. Nuestra aproximación, que se ve plasmada con los dos métodos que describimos en las siguientes secciones, considera como punto de partida objetos más sencillos que serán refinados bajo demanda una vez calculada la iluminación de la escena utilizando los métodos de subdivisión de superficies descritos en el Capítulo 3.

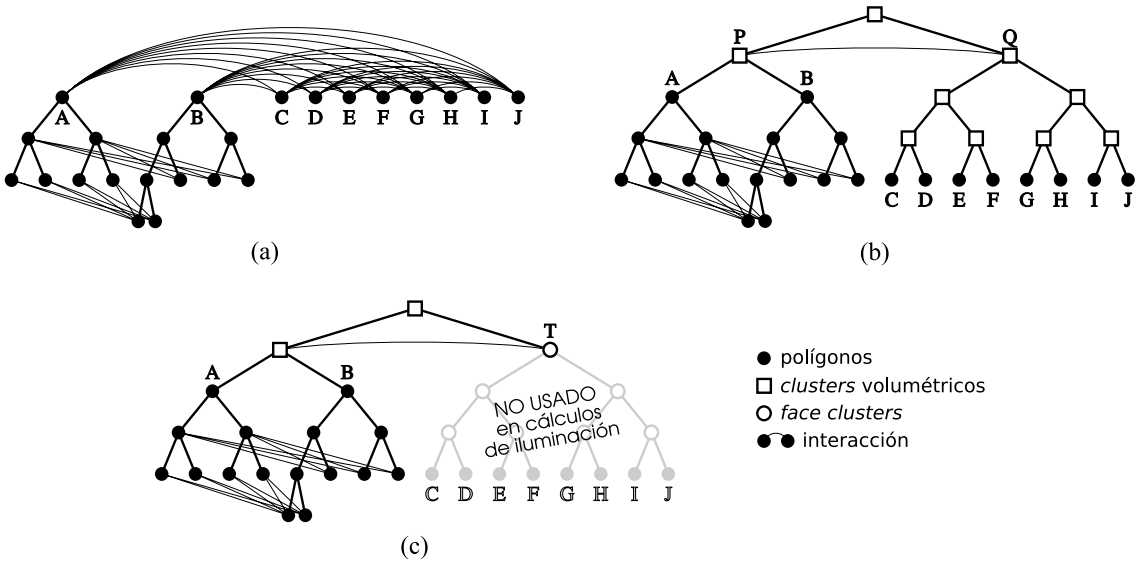


Figura 4.3: Cálculo de la radiosidad para un conjunto de polígonos: (a) radiosidad jerárquica clásica (b) radiosidad jerárquica con *clustering* volumétrico (c) radiosidad jerárquica con *face clustering*.

4.3. Método de radiosidad multiresolución basado en subdivisión de superficies

Nuestra primera aproximación consiste en un método multiresolución para el cálculo de la radiosidad jerárquica que se basa en la subdivisión de superficies [142]. Como hemos visto, la utilización de modelos excesivamente detallados directamente en el cálculo de la radiosidad es computacionalmente inviable y puede no influir en el aspecto realista final en términos de iluminación. Por tanto, nuestro método calcula la iluminación utilizando modelos sencillos que serán refinados posteriormente mediante algún esquema de subdivisión de superficies y que junto a una codificación jerárquica del color (CJC) darán lugar a objetos altamente realistas, tanto en términos de iluminación como geométricos. Hay que destacar que, al contrario que en los métodos previos de multiresolución, en nuestra aproximación no es necesario disponer de los modelos detallados en ningún momento del cálculo de la radiosidad.

En la Figura 4.4 se presenta la estructura del método de radiosidad multiresolución que proponemos. La escena inicial dispone de una serie de objetos simples que sirven como entrada para la *primera fase*, cálculo de la iluminación, donde se obtiene la escena iluminada con una subdivisión adaptativa en función del refinado en la

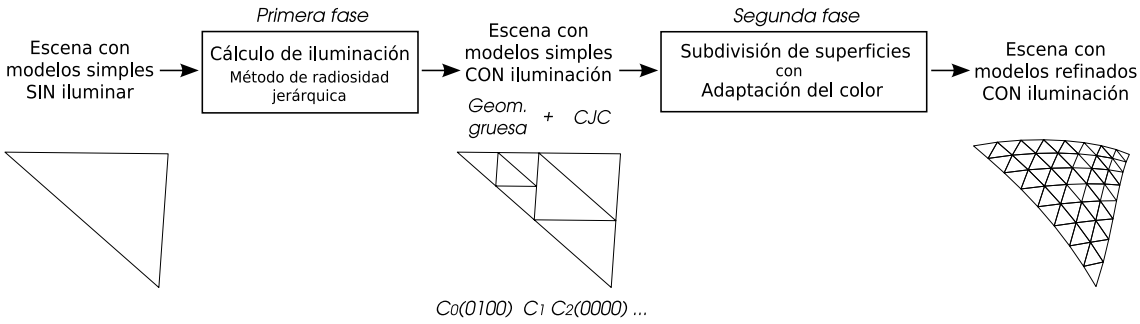


Figura 4.4: Estructura del método de radiosidad con subdivisión de superficies.

propagación de la energía de la escena. Los colores asignados a los diferentes polígonos de la jerarquía adaptativa que resulta del proceso de iluminación se codifican en una estructura CJC, descartando las componentes geométricas de los polígonos. La codificación CJC utilizada es descrita en la Subsección 4.3.1, pero básicamente, y como se muestra en la Figura 4.4, consiste en una lista de colores que corresponde a un recorrido por la jerarquía adaptativa, indicando tras el color de cada elemento si sus hijos son hojas de la jerarquía (0) o están a su vez subdivididos (1) en elementos más finos. Así, C_0 sería el color del elemento raíz, y los cuatro bits 0100 indican que de sus cuatro hijos solo el segundo está subdividido. A continuación iría la descripción del siguiente nivel en la jerarquía y así sucesivamente. La subdivisión aplicada en este caso tiene únicamente en cuenta criterios relativos al cálculo de la iluminación, como la transmisión de energía entre dos polígonos o el grado de visibilidad entre ellos (ver Sección 1.3). Geométricamente consiste en una mera interpolación lineal de los vértices que forman los distintos polígonos. Por tanto, como vemos en la Figura 4.4, los distintos polígonos obtenidos a partir de un mismo elemento grueso se mantienen coplanares, sin aportar más realismo geométrico, aunque el color se calcula de forma apropiada para cada triángulo de la malla.

En la *segunda fase*, subdivisión de superficies con adaptación del color, los objetos para los que se requiera un mayor detalle pueden ser refinados, en función del criterio del usuario o de las necesidades de la aplicación, por ejemplo, teniendo en cuenta el punto de vista del observador. Además, se realiza, junto con el refinado de cada objeto grueso, una asignación de los colores obtenidos tras la aplicación del método de radiosidad. Esta asignación de colores se lleva a cabo en el propio proceso recursivo de subdivisión de superficies, partiendo de la información que la estructura CJC proporciona.

Los esquemas de subdivisión recursiva aplican un refinado a los polígonos con el

que tratan de lograr una malla de superficie final suave, iterando hacia una continuidad C^2 en el caso del esquema de *Loop* (ver Sección 3.2), y C^1 en el de *Modified Butterfly* (ver Sección 3.3). En nuestro método la subdivisión recursiva se aplica directamente sobre cada polígono grueso de entrada de la primera fase. Una vez calculada la geometría detallada final de cada objeto, la jerarquía adaptativa de colores codificada en la estructura CJC, obtenida tras el cálculo de la iluminación de la escena, se utiliza para obtener los colores de cada triángulo de la malla. Los algoritmos de subdivisión utilizados pueden ser *Loop* o *Modified Butterfly* según el algoritmo que requiera el modelo. Opcionalmente, se puede aplicar una subdivisión recursiva adaptativa de la malla geométrica en aquellos puntos donde un test de planitud, un test de variación de los vectores normales o el punto de vista lo aconsejen [2, 17].

En la Figura 4.5 se muestra un ejemplo que ilustra ambos criterios de subdivisión, el llevado a cabo por el método de radiosidad jerárquica y el que resulta de la aplicación de un esquema de subdivisión recursiva, aplicados sobre un mismo polígono. En el primer caso, Figura 4.5a, el árbol jerárquico obtenido resulta de la precisión utilizada en la representación de la distribución de la radiosidad en la escena. En este caso la geometría de los objetos formados por los distintos polígonos de la escena no varía, y simplemente se realiza una discretización más fina de los elementos finitos que almacenan los valores de radiosidad según se haga necesario. La subdivisión recursiva, Figura 4.5b, itera hasta el nivel de suavidad deseado mediante la introducción de vértices nuevos, buscando alcanzar la continuidad en la superficie de la malla (acercándose a la superficie curva ideal) y, si se trata de un esquema de subdivisión aproximada como *Loop*, también modificando, de forma acorde, los vértices existentes.

En cuanto a la asignación de colores sobre la malla de polígonos final, los colores de los triángulos obtenidos con la subdivisión recursiva y que tienen correspondencia con los subtriángulos presentes en la jerarquía adaptativa obtenida tras el proceso de iluminación se asignan directamente; en el caso de triángulos nuevos introducidos por la subdivisión no adaptativa, los colores utilizados serán los del triángulo padre, al no haber un cambio significativo de la iluminación de la superficie.

Para realizar la asignación de ambas estructuras se utiliza la notación CJC que explicamos en la Subsección 4.3.1. La estructura jerárquica obtenida con el cálculo de la iluminación se codifica por completo, ya que el nivel de subdivisión geométrica que será requerido puede ser determinado a posteriori por el esquema de subdivisión aplicado en la segunda fase. De esta forma, con la utilización de una interfaz bien definida como la descrita, se logra independizar de forma efectiva ambas fases, ilumi-

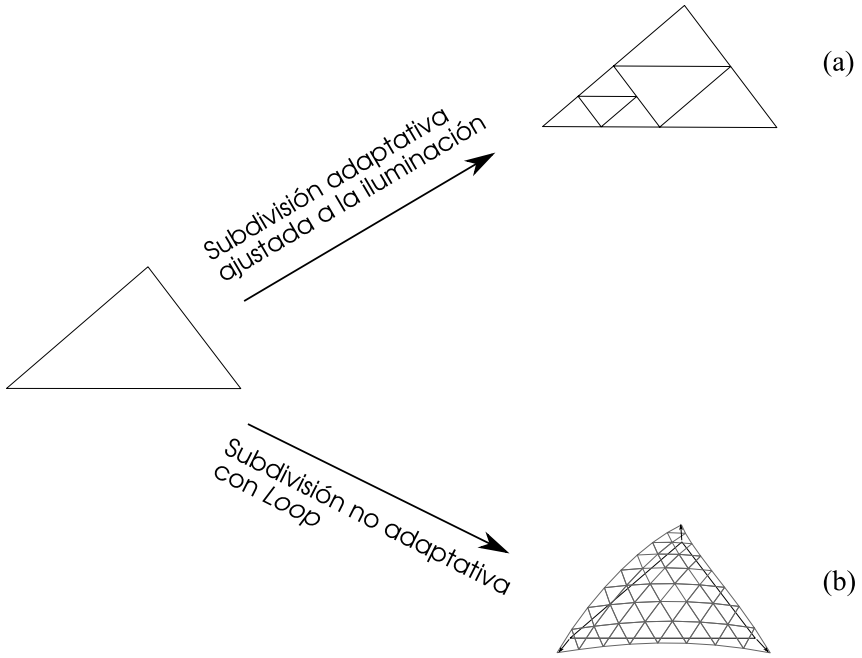


Figura 4.5: Refinado adaptativo producto de la iluminación (a) frente a refinado suave de superficies (b).

nación de la escena y subdivisión de superficies. Así, la información de iluminación definida por las mallas gruesas junto con la codificación CJC puede, por ejemplo, utilizarse en el caso de un recorrido por la escena para mostrar diferentes resoluciones de los objetos en función de su posición respecto al observador, aprovechando que la iluminación obtenida con el método de radiosidad es independiente del punto de vista del observador.

La equiparación de los subtriángulos obtenidos mediante el proceso de subdivisión de superficies con los triángulos que resultan del refinado adaptativo de la radiosidad jerárquica, por simple interpolación lineal, no supone apenas pérdida alguna de calidad en el resultado final, como comprobaremos en la Sección 4.5. En la siguiente sección comprobaremos, sin embargo, cómo el uso de varios valores de radiosidad, correspondientes a distintas direcciones, sobre cada polígono de una malla gruesa nos va a permitir una aproximación todavía más cercana al color que se obtendría con la aplicación del método de radiosidad directamente sobre una malla fina, con apenas un pequeño coste computacional adicional.

Tres son principalmente las posibilidades que se nos plantean para la implementación de la segunda fase de nuestro método, subdivisión de superficies con adaptación

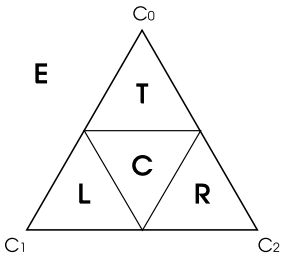


Figura 4.6: Etiquetado del refinado de un triángulo.

del color. Por un lado, la que sería una solución *software*, utilizando el método de subdivisión de superficies paralelo presentado en la Sección 3.4. Una segunda opción sería la implementación de una extensión *hardware* a la tarjeta gráfica, propuesta ya utilizada en otros ámbitos como la subdivisión de superficies mediante mapas de desplazamiento [2, 17] o *NURBS* [38]; mientras que la tercera alternativa consistiría en la implementación en la propia GPU del esquema de subdivisión, utilizando las características del nuevo hardware gráfico programable. Precisamente, dentro esta última alternativa, existe una propuesta reciente para la implementación de la subdivisión de Catmull-Clark en una GPU programable [125]. Como futura tarea a considerar para la extensión del trabajo realizado en este capítulo quedaría la propuesta de alternativas para la subdivisión de superficies con adaptación del color basadas en las dos últimas aproximaciones comentadas.

4.3.1. Estructura CJC

Para la interfaz entre las dos fases de nuestro método, cálculo de la iluminación y subdivisión de superficies (ver Figura 4.4), vamos a utilizar una estructura con los colores de la jerarquía de triángulos obtenida tras la radiosidad jerárquica. En esta estructura, a la que llamamos codificación jerárquica del color (CJC), cada triángulo, P_i, se representa con la siguiente estructura:

$$P_i = E_i\ (T_i\ L_i\ C_i\ R_i)$$

donde E_i es el color del triángulo y el término entre paréntesis consiste en un bit de información para cada triángulo hijo, indicando con un valor 1 o 0 si ese triángulo ha sido subdividido o no en el refinado jerárquico. En nuestro caso, para un árbol de refinado cuaternario, las etiquetas T, L, C y R se refieren a los subtriángulos superior, izquierdo, central y derecho, respectivamente (ver Figura 4.6, donde C_0,

C_1 y C_2 son las coordenadas del triángulo). Con esta notación, la jerarquía de subdivisión adaptativa para un triángulo de entrada se puede representar con una lista secuencial, J, correspondiente a un recorrido *en anchura* del árbol jerárquico, con todos los nodos de un nivel enumerados antes que los del siguiente, y el orden T, L, C, R para los triángulos producto del refinado de un triángulo del nivel inmediatamente superior. Un bit inicial, s, indica si el triángulo raíz es subdividido o no. La lista resultante sería de la forma:

$$J = s\, E_0\ (T_0\ L_0\ C_0\ R_0)\ \ E_1\ (T_1\ L_1\ C_1\ R_1)\ \ E_2\ (T_2\ L_2\ C_2\ R_2)\ldots \tag{4.1}$$

donde los subíndices identifican unívocamente al triángulo en la jerarquía, de forma que los triángulos descendientes del elemento i en un árbol cuaternario ($T_i\ L_i\ C_i\ R_i$) serían etiquetados como $i \times 4 + 1$, $i \times 4 + 2$, $i \times 4 + 3$ e $i \times 4 + 4$. Para los triángulos hojas simplemente aparecería el término E correspondiente.

Considerando la jerarquía adaptativa mostrada en la Figura 4.7a, con cada número representando el color del triángulo correspondiente de la jerarquía, obtendríamos la siguiente representación:

$$J = 1\, E_0\ (0\ 1\ 0\ 0)\ \ E_1\ \ E_2\ (0\ 0\ 0\ 0)\ \ E_3\ \ E_4\ \ E_9\ \ E_{10}\ \ E_{11}\ \ E_{12} \tag{4.2}$$

que sería pasada, junto con la geometría de la malla gruesa de partida, a la segunda fase, que se encargaría de obtener la malla suave de la Figura 4.7b utilizando los colores obtenidos para la Figura 4.7a.

La correspondencia entre los colores codificados en la estructura CJC y la malla fina obtenida tras la segunda fase de nuestro método va a permitir obtener el acabado final suave. El procedimiento a seguir consiste en recorrer la jerarquía de color codificada en la lista CJC, asignando el color de cada uno de sus nodos hoja al subárbol de subdivisión recursiva correspondiente, como se ejemplifica en la Figura 4.7. Como vemos en el ejemplo, el color del triángulo hoja 1 de la jerarquía adaptativa (Figura 4.7a) es asignado a los triángulos del subárbol correspondiente del refinado recursivo (Figura 4.7b), que en este caso tiene un total de 16 triángulos. Por su parte, el triángulo al que corresponde el color E_2 en la Ecuación 4.2, subtriángulo L del triángulo raíz de la Figura 4.7a, presenta un nivel de subdivisión adaptativa más fino, por lo que son los colores de sus triángulos hijo, 9, 10, 11 y 12,

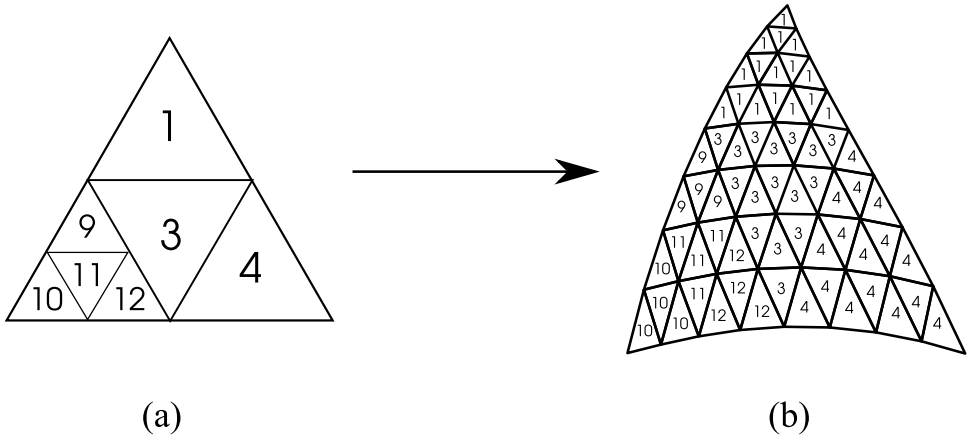


Figura 4.7: Triángulos y asignación de color (a) después de la primera fase (b) después de la segunda fase.

los que se asignan en el nivel de subdivisión recursiva correspondiente, en este caso con cuatro elementos cada uno. Cuando se trate el caso de triángulos en el mismo nivel de subdivisión en ambas jerarquías, adaptativa y recursiva, la correspondencia sería uno a uno.

4.4. Método de radiosidad multiresolución basado en agrupamiento de normales

El segundo método que proponemos en este trabajo para simplificar el cálculo de la radiosidad en escenas que contienen objetos con un nivel de detalle excesivo se basa en la simplificación de los cálculos de radiosidad basado en el agrupamiento de normales.

El objetivo continúa siendo evitar la aplicación del método de radiosidad jerárquica, con complejidad cuadrática sobre el número de polígonos de entrada para el cálculo de la matriz de interacción inicial, directamente sobre una escena con objetos muy detallados. Al igual que en el caso anterior, el método de radiosidad jerárquica se aplica sobre versiones más simples de estos modelos. Los colores obtenidos con la iluminación son traspasados a continuación a modelos detallados, que son calculados sólo en ese momento aplicando alguno de los esquemas de subdivisión recursiva paralelos descritos en el Capítulo 3. En ningún caso es necesario mantener previamente almacenados los modelos detallados. A diferencia del primer método, sin embargo,

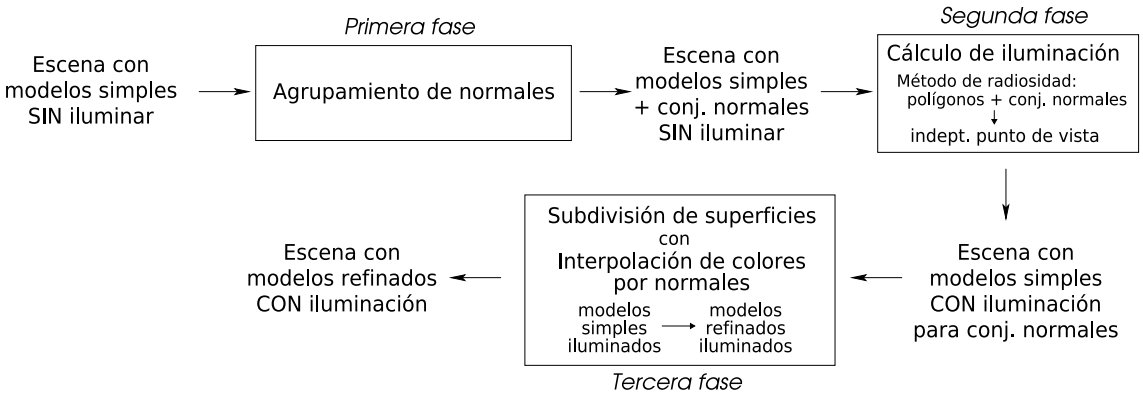


Figura 4.8: Estructura del método de radiosidad basado en agrupamiento de normales.

en este caso se realizan varios cálculos de radiosidad para cada polígono de la malla gruesa, considerando diversas orientaciones según el futuro refinado de ese elemento, lo que nos permitirá afinar más la iluminación sobre el objeto final detallado.

4.4.1. Agrupamiento de normales

La Figura 4.8 muestra de forma esquemática los distintos pasos que sigue el método. Se distinguen tres fases principales, que aparecen enmarcadas en la figura. En la *primera fase*, se dispone de una escena con objetos poco detallados sobre la que se desea calcular la iluminación mediante el método de radiosidad. Para aquellos objetos de la escena que requieren un acabado final más fino, con un mayor detalle geométrico, se aplicará un esquema de subdivisión recursivo, *Loop* o *Modified Butterfly* según sea lo deseado para cada modelo. Sin embargo, al igual que en el método anterior, en lugar de refinar los objetos antes del cálculo de la iluminación, lo que introduce un coste extremadamente difícil de asumir, nuestro método va a posponer la obtención de las mallas finas a una fase posterior al cálculo de la iluminación. En este caso, se añade una fase previa en la que, para cada objeto susceptible de ser posteriormente refinado, se realiza un cálculo de las normales que resultarían de calcular su subdivisión por el esquema de subdivisión recursiva que luego se utilice para refinarlo. Los conjuntos de normales obtenidos permitirán calcular la iluminación recibida por estos polígonos para cada una de las direcciones de estos vectores.

Para la obtención de los conjuntos de vectores normales necesarios, si con cada

paso de subdivisión de un triángulo se obtienen cuatro nuevos triángulos, con su correspondiente normal, en esta fase se va a realizar una selección adaptativa de las normales necesarias para obtener una descripción adecuada de la curvatura del objeto detallado final. Para ello, en cada paso de la subdivisión recursiva se van a ir aplicando una serie de tests que miden la mejora en la calidad de la malla si esa iteración de subdivisión fuese realizada dado un umbral de calidad. Los umbrales utilizados son dependientes de la calidad deseada para cada malla, por lo que son precalculados y proporcionados junto con la información de la malla geométrica de cada objeto grueso. Así, muchas zonas prácticamente planas, con la misma normal o muy similar, no aportan calidad en términos de iluminación y pueden ser consideradas como un único polígono a efectos de su iluminación. En estos casos el número de normales a utilizar para el cálculo de la iluminación será menor, pudiendo incluso ser suficiente con un único vector.

Para su utilización en la primera fase de nuestro método proponemos los siguientes tests para la selección recursiva de normales, basándonos en su simplicidad y sus buenos resultados en términos de calidad de la imagen final (como veremos en la Sección 4.5):

- *Test de media* ($test_1$). Antes de subdividir un polígono, este test compara la normal del polígono original con la media de las normales de los polígonos que se obtendrán fruto de la subdivisión:

$$test_1 : \left\| \vec{n} - \frac{\vec{n}_T + \vec{n}_L + \vec{n}_C + \vec{n}_R}{4} \right\| > t_1$$

  en donde $\vec{n}$ es el vector normal del triángulo a subdividir y $\vec{n}_T$, $\vec{n}_L$, $\vec{n}_C$ y $\vec{n}_R$ son las normales de los triángulos que se obtienen tras las subdivisión. Si no se supera el umbral t_1, que marca la diferencia mínima requerida, pasamos a aplicar el $test_2$, que si tampoco es superado satisfactoriamente hará que nos quedemos con la normal del polígono que se estaba probando a subdividir. En caso contrario, cuando se sobrepasa el umbral t_1, el test se aplica recursivamente a cada uno de los triángulos hijos que se obtienen con la subdivisión.

- *Test de diferencia* ($test_2$). Este test determina si la diferencia entre las normales de los triángulos que se obtienen tras una iteración de subdivisión es mayor que un determinado umbral t_2. Para ello se comparan dos a dos las normales obtenidas:

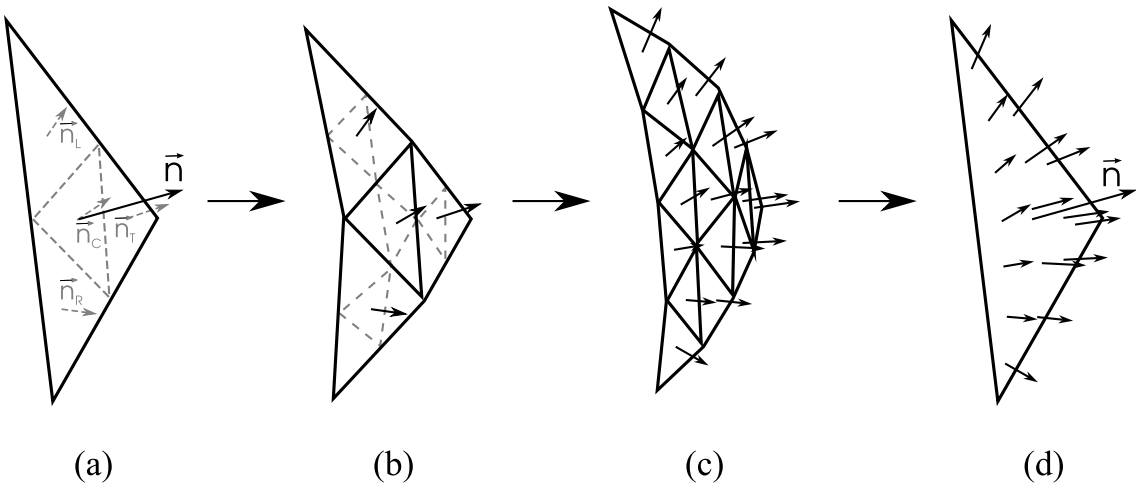


Figura 4.9: Obtención de un conjunto de vectores normales para un polígono.

$$test_2 : \ \|\vec{n}_i - \vec{n}_j\| > t_2, \ \forall \vec{n}_i, \vec{n}_j \in \{\vec{n}_T, \vec{n}_L, \vec{n}_C, \vec{n}_R\}$$

de manera que el test se considera superado si la diferencia entre las cuatro normales sobrepasa el umbral t_2, en cuyo caso se continúa con los triángulos hijos, a los que se aplicaría el $test_1$, comenzando de nuevo el proceso.

En la Figura 4.9 se muestra este ejemplo de agrupamiento de normales: partiendo de un polígono grueso con normal $\vec{n}$, se comprueba si los vectores normales que se obtendrán con la subdivisión del polígono, $\vec{n}_T$, $\vec{n}_L$, $\vec{n}_C$ y $\vec{n}_R$, superan los dos tests aplicados (Figura 4.9a). En caso afirmativo, el proceso se aplica recursivamente sobre cada uno de los cuatro subtriángulos obtenidos con la subdivisión (Figura 4.9b y Figura 4.9c). Una vez llegado el punto en el que las normales de todos los subtriángulos quedan por debajo de los umbrales escogidos para los test aplicados, o que se alcanza un nivel de profundidad máxima fijado para la subdivisión, el resultado es un conjunto de normales que se asociarán al polígono grueso original (Figura 4.9d, en este caso con un total de 16 vectores para al polígono inicial).

4.4.2. Cálculo de la iluminación

Aplicada la primera fase, disponemos de una escena en la que se integran diversos objetos formados por mallas geométricas gruesas y un conjunto de vectores normales

asociado a cada triángulo de esas mallas, $N_i = \{\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_{mi}\}$, donde m_i es el número de normales necesario para el triángulo i.

A continuación se aplica el método de radiosidad jerárquica sobre la escena con objetos gruesos, aunque en este caso obteniendo, en cada polígono, un valor de radiosidad para cada uno de los vectores de su conjunto asociado de normales (*segunda fase*). De esta manera, la distribución de la energía de la escena mediante el método de radiosidad se lleva a cabo utilizando los modelos geométricos simples, rebajando drásticamente el coste temporal y de almacenamiento en memoria que su aplicación sobre mallas más detalladas supondría. En este caso, sin embargo, además de la dirección del vector normal a la superficie de los polígonos de la malla de entrada, con el conjunto de vectores asociados a cada polígono disponemos de un conjunto adicional de direcciones significativas para ese polígono, lo que nos va a permitir una mejor aproximación posterior a la iluminación sobre las mallas detalladas que se obtendrán como resultado del refinado de los objetos gruesos en la siguiente fase. Una pequeña modificación en el método de radiosidad jerárquica tradicional nos permitirá calcular la energía que llega a cada polígono a través de la dirección de cada una de las normales asociadas a ese polígonos, además de para la suya propia. Para ello utilizamos la expresión clásica de la ecuación de radiosidad, Ecuación 1.3, para cada una de las direcciones $\vec{n}_{li}$ del conjunto de vectores normales N_i, calculando en cada caso el factor de forma correspondiente:

$$F_{ij}^{l} = \frac{(\vec{w} \cdot \vec{n}_{li})(\vec{w}' \cdot \vec{n}_j)}{\pi r^2 + A_j} \; A_j \; V(i,j), \tag{4.3}$$

en donde F_{ij}^{l} representa al factor de forma para la transmisión de la energía entre los elementos i y j en la dirección del vector $\vec{n}_{li}$ asociado a i. El término r corresponde a la distancia entre los dos polígonos, y $\vec{w}$ y $\vec{w}'$ son los vectores de dirección correspondientes a la línea imaginaria que uniría ambos triángulos (ver Figura 1.4).

Como resultado del proceso de iluminación, para un polígono grueso i con un conjunto muestreado de m_i vectores, obtenemos un conjunto de m_i valores de radiosidad, $B_i = \{B_1^i, B_2^i, \ldots, B_{mi}^i\}$, que nos permitirán una mayor aproximación de la iluminación de los polígonos finos correspondientes. Cada uno de estos valores escalares se obtendría mediante la expresión:

$$B_i^l = \sum_j F_{ij}^{l} \; B_j \tag{4.4}$$

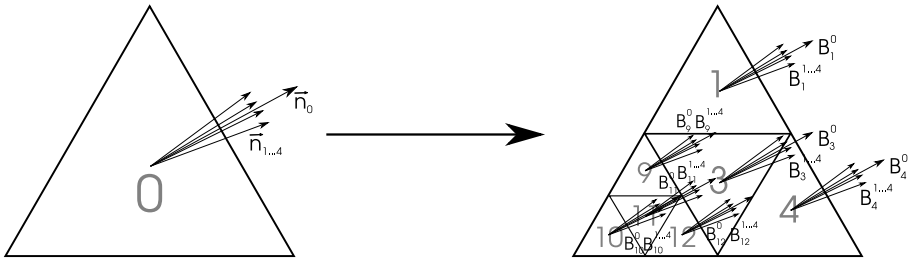


Figura 4.10: Cálculo de la radiosidad utilizando un conjunto de normales adicionales en los polígonos.

siendo B_i^l la radiosidad obtenida en el polígono i en la dirección del vector normal $\vec{n}_l$ y B_j la radiosidad en el polígono j, que es ponderada utilizando la expresión de la Ecuación 4.3 para el factor de forma correspondiente. En la Figura 4.10 se representa un ejemplo que muestra el resultado de la aplicación del cálculo de la radiosidad jerárquica sobre un polígono de entrada con un conjunto de cuatro vectores adicionales producto de la primera fase del método, $\vec{n}_1$, $\vec{n}_2$, $\vec{n}_3$ y $\vec{n}_1$, que se añaden a su propio vector normal, $\vec{n}_0$. Cada elemento resultante de la subdivisión adaptativa del refinado jerárquico de radiosidad va a obtener un valor de radiosidad, B_i, para cada una de estas cinco normales, B_0^i, B_1^i, B_2^i, B_3^i y B_4^i.

4.4.3. Subdivisión de superficies con asignación de colores por normales

La última etapa del proceso (*tercera fase*) corresponde, como en el caso del método anterior, a la obtención de objetos suaves para su incorporación en la escena ya iluminada. De nuevo se aplican los esquemas de subdivisión recursiva *Loop* y *Modified Butterfly*, aunque ahora el proceso de asignación del color a los polígonos finales va a utilizar la información de las normales y los colores obtenidos de las fases anteriores. Para la interfaz entre las fases segunda y tercera se utiliza ahora una variante de la codificación CJC descrita en la sección anterior, al tener que incorporar un conjunto de valores de color para cada nodo de la jerarquía, en lugar del único valor que veíamos en el método anterior. La codificación quedaría entonces, para un elemento i de la jerarquía adaptativa, como sigue:

$$P_i = E_i^{0...mi}\ (T_i\ L_i\ C_i\ R_i)$$

El número de valores de color, $m_i + 1$, de todos los elementos de una misma jerarquía adaptativa será el mismo, y corresponde al número de vectores adicionales de dirección que se obtuvieran para el elemento raíz de la jerarquía (elemento grueso de entrada) tras la primera fase. Ese número y los vectores de dirección correspondientes a los que se asocia cada valor de color se encontrarían en la codificación del primer elemento de la jerarquía, tras el bit que indicaba si el elemento era subdividido o no. La lista resultante sería, por tanto, de la forma:

$$J = s\, m_i\, N_i\ E_0^{0\ldots mi}\ (T_0\ L_0\ C_0\ R_0)\ \ E_1^{0\ldots mi}\ (T_1\ L_1\ C_1\ R_1)\ldots$$

donde los subíndices corresponden al identificador del triángulo en la jerarquía adaptativa, y los superíndices el vector de dirección al que corresponde el valor. El bit s conserva el mismo significado que en la Ecuación 4.1, indicando si el polígono grueso de partida está o no subdividido. Nótese que, aunque se enumeran los $m_i + 1$ colores de cada triángulo de la jerarquía, las normales listadas son solo las m_i obtenidas de la primera fase del método. La normal del propio polígono grueso (raíz de la jerarquía) no es necesario incluirla, al ser fácilmente calculable a partir de la geometría del polígono, que acompaña al código CJC como entrada para la tercera fase, como veíamos en la Sección 4.3.1.

Una vez que en la tercera fase un polígono es subdividido, obteniéndose la correspondiente malla detallada, la asignación de los colores de la estructura CJC a los elementos de esa malla se realiza de forma análoga a la vista en la Sección 4.3 para el método anterior, recorriendo la estructura adaptativa codificada en la lista CJC y equiparando sus triángulos con los de la malla obtenida mediante subdivisión recursiva (ver Figura 4.7). La diferencia, en este caso, radica en la existencia de varios colores posibles para cada elemento, de los cuales se seleccionará el que corresponda al vector $\vec{n}_j$ asociado cuya dirección sea más cercana a la dirección del vector normal del propio elemento refinado. Para ello se realiza el producto escalar entre el vector normal a la superficie del triángulo P_i, $\vec{n}_i$ y los m_i+1 vectores a los que corresponden los valores de color $E_i^{0\ldots mi}$ posibles, escogiendo el color E_i^j tal que

$$\vec{n}_i \cdot \vec{n}_j < \vec{n}_i \cdot \vec{n}_k,\ \forall \vec{n}_k \in N_i$$

En la siguiente sección mostramos los resultados obtenidos con la aplicación de los dos métodos descritos en este capítulo.

Figura 4.11: Escena *teatime*.

4.5. Resultados experimentales

Para probar nuestra solución de radiosidad jerárquica con multiresolución basada en la subdivisión de superficies se ha utilizado la escena de prueba de la Figura 4.11 (*teatime*). Entre los distintos objetos presentes en esta escena, nuestras pruebas se centran en tres modelos concretos. Estos objetos se encuentran, en principio, modelados de manera sencilla, pero son adecuados para ser refinados y permitir así la obtención de versiones más suaves de los mismos, con el objetivo de dotar de mayor realismo a la escena final a *renderizar*. En la Figura 4.12 se muestran estos tres objetos: una tetera con 240 triángulos, una taza con 208 triángulos, y una cucharilla con 126 triángulos, junto con sus correspondientes versiones refinadas tras una y dos iteraciones con el esquema de subdivisión recursivo de *Loop*.

Así pues, para probar nuestros dos métodos hemos trabajado con versiones de la escena con diferente complejidad, según se incluyan los tres objetos descritos con su mallado básico, lo que denominaremos como *escena de nivel base*, tras una iteración de subdivisión recursiva (los objetos son, por tanto, cuatro veces más complejos), *escena de nivel uno*, o después de dos iteraciones de subdivisión (objetos 16 veces más complejos), *escena de nivel dos*. Esta última escena (nivel dos) se va a tomar como escena ideal o de referencia, lo que nos permitirá comparar los resultados obtenidos con la aplicación de nuestros métodos sobre las otras dos escenas más sencillas.

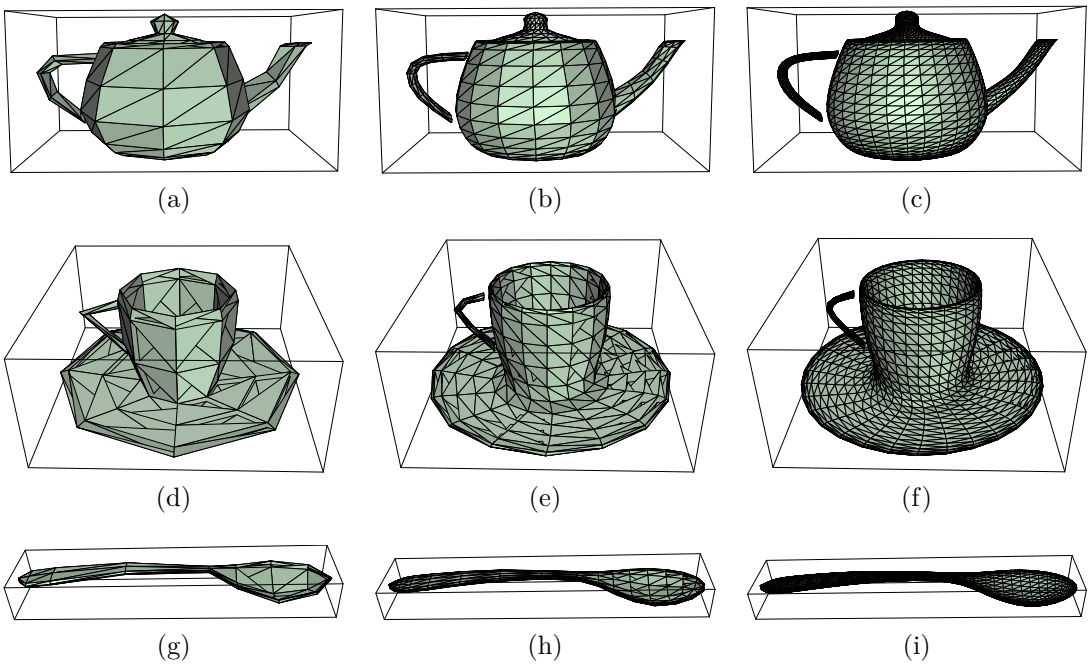


Figura 4.12: Modelos refinados en la escena: (a) (b) (c) tetera - modelo base, *Loop* 1 iter. y *Loop* 2 iter. (d) (e) (f) taza - modelo base, *Loop* 1 iter. y *Loop* 2 iter. (g) (h) (i) cuchara - modelo base, *Loop* 1 iter. y *Loop* 2 iter.

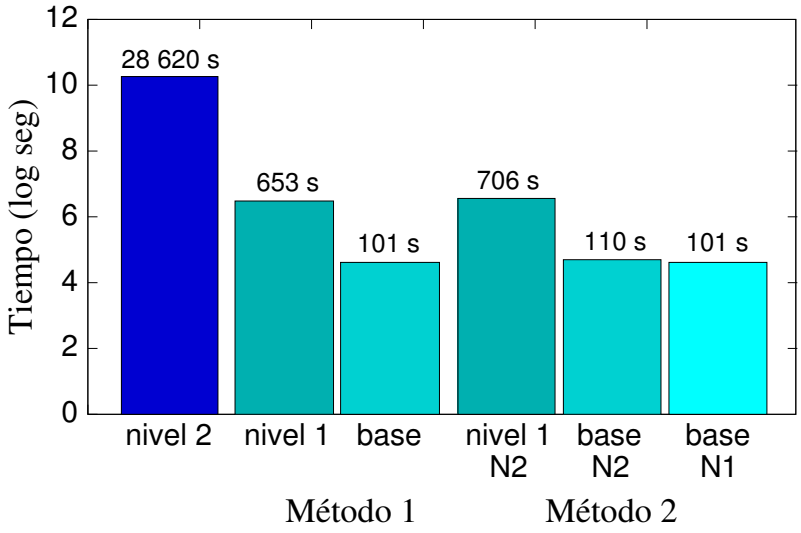
Tabla 4.1: Tiempo de ejecución total (en segundos) para el cálculo de la radiosidad con los métodos multiresolución propuestos.

Nivel dos	Nivel uno	Nivel base		
28.620 s.	653 s.	101 s.		Método 1
	706 s.	110 s.	101 s.	Método 2
		Normales nivel dos (N_2)	Normales nivel uno (N_1)	

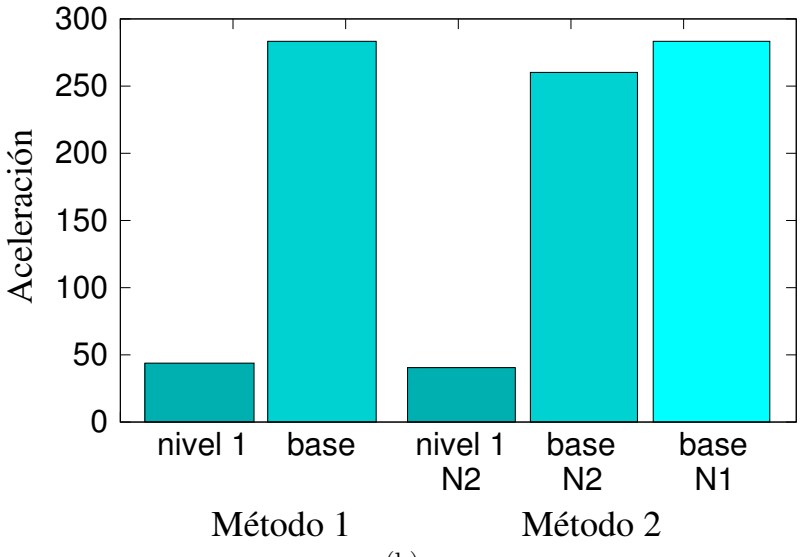
Aunque en todo momento los resultados mostrados corresponden a la aplicación del esquema de subdivisión de *Loop*, estos son totalmente análogos a los que se obtienen empleando el esquema *Modified Butterfly*.

En la Tabla 4.1 se muestra el tiempo de ejecución necesario para el cálculo total de la iluminación de la escena utilizando los métodos multiresolución basados en subdivisión de superficies (*método 1*) y agrupamiento de normales (*método 2*). El tiempo de ejecución de referencia, que corresponde a la escena de nivel dos, es de 28.620 segundos, casi 300 veces más que el mejor tiempo conseguido con la aplicación de nuestros métodos. Para el segundo de nuestros métodos, agrupamiento de normales, se han utilizado dos versiones de la escena de nivel base con normales, etiquetadas con N_i en la tabla, donde i indica el nivel de refinado máximo para obtener el conjunto de normales. En ambos casos el número de polígonos es el mismo (el correspondiente a la escena de nivel base), pero en un caso (N_1) se utilizan, junto con los polígonos gruesos, los vectores normales que resultan de aplicar un único nivel de subdivisión (4 normales por polígono), mientras que el otro (N_2) corresponde al uso de normales de dos niveles de subdivisión (lo que nos da 16 normales por polígono).

En la Figura 4.13 se muestran los resultados de tiempo en forma de aceleración (Figura 4.13b), relativa al tiempo de ejecución de la escena de referencia, junto con la representación logarítmica de los tiempos que veíamos en la Tabla 4.1 (Figura 4.13a). Teniendo en cuenta que cada uno de los dos niveles de complejidad de la escena de prueba contienen prácticamente cuatro veces más polígonos que el nivel anterior, en la gráfica podemos apreciar que la aceleración lograda con la aplicación de técnicas multiresolución en el cálculo de la radiosidad de una escena con objetos complejos supera ampliamente la disminución en complejidad geométrica correspondiente.



(a)



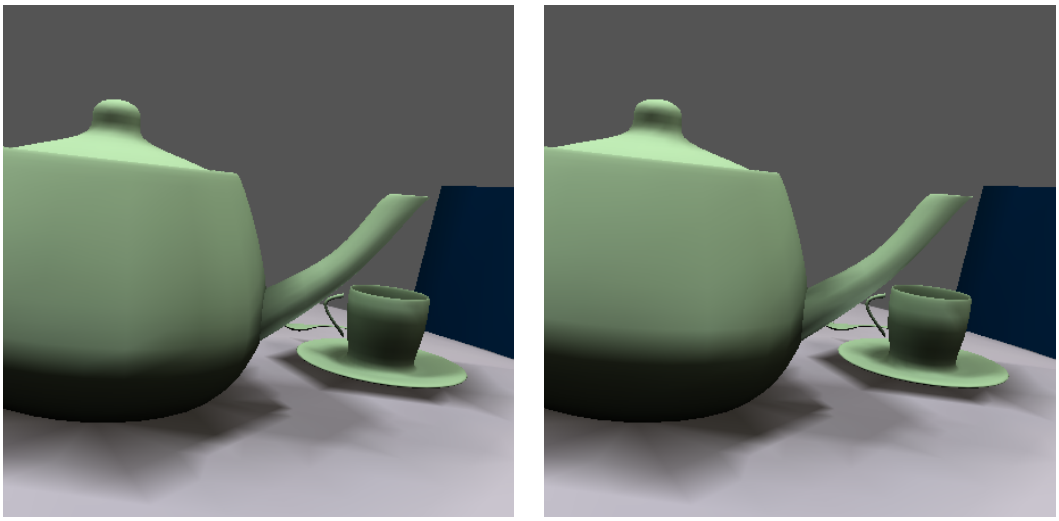
(b)

Figura 4.13: Aceleración lograda con los métodos de radiosidad multiresolución: (a) tiempo de ejecución (b) aceleración.

Tabla 4.2: Medición del error cometido en la escena final: relación señal a ruido (SNR_{ms}).

	Nivel uno	Nivel base	
Método 1	51.5	43.6	
Método 2	52.1	44.9	44.7
		Normales nivel dos (N_2)	Normales nivel uno (N_1)

Respecto a la medida del error, se ha utilizado la misma métrica descrita en el Capítulo 2 para la comparación de dos imágenes diferentes: la relación media señal-a-ruido media (*mean-square signal-to-noise ratio*, SNR_{ms}) [101]. Como se puede apreciar, la aplicación de los dos métodos multiresolución rebaja considerablemente el tiempo de ejecución necesario, manteniendo unos niveles de calidad más que aceptables en el peor de los casos, que supera ampliamente el valor de 40 de SNR_{ms}. Vemos un ejemplo del resultado logrado en la Figura 4.14.



(a) (b)

Figura 4.14: Ejemplo del *rendering* logrado con la escena de referencia (a) y con la aplicación del método con agrupamiento de normales sobre la escena con complejidad intermedia (b).

Capítulo 5

Radiosidad jerárquica paralela

El uso de técnicas de *clustering* y de aceleración de la determinación de visibilidad permite, como hemos visto hasta ahora, reducir drásticamente la complejidad del método clásico de radiosidad jerárquica. Sin embargo, el coste computacional y los requisitos de memoria necesarios para el procesamiento de la iluminación global en escenas grandes y complejas mediante este método se mantienen, todavía, demasiado elevados como para, en muchas ocasiones, ser resuelto en máquinas monoprocesador convencionales. La búsqueda de una solución paralela, que permita el aprovechamiento de sistemas multiprocesador, constituye una aproximación alternativa para la resolución del método de radiosidad jerárquica.

En una primera aproximación, la paralelización del método clásico de radiosidad puede ser abordada como la resolución común de un sistema de ecuaciones lineales, simplemente añadiendo como factores especiales a considerar las características intrínsecas al dominio geométrico y de iluminación global en el que nos encontramos, como las relaciones de visibilidad entre los diferentes objetos de una escena y la coherencia en la transmisión de la luz a través del medio. El principal problema surge en entornos de memoria distribuida (que son los más interesantes desde el punto de vista de la escalabilidad y portabilidad), ya que la necesidad de resolver la relación de visibilidad entre los distintos polígonos para el cálculo de los factores de forma obliga, bien a mantener almacenada la geometría total de la escena en cada memoria local, lo que limita las posibilidades de escalado de la paralelización a escenas grandes y complejas, o bien a distribuir la escena aumentando y complicando la comunicación entre los procesadores.

Además, por otro lado, el carácter irregular del refinado adaptativo que propone

el método de radiosidad jerárquica añade, todavía, una mayor complejidad a la tarea de alcanzar una solución paralela eficiente en entornos de memoria distribuida. Así, el progresivo refinado que se aplica sobre la escena gruesa inicial conlleva, tanto una mayor dificultad a la hora de mantener el equilibrio de la carga entre los distintos procesadores, como la necesidad continua de calcular nuevos factores de forma para las interacciones entre los nuevos y más finos elementos obtenidos, lo que aumenta la necesidad de comunicación entre los distintos procesadores y complica la concurrencia entre ellos cuando la escena ha sido distribuida entre las memorias de los procesadores.

En este capítulo abordamos la paralelización del método de radiosidad jerárquica utilizando un paradigma de paralelismo basado en memoria distribuida. Uno de los objetivos de nuestra propuesta consiste en el desarrollo de un método que sea lo más general posible, válido para procesar distintos tipos de escenas de entrada, y que además sea también fácilmente trasladable a cualquier tipo de arquitectura multiprocesador, ya sea con memoria compartida o distribuida. Es en este último tipo de sistemas, sin embargo, en el que se ha centrado nuestro trabajo, ya que permiten soluciones más generales y escalables, en las que prime la portabilidad. Con este fin se ha optado por una aproximación SPMD, con un esquema de paso de mensajes para las comunicaciones entre procesadores, lo que también garantiza la generalidad de la aproximación.

Otro aspecto en el que nuestro diseño ha hecho hincapié es en el mantenimiento de un control de la ejecución paralela totalmente distribuido, evitando así la necesidad de mantener un procesador central a cargo del gobierno del proceso de iluminación o de la asignación de procesadores específicos a tareas concretas, como pudiera ser el cálculo de la visibilidad.

De acuerdo a los criterios y objetivos expuestos, se han desarrollado dos implementaciones paralelas diferentes para la resolución del método de radiosidad jerárquica, que son descritas en este capítulo. En la primera de ellas se lleva a cabo un balanceo dinámico de la carga computacional entre los procesadores, aunque con el inconveniente de mantener una replicación de la geometría inicial de la escena a procesar en la memoria de todos los procesadores. Nuestra segunda implementación, por su parte, carece de esta limitación, al basarse en la completa distribución de la escena de entrada entre las diferentes memorias de los procesadores del sistema, lo que permitirá el trabajo con escenas de mayor tamaño. Este segundo método, además, hace uso de un sistema de ejecución multi-hilo (*multi-thread*), que permite a cada procesador un control más fino de la granularidad con la que se encarga del

cálculo de la iluminación para su subescena local, a la vez que, de forma concurrente, atiende consultas de visibilidad remotas relativas a su subespacio dentro de la escena total.

En ambas implementaciones se ha utilizado una aproximación de grano grueso para la resolución en paralelo del método de radiosidad sobre una escena de entrada. Así, el reparto del trabajo entre los procesadores se hace a nivel de triángulo de entrada, previa aplicación de un proceso de triangulización[1] sobre la escena inicial, de manera que cada procesador se va a encargar por completo del cálculo de la radiosidad para el conjunto de polígonos asignados.

En la primera sección del capítulo se realiza una breve revisión de los principales trabajos existentes en la paralelización del método de radiosidad, destacando los puntos fuertes y débiles de las distintas aproximaciones. A continuación se introducen las dos implementaciones propuestas en este trabajo, enumerando los aspectos que se han considerado más importantes a la hora de diseñar nuestras propuestas. En ambos casos se describen las partes más relevantes del funcionamiento del método y se exponen los resultados más significativos obtenidos tras las distintas pruebas experimentales aplicadas sobre las implementaciones realizadas.

5.1. Trabajo previo en radiosidad paralela

Los primeros trabajos sobre algoritmos paralelos de radiosidad se basan en los métodos de resolución de sistemas de ecuaciones lineales mediante procesos iterativos clásicos (Jacobi, Gauss-Seidel...) [24, 109]. Pronto aparecen, sin embargo, las primeras implementaciones paralelas de las soluciones específicas para la resolución del método de radiosidad. El método de radiosidad progresiva fue inicialmente el más utilizado, ya que su funcionamiento permite una paralelización relativamente sencilla del mismo. Así, por un lado aparecen implementaciones en las que la escena completa a procesar se encuentra totalmente accesible al conjunto de procesadores, lo que simplifica notablemente la paralelización, bien porque se trata de sistemas de memoria compartida [23, 113], o bien porque la escena completa se almacena repetida en las memorias de los distintos procesadores en sistemas con memoria distribuida [23, 25, 135].

Salvo excepciones [135], se trata, en su mayoría, de implementaciones paralelas

[1] Proceso que habitualmente recibe el nombre de *tessellation*

de grano grueso en las que el objetivo prioritario consiste en reducir en lo posible el tiempo de ejecución mediante el reparto de la carga de trabajo entre los distintos procesadores, sin consideraciones adicionales que compliquen la paralelización, realizando para ello el mínimo número de comunicaciones posible entre los procesadores. El principal inconveniente de esta aproximación, además de los problemas de escalabilidad y del control de acceso a secciones críticas en el caso de las implementaciones sobre sistemas de memoria compartida, inherentes a este tipo de plataformas, reside en la limitación del tamaño de las escenas a procesar. En estas implementaciones se realiza un reparto de los polígonos de la escena inicial, utilizando para ello alguna métrica determinada, de manera que cada procesador actúa únicamente sobre un subconjunto de los polígonos de la escena, sin embargo, la geometría completa de la escena de entrada reside en todas las memorias de los distintos procesadores. Se evitan, de esta forma, las comunicaciones derivadas de la determinación de visibilidad entre subescenas, pero coartando con ello, sin embargo, la posibilidad de trabajar con escenas de entrada excesivamente grandes, que sobrepasen la capacidad de almacenamiento de la memoria local de una sola de las máquinas del sistema.

Trabajos posteriores optan por implementaciones totalmente distribuidas, repartiendo no solo los cálculos a realizar sobre los polígonos, sino también la geometría de la propia escena, cuya partición geométrica se distribuye entre los distintos procesadores. Aunque en alguna de las soluciones de este tipo la ejecución del método se dirige de forma centralizada desde un nodo [39], con una aproximación de grano fino, la mayoría de los trabajos optan, de nuevo, por una solución de grano grueso, obteniendo además mejores resultados [9, 50, 61, 62]. La idea de distribuir la partición creada sobre la escena entre los procesadores presenta, a priori, el inconveniente de un aumento significativo en las necesidades de comunicación entre los distintos procesadores. Este incremento en las comunicaciones se produce como consecuencia de la necesidad de determinar el porcentaje de visibilidad entre objetos en subespacios almacenados en procesadores diferentes, lo que conlleva la necesidad de acceder a información geométrica no almacenada localmente. La reducción en lo posible del número de comunicaciones entre procesadores pasa, en estas aproximaciones, habitualmente por la simplificación o aproximación de la visibilidad entre subescenas.

Entre las técnicas que más se han popularizado para la resolución del problema que supone la transmisión de la información de visibilidad entre los distintos procesadores, se encuentran las basadas en lo que se conocen como interfaces virtuales (*Virtual Interfaces*) y máscaras de visibilidad (*Visibility Masks*) [9]. Las máscaras de

visibilidad suponen una simplificación de las relaciones de visibilidad entre polígonos en diferentes subescenas, permitiendo que la partición y distribución de la geometría entre las diferentes memorias en un sistema distribuido no suponga un aumento excesivo en las comunicaciones a realizar entre los procesadores. Se trata, por tanto, de utilizar fronteras artificiales de polígonos virtuales entre las diferentes subescenas para acumular y transmitir la información relativa a la relación de visibilidad entre un polígono local y los polígonos de los subespacios vecinos. El error introducido con esta simplificación puede ser, sin embargo, difícil de cuantificar y de acotar.

Otros trabajos basados en la técnica de las máscaras de visibilidad para la transmisión de la información de visibilidad en un entorno de radiosidad progresiva paralela, en este caso desarrollados por nuestro grupo de investigación, son, por un lado, [5] y [6], donde se presenta un nuevo método para el recorte (*clipping*) de polígonos, aplicado a la partición y distribución de la geometría de la escena mediante una rejilla uniforme; y, por otro lado, [118], donde se logra mejorar el equilibrio entre la carga computacional de los distintos procesadores mediante la aplicación de una partición no uniforme del espacio.

Otra técnica para evitar el excesivo número de comunicaciones que supone la determinación de la visibilidad entre polígonos en subescenas asignadas a distintos procesadores es la utilizada por Gibson y Hubbold [50]. En este caso se aplica una simplificación todavía más agresiva del cálculo de la visibilidad entre subescenas, al utilizar una malla de *voxeles* con información de visibilidad muy básica: siete direcciones por *voxel* con información binaria de si hay o no un obstáculo en esa dirección dentro del *voxel*. La malla es precalculada y mantenida en cada procesador, y se utiliza para aproximar la visibilidad entre procesadores sin necesidad de comunicación alguna como parte de un método de radiosidad progresiva paralelo dirigido por percepción (*perceptually-driven*). Se parte de la suposición de que la interacción entre objetos en diferentes subescenas tendrá una incidencia relativamente pequeña, o incluso muy pequeña, en la iluminación global resultante, haciendo innecesario un gasto de tiempo excesivo en la determinación exacta de la visibilidad para esa interacción.

En cuanto al método de radiosidad jerárquico, la incorporación del refinado adaptativo de la escena durante el cálculo de la iluminación complica el diseño de una aproximación paralela, al acentuar los problemas ya existentes de por sí en la aproximación progresiva: dificultad para equilibrar la carga computacional entre los procesadores y altas necesidades de comunicación si los datos de la escena se reparten en un sistema distribuido. Los primeros resultados reseñables se obtuvieron en arqui-

tecturas de memoria compartida [130], donde todos los procesadores tienen acceso a la escena común completa, lográndose muy buenos resultados en trabajos basados en la segmentación del problema en colas de tareas de granularidad muy fina, que permiten un buen balanceo en el reparto de la carga [126]. Los resultados obtenidos en trabajos sobre sistemas con memoria distribuida son, sin embargo, más pobres, debido principalmente a la sobrecarga que supone el aumento de las comunicaciones necesarias, así como a la mayor complejidad de su diseño, como se ha descrito en párrafos anteriores.

Zareski [146] desarrolló una implementación paralela del algoritmo de radiosidad jerárquica sobre una red de estaciones de trabajo, aplicando un paralelismo de grano fino con un paradigma *master-slave*, donde los esclavos ejecutan los cálculos de intersección rayo-polígono necesarios para el cálculo de los factores de forma, cada uno sobre un subconjunto de polígonos de la escena.

Otra aproximación *master-slave* de grano fino se describe en [40], trabajando con celdas 3D y conjuntos de polígonos visibles desde esas celdas (*CV-set*). En cada paso del algoritmo, un *CV-set* completo se almacena en la memoria de cada procesador, repartiendo el maestro trozos de la celda asociada entre los procesadores esclavos para su cálculo de la radiosidad. Este tipo de aproximaciones suponen, sin embargo, un número de comunicaciones excesivamente elevado en entornos de memoria distribuida, por lo que los trabajos más recientes tienden a centrarse en un paralelismo de grano grueso, donde cada procesador se va a encargar de calcular la radiosidad sobre su subescena correspondiente.

En [13] y [45] se mantiene la aproximación *master-slave*, con un esquema de planificación dinámico centralizado en el que un procesador maestro es el encargado del análisis de la convergencia y del reparto dinámico de los polígonos de la escena entre los procesadores esclavos, que se encargan de todo el cálculo de la radiosidad jerárquica sobre la parte asignada. Mientras que en la aproximación de Funkhouser [45] la escena es repartida entre los procesadores, lo que permite trabajar con escenas de mayor tamaño a costa de aumentar y complicar las comunicaciones entre los procesadores, en el otro trabajo se opta por mantener la escena completa almacenada en la memoria de cada procesador, minimizando las comunicaciones pero con los problemas ya comentados para estos casos a la hora de trabajar con escenas de gran tamaño. La diferencia entre ambas aproximaciones se centra en la granularidad del balanceo dinámico de la carga, que en el primer caso se hace a nivel de polígonos y en el segundo a nivel de interacciones. Una aproximación parecida, pero en la que se elimina la necesidad de dedicar un procesador a tareas de planificación se describe

en [4], [106] y [107], con la introducción de un algoritmo de planificación dinámico distribuido para el balanceo de la carga.

Otra paralelización de grano grueso que no usa una aproximación *master-slave* se presenta en [94], aprovechando en este caso la memoria secundaria como un espacio común entre los procesadores para simplificar la implementación y reducir además el número de comunicaciones necesarias. Los autores confían al sistema de ficheros NFS (*Network File System*) la labor de acceder concurrentemente de forma óptima a los datos, lo que supone el mayor cuello de botella de esta propuesta.

Por último, comentar algún trabajo centrado en la paralelización de métodos de Monte Carlo aplicados al cálculo de la radiosidad, última tendencia en iluminación global pero que apenas cuenta con intentos de paralelización todavía. En [91] se presenta una paralelización para sistemas de memoria compartida del método de Monte Carlo Multi-camino, mientras que en [92] se propone un modelo analítico para la implementación del método de Iteración Estocástica, demostrando que es posible independizar, prácticamente por completo, el cálculo llevado a cabo en cada uno de los procesadores, minimizando el número de comunicaciones. El tema queda abierto, sin embargo, al no mostrarse en el trabajo un excesivo número de pruebas ni alcanzar conclusiones definitivas al respecto.

5.2. Método paralelo con replicación de la escena

Nuestra primera aproximación a la radiosidad jerárquica paralela [4, 106, 107] se basa en la posibilidad de acceso a la escena inicial completa en cada procesador, lo que implica la existencia de una copia de la escena gruesa de entrada en la memoria de cada procesador en el caso de sistemas de memoria distribuida.

Los polígonos de entrada son inicialmente asignados a los procesadores, intentando equilibrar en lo posible el coste computacional estimado. Para ello, se distribuye cíclicamente una lista de polígonos ordenada de acuerdo a algún criterio que nos permita, de alguna manera, estimar el tiempo de cálculo probable de cada polígono (ver Figura 5.1a). Así, si se dispone de p procesadores, para un total de n triángulos de entrada, a un procesador i le correspondería inicialmente el siguiente conjunto de triángulos a procesar:

$$T(i) = \{t \ mod \ p = i, \ 1 \leq t \leq n\}$$

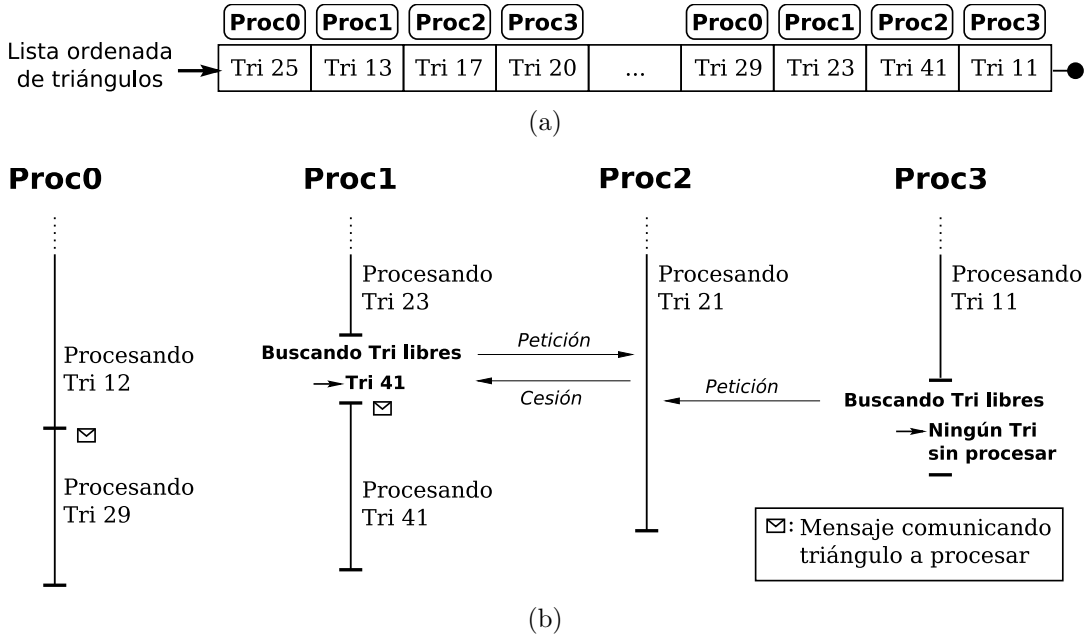


Figura 5.1: Esquema para equilibrar dinámicamente la carga de trabajo entre los distintos procesadores: (a) Asignación estática (b) Balanceo dinámico.

donde t es la posición de un triángulo en la lista ordenada y $T(i)$ la lista de los triángulos asignados estáticamente el procesador i.

Para nuestras pruebas empleamos el área del polígono y el número de interacciones con el resto de los polígonos de la escena como medidas con las que intentar predecir el coste del cálculo de la radiosidad jerárquica para un determinado polígono, lo que nos va a permitir mantener una lista ordenada de triángulos a procesar en función de su coste estimado. En cualquier caso, la naturaleza irregular e impredecible del método de radiosidad jerárquica dificulta, en la mayoría de los casos, un reparto estático de la carga equilibrado, lo que nos lleva a nuestra propuesta para un esquema dinámico de balanceo del trabajo asignado a cada procesador.

Una solución frecuente al problema de la obtención de un reparto de la carga equitativo entre los procesadores, sería la de tratar de equilibrar la carga tras una primera iteración con la asignación estática. Esta aproximación, sin embargo, no es adecuada en nuestro caso, al ser habitual que la primera iteración del algoritmo consuma una gran parte del tiempo total de cálculo de la radiosidad de la escena. Es necesario, por lo tanto, un esquema de balanceo dinámico de la carga que se aplique desde el inicio del proceso.

Para equilibrar la carga entre los procesadores, nuestro esquema aplica una reasignación de los triángulos a procesar, a medida que los procesadores terminan con el trabajo preasignado estáticamente; todo ello de forma totalmente distribuida, sin la necesidad de establecer un procesador maestro o principal que se encargue de repartir el trabajo (ver Figura 5.1).

La granularidad del balanceo dinámico de la carga es configurable, con un mínimo de un triángulo de entrada, y en todo momento cada procesador mantienen información sobre el número de triángulos disponibles en el resto de los procesadores. En la Figura 5.1b se muestra de forma esquemática cómo actuaría el esquema de balanceo dinámico de la carga. En este ejemplo, se reparte cíclicamente un conjunto de triángulos de entrada, previa ordenación de los mismos, entre cuatro procesadores. Los procesadores se mantienen informados de sus progresos mediante el envío de mensajes cuando se disponen a procesar un grupo de sus triángulos asignados (en el caso del ejemplo, con una granularidad de un triángulo). Cuando los procesadores uno y tres terminan, en ese orden, con su subescena, el procesador cero se encuentra trabajando sobre su último triángulo y sólo el procesador dos cuenta todavía con algún triángulo disponible sin procesar. Ambos procesadores envían una petición solicitando el traspaso de algún triángulo desde ese procesador. Al procesador uno, cuya petición llega en primer lugar, le es adjudicado así el último triángulo todavía sin procesar de la escena. La petición del procesador tres es desestimada, con lo que la iteración concluye una vez procesados los triángulos en curso.

Durante cada iteración, la ejecución del método transcurre de forma independiente y asíncrona en cada uno de los procesadores, de forma que cada uno de ellos aplica el método de radiosidad jerárquica sobre el conjunto de polígonos asignados. Esta independencia en la ejecución plantea una dificultad a la hora de gestionar el intercambio de mensajes entre los procesadores destinados a equilibrar la carga, al no disponer de puntos o barreras de sincronización para los mismos. Se trata de mensajes muy cortos, en los que únicamente se transmiten peticiones e información de identificación de los polígonos a reasignar. No es necesario transmitir ninguna información geométrica ni de visibilidad, al disponer cada procesador de acceso completo a la escena gruesa inicial. Además, los mensajes de comunicación entre los distintos procesadores, necesarios para la implementación del esquema de balanceo dinámico de la carga, se llevan a cabo mediante operaciones no bloqueantes, solapando en la medida de lo posible comunicación y cálculo. Para la atención de las peticiones de triángulos sin procesar que los procesadores se envían entre sí durante esta fase se utiliza un sistema de consulta (*polling*), estableciendo una serie de puntos fijos, y en

cierto modo arbitrarios, de comprobación o escucha durante el cálculo de la radiosidad de la subescena local. De esta manera, se realizan comprobaciones periódicas de la posible recepción de mensajes provenientes del resto de procesadores, evitando el uso de barreras de sincronización que entorpezcan la ejecución independiente de cada uno de los procesadores. Un inconveniente de esta aproximación radica en la dificultad de encontrar una granularidad adecuada con la que realizar las operaciones de consulta para alcanzar el compromiso entre el coste que supone la realización de consultas innecesarias y el tiempo que se mantiene a los procesadores esperando por la resolución de estas consultas. En cualquier caso, la espera siempre se va a producir por parte de un procesador que ya ha completado su subescena asignada en esa iteración, con lo que la aplicación del esquema dinámico de balanceo apenas grava la ejecución puramente estática, salvo por el coste en la realización de las consultas que no eran necesarias.

En cuanto al cálculo de la radiosidad propiamente dicho, éste se lleva a cabo, como ya se ha comentado, con una granularidad gruesa a nivel de polígono, de forma que al finalizar cada iteración los procesadores realizan una fase de comunicación en la que actualizan las radiosidades de los elementos que cada uno ha procesado. Para ello, cada procesador envía una petición al resto, indicándoles cuáles de sus elementos son necesarios para el cálculo de la radiosidad sobre su lista de polígonos local, elementos de los cuales, por lo tanto, necesita actualizar el valor de radiosidad. Esta fase de sincronización se completa con un intercambio de las radiosidades solicitadas entre los procesadores. Se trata, pues, de una fase de comunicación todos-a-todos (*all-to-all scatter/gather*), para la implementación de la cual se han empleado las funciones colectivas de MPI, *MPI_Alltoall* y *MPI_Alltoallv*.

En este proceso de sincronización de las radiosidades entre los distintos procesadores se encuentran implícitas, además, las actualizaciones necesarias de las estructuras jerárquicas de elementos que resultan en los diferentes procesadores. Así, cuando un procesador refina un polígono que no se encuentra entre su conjunto de polígonos asignados, la propia petición de los valores de radiosidad de los elementos más finos provocará que el procesador a cargo de este polígono lo refine hasta el nivel necesario.

Otro detalle a resaltar es el uso de comunicaciones persistentes, al tener los mensajes intercambiados un patrón y unos destinatarios fijos. Se hace uso, por lo tanto, de las llamadas MPI *MPI_Send_init*, *MPI_Recv_init*, *MPI_Start* y *MPI_Request_free*, con lo que se evita realizar múltiples veces una serie de tareas redundantes asociadas al establecimiento de una comunicación.

A continuación mostramos los resultados experimentales más significativos obtenidos con el método planteado. Una descripción más exhaustiva de la implementación realizada, así como una exposición completa de los resultados obtenidos, puede consultarse en [4, 107, 106]. En el resto de este capítulo vamos a centrarnos en nuestra segunda aproximación a la radiosidad jerárquica paralela.

5.2.1. Resultados experimentales

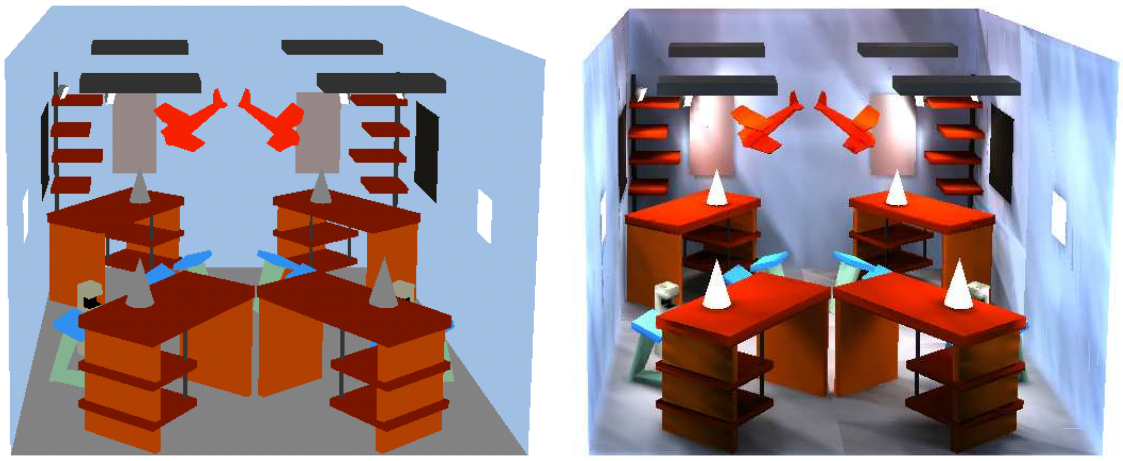
La implementación de la paralelización propuesta en esta sección se ha desarrollado íntegramente en el lenguaje de programación C, utilizando la librería de paso de mensajes *MPI* (concretamente la especificación *MPI-1.2*) para la comunicación entre los procesadores.

Las plataformas de prueba utilizadas para obtener los resultados que se proporcionan en esta sección han sido las siguientes:

- Un Fujitsu AP3000, computador paralelo MIMD con memoria distribuida y procesadores UltraSparc-II a 300 MHz, conectados mediante una red AP-NET (*Advanced Parallel Systems Network*), que proporciona un ancho de banda de 200 MBytes/s, con una topología de toro 2D (*ap3000*).
- Un Cray T3E, sistema multiprocesador con memoria distribuida con procesadores Alpha 21164 a 300 MHz con una topología de interconexión en toro 3D que proporciona un ancho de banda de 480 MBytes/s (*cray*).

La escena *Habitación*, utilizada para las pruebas que aquí presentamos, se muestra en la Figura 5.2. Se trata de una variación de la habitación clásica de Hanrahan [64], a la que se le han añadido varios objetos y se le ha aplicado un efecto espejo. El número de triángulos inicial de la escena es de 1.292 (Figura 5.2a), obteniéndose 10.500 triángulos tras la aplicación del método de radiosidad jerárquica (Figura 5.2b).

En la Figura 5.3 aparecen las gráficas con la aceleración (*speed-up*) alcanzada en la ejecución paralela de la escena *Habitación*, con hasta 12 procesadores para el *ap3000* (Figura 5.3a) y 30 procesadores en el caso del *cray* (Figura 5.3b). En ambas gráficas se muestra el rendimiento alcanzado con una distribución estática de la carga y con el esquema de balanceo dinámico implementado. Los tiempos de ejecución del algoritmo secuencial son de 2.309 segundos en el *ap3000* y de 1.293 segundos en el



(a) (b)

Figura 5.2: Escena de prueba utilizada: *Habitación* (a) escena de entrada (b) escena iluminada.

cray, quedando con la ejecución paralela en el *ap3000* para 12 procesadores en 222 segundos, el caso estático, y 198 segundos, con el balanceo dinámico; y en el *cray*, para 30 procesadores, en 57 y 48 segundos, respectivamente.

Como se desprende de las gráficas, el esquema de balanceo dinámico de la carga implementado consigue equilibrar la carga computacional entre los procesadores, reduciendo los tiempos de espera producidos por la diferencia en la carga inicial de los procesadores, lo que mejora sustancialmente la aceleración lograda y la acerca al caso lineal ideal. Como muestra del balanceo de la carga alcanzado con el esquema dinámico, en la Figura 5.4 se muestra el tiempo de ejecución que la primera iteración del método de radiosidad consume en cada uno de los procesadores del *ap3000*. A la derecha se representan los tiempos de una ejecución con la distribución estática inicial de la carga, mientras que a la izquierda aparecen los tiempos que se logran cuando el algoritmo de balanceo dinámico de la carga entra en funcionamiento.

5.3. Método paralelo con distribución de la escena

El segundo método que planteamos en este trabajo trata de superar una de las principales limitaciones de nuestra aproximación anterior: la necesidad de mantener

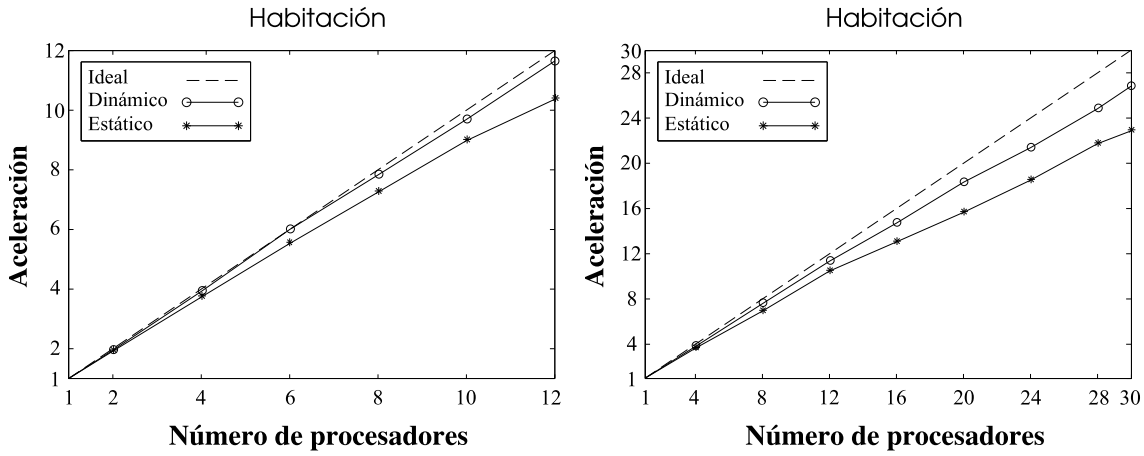


Figura 5.3: Aceleración para la escena *Habitación* (a) sobre el AP3000 (b) sobre el Cray T3E.

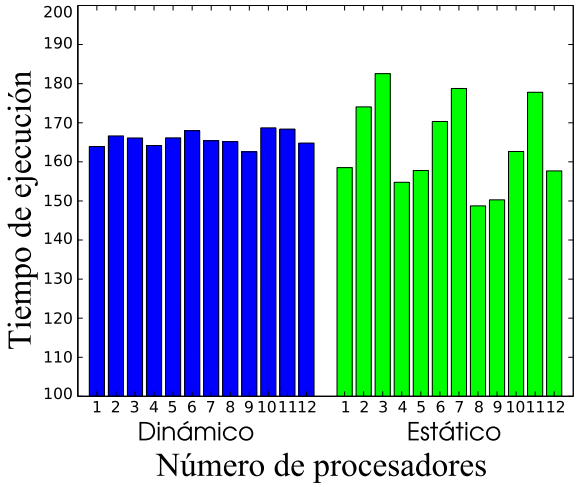


Figura 5.4: Equilibrio de la carga: tiempo de ejecución (en segundos) de la primera iteración en cada uno de los procesadores del AP3000.

la escena de entrada completa y permanentemente almacenada en la memoria de cada procesador. Con este fin, pasamos del simple reparto de los cálculos a realizar sobre la escena de entrada del método anterior, a la aplicación de una distribución real de los datos de la escena entre las memorias de los distintos procesadores. Esta partición, llamémosle *física*, de los datos complica en gran medida el diseño de una solución paralela en un ámbito de modelos de iluminación global como en el que nos encontramos. La correcta simulación de la distribución de la luz en un entorno mediante el modelo de radiosidad hace necesario considerar la interacción entre todos los objetos relevantes de la escena, lo que implica la necesidad de garantizar el acceso de un procesador no sólo a la subescena asignada (polígonos locales), sino también a polígonos ajenos a la misma (polígonos remotos). Este hecho obliga, respecto a la solución anterior en la que se mantenían réplicas completas de la escena, a un aumento en el número y en la complejidad de las comunicaciones necesarias.

La distribución *física* de los datos de la escena entre los procesadores supone, además, la aplicación de una partición geométrica sobre la escena, en contraste con la paralelización anterior. Una de las consecuencias más inmediatas de la aplicación de este tipo de partición es el aprovechamiento de la coherencia geométrica de la escena y del principio de localidad de los datos, respecto a la distribución cíclica de los mismos realizada en el caso anterior. Además, con el objetivo de reducir al máximo el número de mensajes necesarios referidos a consultas de visibilidad entre procesadores, los subespacios que resultan de la partición de la escena son geométricamente convexos, esto es, un segmento que une dos puntos del subespacio se encuentra también completamente dentro de ese subespacio. De esta forma se garantiza que la relación de visibilidad entre objetos de un mismo subentorno puede obtenerse sin necesidad de comunicación alguna, haciendo uso únicamente de los datos presentes en el procesador asignado.

Frente a la replicación de la escena en todos los procesadores, la mayor implicación derivada del mantenimiento distribuido de los datos de la escena consiste en la necesidad de una mayor comunicación entre los procesadores, al tener estos que intercambiarse consultas de visibilidad referentes al subespacio de la escena del que disponen. En nuestro trabajo proponemos, para ello, un nuevo método para la determinación distribuida de la visibilidad mediante la transmisión de mapas de bits.

Nuestra aproximación trata, por lo demás, de lograr que los diferentes procesadores trabajen de la forma más desacoplada e independiente posible, evitando establecer un número excesivamente alto de puntos de sincronización que introduz-

can tiempos de espera en el sistema. Para que cada procesador sea capaz de atender en el menor tiempo posible las consultas de visibilidad recibidas, se ha optado por una aproximación multi-hilo, frente al establecimiento de los puntos fijos de consulta utilizados en nuestra solución anterior.

En la Figura 5.5 se muestra la estructura general del método, mediante un diagrama de flujo que representa el comportamiento de cada procesador del sistema. Por un lado, y al igual que el método paralelo anterior, el diseño consiste en una aproximación SPMD (*single-program, multiple-data*), donde todos los procesos ejecutan un mismo código, con un único proceso por procesador, y cuyo comportamiento se personaliza básicamente mediante bloques condicionales. Por otro lado, sin embargo, cada proceso va a constar de múltiples hilos de ejecución simultáneos, por lo que se basa en una implementación multi-hilo (*multi-thread*). El concepto de programación multi-hilo no se diferencia, en esencia, de la programación multi-proceso clásica. Se trata de disponer simultáneamente de varias tareas en ejecución, con una determinada granularidad, que se van repartiendo el tiempo y los recursos del sistema. Un hilo o *thread* es lo que se conoce como *proceso ligero* (*light-weighted process*): varios hilos comparten el contexto y los recursos de un mismo proceso, permitiendo simultanear tareas sin la necesidad de costosos cambios de contexto.

En la Figura 5.5 se muestra la estructura de nuestro método con distribución de la escena. En nuestra aproximación, se lanza un proceso por cada procesador del sistema con, inicialmente, un único hilo de ejecución, al que denominaremos hilo principal. El hilo principal (flujo de la izquierda en la Figura 5.5) se va a encargar, primero, de todo el preproceso necesario para aplicar el método de radiosidad (fase inicial en el diagrama de flujo de la Figura 5.5), lo que incluye tareas como la partición de la escena y la construcción de las estructuras de aceleración necesarias para el posterior cálculo de la visibilidad (según lo visto en el Capítulo 2). A continuación, el hilo principal pasará a dedicarse exclusivamente del cálculo de la iluminación sobre la subescena local, aplicando el proceso iterativo habitual para el método de radiosidad jerárquica (visto en el Capítulo 1). De forma simultánea, en cada procesador se va a lanzar un segundo hilo, que se va a ejecutar de forma concurrente con el hilo principal (flujo de la derecha en la Figura 5.5). Este hilo auxiliar se va a encargar de dar servicio al resto de procesadores, de los cuales recibe distintas peticiones relativas a su subescena, de forma completamente desacoplada e independiente del funcionamiento del hilo principal. Los mensajes a los que deberá dar respuesta el hilo auxiliar son básicamente consultas de visibilidad relativas a interacciones que traviesan el subespacio local asignado al procesador. La idea

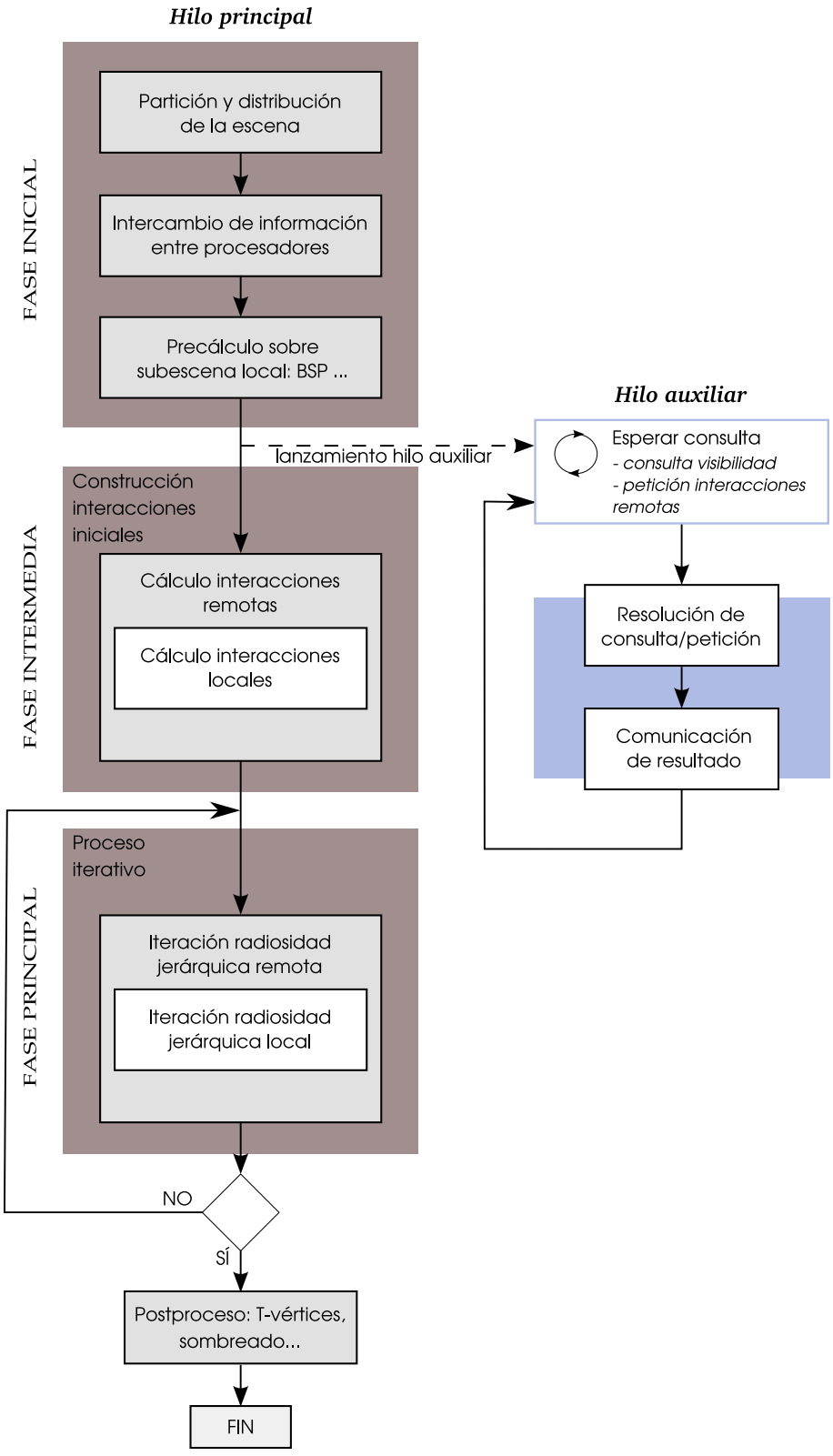


Figura 5.5: Estructura del método paralelo con distribución de la escena.

es evitar las esperas en la atención de las consultas y peticiones remotas propias de un esquema de encuesta, que es el que habitualmente se emplea en estos casos. En nuestra implementación, el hilo auxiliar será el encargado de, secuencialmente, atender las peticiones de visibilidad recibidas (como se muestra en el diagrama de la Figura 5.5). Una posible ampliación sería la implementación de una versión totalmente multi-hilo, en la que el hilo auxiliar se encargaría únicamente de recibir las peticiones de otros procesadores, lanzando un hilo adicional independiente para cada consulta recibida; hilo que se encargaría única y exclusivamente de resolver esa consulta, aumentando así el nivel de multitarea existente.

Respecto a la ejecución del proceso de radiosidad jerárquica que se lleva a cabo en el hilo principal tras la fase de preproceso, decir que éste consta principalmente de dos partes diferenciadas (las fases intermedia y final en la Figura 5.5, respectivamente). Por un lado, el cálculo de las interacciones iniciales entre los elementos de la escena, que constituye el nivel más grueso en la propagación de la energía en la escena, como veíamos en el Capítulo 1; y por otro lado el proceso iterativo de cálculo de la radiosidad jerárquica para los elementos de la subescena local, utilizando como punto de partida las interacciones iniciales calculadas. Cada una de estas dos fases se puede separar en una etapa local y una etapa remota, que se intercalan, como vemos en la Figura 5.5, con el objetivo de solapar lo más posible computación y comunicación. En ambos casos, la etapa local simplemente consiste en la aplicación del método de radiosidad jerárquica secuencial sobre el dominio de la subescena local, de forma totalmente análoga a lo descrito en el Capítulo 1 para el método secuencial. Cada iteración remota consta, sin embargo, de varios pasos, como veremos en los siguiente apartados, de forma que primero se realiza una petición de datos de interacciones remotas y, tras un periodo de espera para recibir los datos solicitados, se distribuye la energía remota procedente de esas interacciones. Así, los cálculos sobre la subescena local se realizan de forma solapada con la espera necesaria para disponer de la información remota.

Tras cada iteración, y si no se ha especificado un número concreto (y fijo) de iteraciones a procesar, cada procesador comprueba si se ha alcanzado la convergencia en la iluminación global de la escena. Esta comprobación es realizada en paralelo por todos los procesadores, utilizando, al igual que en el método paralelo descrito en la sección 5.2, una operación de reducción global. Se considera finalizado el proceso iterativo de iluminación, y por tanto decimos que se ha alcanzado la convergencia, cuando la diferencia entre la energía total distribuida en la escena entre dos iteraciones es menor que un determinado umbral. Alcanzada la convergencia del algorirmo,

todos los procesadores dan por concluido el proceso iterativo, al considerar que se ha alcanzado el equilibrio en la distribución de la energía de la escena, de forma totalmente análoga al algoritmo secuencial.

Las siguiente subsecciones analizan en detalle los aspectos más relevantes de esta implementación paralela, como son el tipo de partición utilizado, el desarrollo paralelo del proceso iterativo y la determinación distribuida de la visibilidad entre los procesadores.

5.3.1. Partición geométrica uniforme adaptada al contenido

La primera fase del hilo de ejecución principal (fase inicial en la Figura 5.5), que al comienzo del proceso es el único hilo existente, comienza con la creación de una partición geométrica sobre la escena inicial, con la correspondiente distribución de la información geométrica entre los procesadores. Cada procesador se va a quedar, así, con un subconjunto diferente de los datos de entrada. Tras el reparto de los datos, una primera fase de comunicación sirve para que los procesadores se comuniquen el volumen de contorno de su subescena local.

El reparto de los cálculos en un sistema distribuido no implica la creación de una partición en el espacio de datos que se encuentre «físicamente» distribuida entre las diferentes memorias del sistema. De hecho, como hemos visto, nuestro primer método (Sección 5.2) y muchas de las aproximaciones comentadas en la sección introductoria de este capítulo trabajan con un espacio de datos, en nuestro contexto la escena a iluminar, totalmente accesible a todos los procesadores. En las ocasiones en las que se trata de trabajos centrados en sistemas paralelos con memoria compartida, esta aproximación está plenamente justificada, sin embargo, como hemos visto, también el grueso de paralelizaciones para sistemas de memoria distribuida existentes almacenan repetidas veces la escena completa de entrada en las diferentes memorias del sistema. Una distribución de los datos entre los procesadores supone, fundamentalmente, la ventaja de poder trabajar con escenas de una mayor complejidad y tamaño que el máximo impuesto por la memoria física de una sola de las máquinas de nuestro sistema.

El principal problema que plantea la distribución de la carga computacional entre un conjunto de procesadores puede resolverse mediante la aplicación de técnicas para la partición de grafos [120]. En la práctica, sin embargo, la aproximación más habitual consiste en recurrir a las típicas distribuciones estáticas cíclicas o por blo-

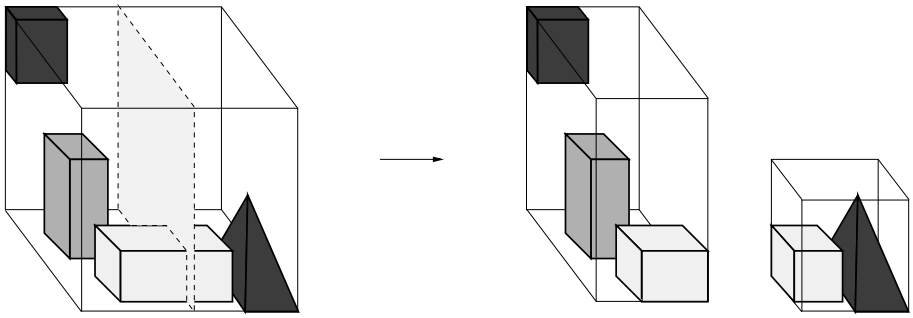


Figura 5.6: Subdivisión uniforme de un espacio (a) y posterior proceso de ajuste al contenido de los subentornos resultantes (b).

ques, con ordenación en función de algún parámetro de coste y con la posibilidad de un posterior balanceo dinámico de la carga [4, 106, 107], como hemos visto en la Sección 5.2. No obstante, la distribución de los datos entre los distintos procesadores, junto con los cálculos a efectuar sobre ellos, puede restringir la elección del tipo de partición a aplicar. En el contexto de la iluminación global de una escena en el que nos encontramos, la distribución de los datos nos obliga a optar por particiones de bloques geométricos convexos. De esta manera, se conserva la coherencia espacial dentro de cada subespacio asignado a un procesador, lo que permite que la determinación de la visibilidad entre polígonos de un mismo subespacio se realice como una operación local, lo que mantiene el número de comunicaciones entre procesadores dentro de unos límites razonables, respecto a otro tipo de particiones no geométricas. Por otro lado, las particiones geométricas en las que se utilizan planos de subdivisión alineados con los ejes son muy rápidas y sencillas, además de ofrecer muy buenos resultados [5, 118].

El diseño de nuestro método paralelo de radiosidad permite la aplicación de cualquier tipo de partición geométrica, siempre que se cumpla que el conjunto de subentornos obtenidos sean convexos y disjuntos. Además, el método no restringe el número de subespacios de la escena que puede ser asignado a cada procesador, de forma que cada procesador puede hacerse cargo de varios subentornos diferentes de la partición. Para simplificar la explicación, sin embargo, consideraremos en adelante un subentorno o subescena por procesador.

En nuestro trabajo hemos optado por una partición geométrica uniforme ajustada al contenido (ver Figura 5.6), utilizando planos alineados con los ejes para la división del espacio (Figura 5.6a). Se trata, en principio, de la construcción de una

simple rejilla tridimensional uniforme [60, 61], muy utilizada en informática gráfica por su versatilidad y sencillez. Como paso posterior, un proceso de ajuste de los límites del volumen de contorno de cada una de las celdas al contenido de la misma (Figura 5.6b) es llevado a cabo en todos los procesadores, cada uno de ellos sobre su subescena local. Con este paso se consigue eliminar parte de los espacios vacíos de la escena, permitiendo ahorrar ciertas comunicaciones, al evitar realizar algunas consultas de visibilidad innecesarias.

El primer paso para la creación de la partición consiste, por tanto, en la construcción de una rejilla tridimensional uniforme, con un determinado número de celdas o *voxeles* del mismo tamaño, sin consideraciones adicionales en cuanto a la composición o número de objetos en cada subescena. El número de subespacios en los que dividir la escena se elige en tiempo de ejecución, y será como mínimo el número de procesadores sobre los que se desea lanzar la ejecución paralela del método. En cuanto a la configuración exacta de la rejilla, por un lado, esta puede ser introducida como parámetro de entrada para la ejecución paralela, lo que puede resultar útil si se conoce de antemano la distribución de los objetos en la escena y qué reparto de la misma puede proporcionar unos mejores resultados. Opcionalmente, la configuración a utilizar puede calcularse de forma genérica a partir del número total de subespacios deseados mediante la aproximación presentada en [61]. Se trata de la aplicación de un algoritmo basado en la descomposición en factores primos del número de *voxeles* buscado:

1. Inicializamos el número de celdas por dimensión:

$$(p_x, p_y, p_z) = (1, 1, 1)$$

2. El número de celdas finales p se descompone en factores primos d_i,

$$p = \prod_{i=1}^{m} d_i$$

3. Los distintos factores primos se asignan en orden decreciente a las distintas dimensiones, de manera que el factor primo d_i se asignará a la dimensión con menor número de divisiones acumuladas hasta el momento:

$$p_j = \text{mín}(p_x, p_y, p_z)$$
$$p_j = p_j d_i$$

Así, por ejemplo, para 36 procesadores, con una rejilla de 36 celdas, tendríamos una lista ordenada de factores primos de $3, 3, 2, 2$, lo que proporcionaría una configuración automática de rejilla de $(3, 3, 4)$. Por supuesto, al no tener en absoluto en cuenta el contenido, la selección automática de divisiones por dimensión utilizada no garantiza un buen ajuste a la escena deseada, por lo que una configuración manual de la rejilla, escogida por el usuario en tiempo de ejecución, a menudo supondrá la creación de una partición más lógica de la escena a procesar.

Con la configuración de la rejilla uniforme a construir decidida, la creación de la partición uniforme se lleva a cabo simplemente con el trazado de los planos divisores alineados con los ejes necesarios. Para determinar en qué puntos del espacio es necesario colocar los planos de subdivisión es necesario que los procesadores conozcan el volumen de contorno (*bounding volume*) completo de la escena, información que deberán obtener bien como parámetro, bien incluida en el propio formato en el que esté guardada la escena de entrada, o bien calculándola ellos mismos, mediante una lectura y recorrido previo de la escena completa, sin necesidad de almacenar ningún polígono en memoria. Una vez trazados los planos divisores que crean la partición deseada, cada procesador solamente leerá de la escena el contenido del *voxel* o *voxeles* que, de forma unívoca, le han sido asignados, descartando el resto de polígonos de la escena, que en este caso ni siquiera llegan a guardarse en memoria. Por supuesto, los polígonos a caballo entre varios subentornos son subdivididos, y sus fragmentos asignados, cada uno, al subentorno correspondiente. La aplicación de una partición no uniforme, o de alguna técnica adicional destinada a equilibrar la carga computacional entre los procesadores, podría hacer necesaria una precarga completa de la escena, posiblemente en bloques de un determinado tamaño, para permitir un análisis de la misma basándose en su contenido. En cualquier caso, una vez construida la partición y repartidos los subespacios creados, cada procesador dispondrá únicamente de la parte de la escena que le haya sido asignada.

Una vez seleccionada la subescena, el volumen de contorno de la misma se recorta, ajustándolo al contenido de su interior (como veíamos en la Figura 5.6b). Este paso permite un ahorro posterior tanto de comunicaciones como de lanzamiento de rayos, al restringir, durante el cálculo de la radiosidad de la escena, las consultas de visibilidad entre procesadores a las zonas donde realmente hay polígonos. Un ejemplo de partición ajustada a contenido, y sus ventajas, se representa en la Figura 5.7. En este ejemplo, como se puede apreciar, tras una partición uniforme en 8 subespacios (Figura 5.7a), el ajuste al contenido evita consultas innecesarias (Figura 5.7b), como el lanzamiento de los rayos $R1 - R4$ en el subespacio P_3, que en la rejilla uniforme

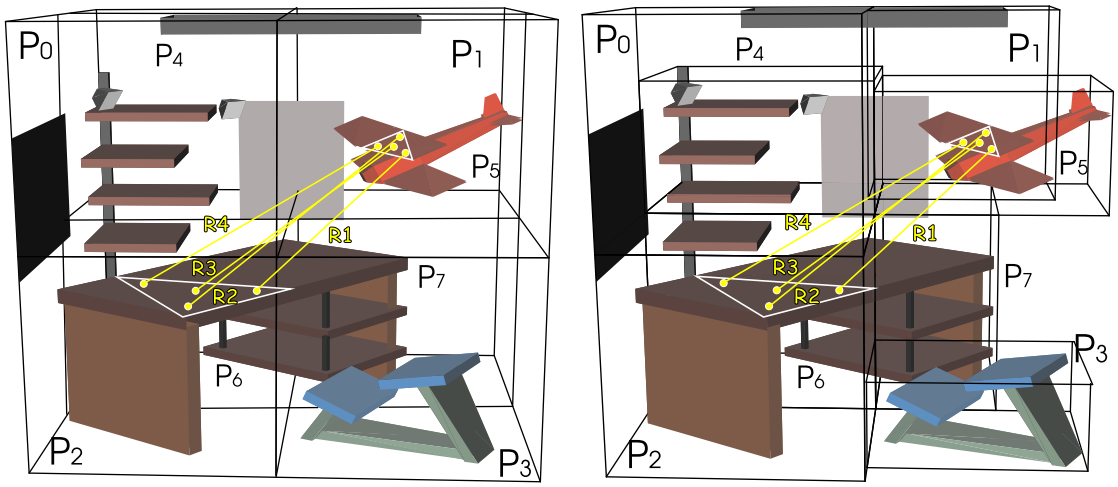


Figura 5.7: (a) Rejilla uniforme (b) Partición uniforme ajustada al contenido.

de la Figura 5.7a sí tenía que ser atravesado. En la subsección 5.3.4 explicaremos con más detalle el cálculo de la visibilidad en una escena distribuida en subescenas y con ajuste al contenido.

El aprovechamiento del posible recorte en el volumen de los distintos *voxeles* de la partición pasa por que todos los procesadores dispongan de la configuración final obtenida de la rejilla. Puesto que cada procesador se encarga únicamente del proceso de *ajuste al contenido* del volumen de contorno de su propia subescena, es necesario, entonces, un posterior intercambio de información entre los procesadores para disponer, en todos ellos, de la información acerca de qué volumen del espacio total de la escena corresponde a cada procesador. Esta comunicación (representada con la segunda etapa de la fase inicial en el diagrama de flujo de la Figura 5.5) consiste en el envío, y correspondiente recepción, por parte de cada procesador de las coordenadas de los puntos mínimo y máximo del volumen de contorno de su subespacio. Para llevar a cabo esta comunicación todos-a-todos utilizamos la función MPI *MPI_Allgather*, mediante la cual todos los procesadores envían las coordenadas de su *voxel* al resto y obtienen un vector con las coordenadas de los *voxeles* de todos los procesadores. Como ya se ha comentado, esta información hará posible optimizar el camino a seguir a la hora de comprobar la visibilidad de cada rayo lanzado entre polígonos de procesadores diferentes.

5.3.2. Matriz de interacción en un sistema distribuido

La última parte de la fase inicial del hilo de ejecución principal (en la Figura 5.5: *Precálculo sobre subescena local: BSP...*) corresponde a la construcción de las estructuras necesarias para la implementación de los optimizadores que posteriormente permitirán acelerar la resolución de las consultas de visibilidad posteriores, tanto locales como remotas (según se describe en detalle en el Capítulo 2).

Tras la etapa de precálculo de la fase inicial, y como paso preliminar al proceso iterativo de cómputo de la radiosidad jerárquica, introducimos en nuestro método, de forma análoga al esquema secuencial que veíamos en la Figura 1.7 del Capítulo 1, una etapa de cálculo de las interacciones iniciales (fase intermedia de la Figura 5.5), que aparece representada en detalle en la Figura 5.8. En esta figura se muestran los pasos más significativos de esta fase, indicando mediante el símbolo ✉ la información que es necesario comunicar en los pasos que afectan al cálculo de la interacciones remotas.

La matriz de interacciones iniciales, **K** (ver Ecuación 1.6 en el Capítulo 1), indica la lista de interacciones iniciales entre un polígono y el resto. Inicialmente se considera que dos elementos de la escena pueden interactuar si sus caras frontales están enfrentadas, es decir, si uno no se encuentra detrás del otro, independientemente de que pueda haber algún objeto en medio obstaculizando la interacción. Es el cálculo de los factores de forma, concretamente del término que corresponde al grado de visibilidad de la interacción, el que proporciona una matriz de interacciones iniciales definitiva. En el caso de nuestra implementación paralela, cada procesador sólo tiene una parte de los elementos de la escena, por lo que el cálculo de la matriz **K** implica, por un lado, un proceso de cálculo de las interacciones locales (interacciones entre los elementos de la subescena del procesador), y un proceso de cálculo de las interacciones remotas (interacciones entre elementos de un procesador y el resto de elementos de la escena, pertenecientes a subescenas de otros procesadores), por otro.

Como se ha explicado en el Capítulo 1, la matriz **K** de interacciones iniciales entre los polígonos de una escena es una matriz simétrica. Esta simetría permite, en el cálculo de las interacciones remotas, evitar realizar cálculos y comunicaciones redundantes. Como se ve en el paso 1 de la Figura 5.8, cada procesador va a actuar como solicitante de interacciones para un subconjunto del resto de procesadores (Lista P_s), y como receptor de peticiones de interacción remota para el resto de los procesadores (Lista P_r). Así, si tenemos p procesadores, cada uno de ellos establece comunicación con los $p-1$ procesadores restantes, lo que supone un total de $n =$

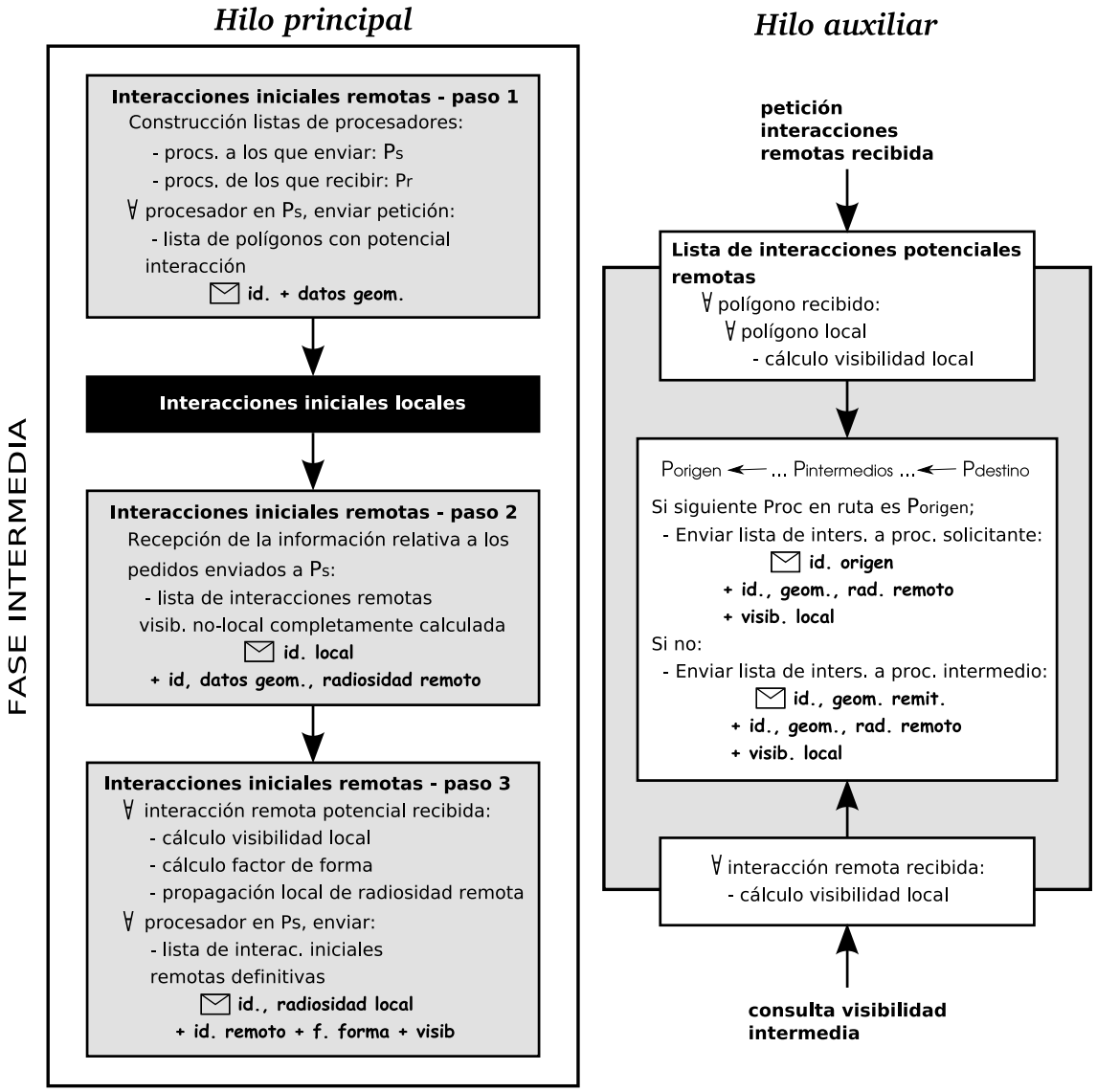


Figura 5.8: Interacciones iniciales.

$p(p-1)/2$ relaciones de comunicación procesador a procesador. En cada una de estas n relaciones, un procesador solicita a otro una lista con los polígonos de su subespacio que podrían interactuar con los suyos propios. El procesador que recibe y resuelve la petición inicia una cadena de propagación de la visibilidad de las interacciones remotas, proceso que veremos descrito en profundidad en la parte final de esta sección, mediante la cual el procesador solicitante recibe la lista definitiva de interacciones iniciales remotas entre los polígonos de ambos procesadores, que transmite a su vez al otro procesador.

El algoritmo utilizado para la construcción de las listas P_s y P_r de cada procesador trata de mantener un equilibrio entre los envíos y las recepciones. Para ello, de las n relaciones de comunicación que es posible establecer entre p procesadores, los $(n \ mod \ p)$ primeros procesadores del sistema serán solicitantes en $((n \div p) + 1)$ de las ocasiones, y receptores en $(n \div p)$, es decir, tendrán $((n \div p) + 1)$ elementos en la lista P_s y $(n \div p)$ elementos en P_r. El resto de los procesadores hará el papel de solicitante en $(n \div p)$ relaciones de comunicación, y el de receptor en $((n \div p) + 1)$. Se ha optado por el criterio de solicitar siempre a los procesadores inmediatamente contiguos, y recibir del resto, con lo que para una configuración de cuatro procesadores tendríamos el resultado mostrado en la Tabla 5.1, ya que:

$$p = 4 \quad \Rightarrow \quad n = \frac{4 \times 3}{2} = 6$$
$$6 \div 4 = 1, \quad 6 \ mod \ 4 = 2$$

Por tanto, las listas finales para nuestro ejemplo serían:

$$P_s^0 = \{1, 2\}, P_0^r = \{3\}$$
$$P_s^1 = \{2, 3\}, P_1^r = \{0\}$$
$$P_s^2 = \{3\}, P_2^r = \{0, 1\}$$
$$P_s^3 = \{0\}, P_3^r = \{1, 2\}$$

es decir, el procesador 0 actúa como solicitante respecto a los procesadores 1 y 2, y como receptor respecto a 3, el procesador 1 solicita interacciones a los procesadores 2 y 3 y recibe solicitudes de 0, y así sucesivamente. El hilo principal de cada procesador se encarga de construir un mensaje para cada destinatario de la lista P_s, con los polígonos que potencialmente pueden interaccionar con ese procesador, como se

Tabla 5.1: Reparto de los papeles de emisor/receptor entre los distintos procesadores para el cálculo de las interacciones iniciales.

$s\backslash r$	P_0	P_1	P_2	P_3
P_0		✓	✓	
P_1			✓	✓
P_2				✓
P_3	✓			

indica en la Figura 5.8. Los envíos consisten en las coordenadas de los vértices y los identificadores de los polígonos en el procesador al que pertenecen, y se realizan mediante operaciones de MPI no bloqueantes utilizando tipos de datos derivados.

La Figura 5.9 muestra una escena de ejemplo sobre la que se aplica una partición uniforme para cuatro procesadores ($P_0 - P_3$, sin considerar el ajuste al contenido de cada celda para simplificar el ejemplo). De acuerdo con este ejemplo, el cálculo de las interacciones iniciales se resolvería con el envío de los siguientes mensajes de petición entre los procesadores:

$$L_1^0 = \{1, 2, 6, 8\},\ L_2^0 = \{1, 2, 4, 6, 7, 9\}$$

$$L_2^1 = \{10, 12, 13, 14, 16\},\ L_3^1 = \{10, 12, 13, 16\}$$

$$L_3^2 = \{17, 20, 21, 22, 23\}$$

$$L_0^3 = \{25, 26, 27, 28, 29, 30, 32\}$$

donde L_j^i es la lista de polígonos del procesador i que pueden interactuar con polígonos del procesador j. Así, vemos en el ejemplo que el procesador 0 tiene que enviar sólo cuatro polígonos al procesador 1, $\{1, 2, 6, 8\}$, ya que son los únicos que por su orientación podrían interactuar con algún elemento del procesador 1.

Tras la fase de envío de peticiones, y antes de recibir la información pertinente a las interacciones remotas, se realiza el cálculo de las interacciones locales, como vemos en la Figura 5.8. De esta forma solapamos en lo posible el cálculo local sobre datos que ya están disponibles con la espera por la información remota solicitada. En la Figura 5.10 se muestra, a modo de ejemplo, la matriz $\mathbf{K}$ correspondiente a las cuatro particiones de la escena de la Figura 5.9. En esta matriz están indicadas, mediante un sombreado, las submatrices con interacciones locales que se tienen que

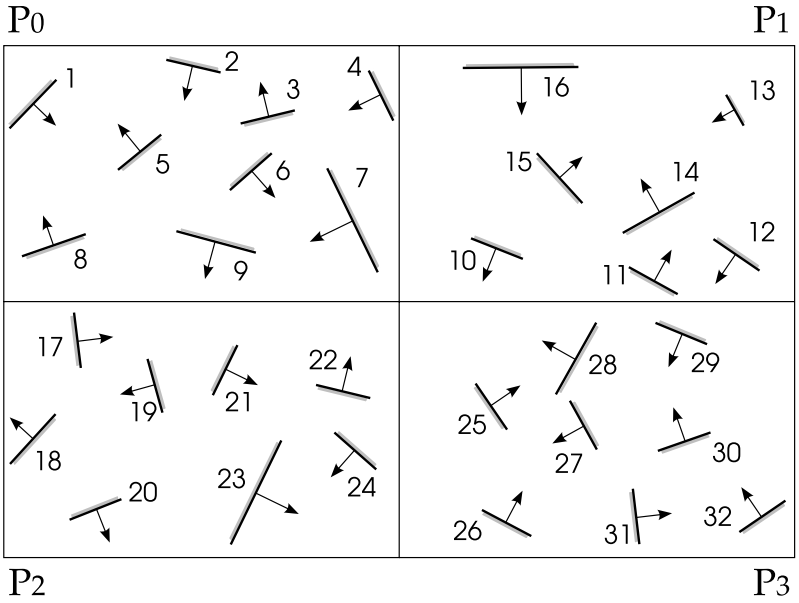


Figura 5.9: Escena con 32 polígonos subdividida en cuatro particiones.

calcular en cada partición, y con línea discontinua las submatrices correspondientes a interacciones remotas. Están marcadas con X en la figura las interacciones locales correspondientes. Así tenemos, por ejemplo, que el polígono 1 interacciona con los polígonos $\{2, 3, 4, 5, 8\}$. La matriz de la Figura 5.11 contiene, a su vez, todas las posibles interacciones iniciales, tanto las locales, que acabamos de ver, como las remotas, considerando las listas de polígonos L_j^i que pueden interactuar en cada procesador con el resto. Las interacciones remotas consideradas en esta matriz son, por tanto, todavía potenciales, a falta de la determinación de la visibilidad entre los objetos correspondientes.

De forma simultánea al cálculo de las interacciones locales, el hilo auxiliar da servicio (ver el flujo de la parte derecha en la Figura 5.8), de manera independiente al funcionamiento del hilo principal, tanto a las peticiones de interacciones iniciales remotas que recibe de los procesadores de la lista P_r, como a las consultas de visibilidad intermedias de interacciones entre procesadores entre cuyos subespacios se encuentra situado. La información de visibilidad de las interacciones remotas se propaga entre los hilos auxiliares de todos los procesadores cuyos subentornos se encuentran en la trayectoria de estas interacciones, como veremos en la parte final de esta sección. Las interacciones que en algún momento del camino son obstaculizadas por completo son descartadas, llegando el resto, al final del proceso, al procesador solicitante, donde se completa el cálculo de las interacciones iniciales remotas con

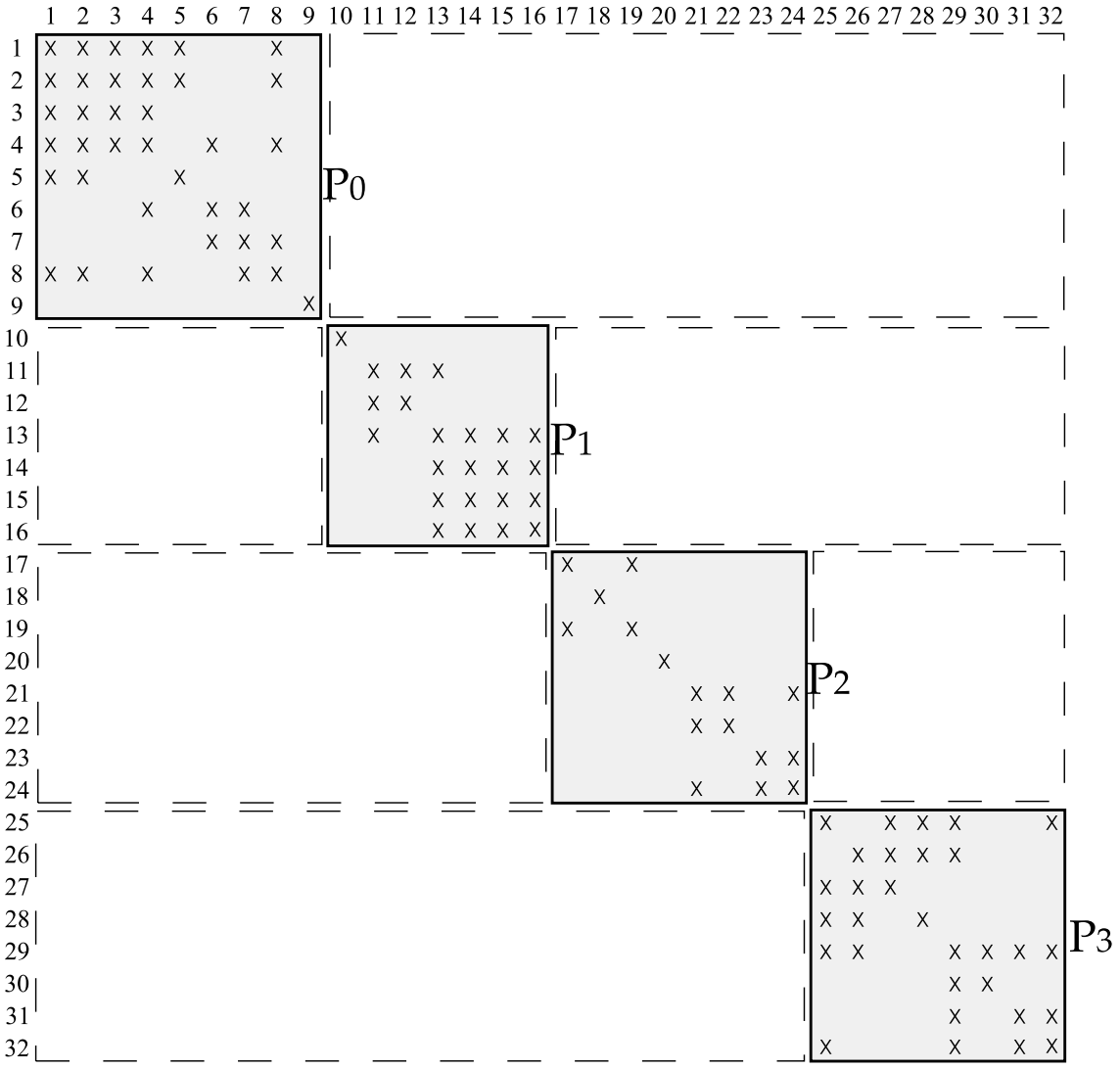


Figura 5.10: Partición sobre una matriz de interacciones: interacciones locales.

la determinación de la visibilidad en la subescena local. Como resultado, se obtiene parte de la matriz de interacciones iniciales definitiva (ver Figura 5.12), para la que se calculan los factores de forma correspondientes que permiten la propagación de la energía remota en la subescena local (paso 2 de la Figura 5.8).

Como último paso, la información final de las interacciones iniciales remotas se envía al procesador de la lista P_s correspondiente, lo que le permitirá establecer las interacciones simétricas correspondientes (paso 3 de la Figura 5.8). El resultado de esta fase es una matriz de interacciones iniciales distribuida entre todos los procesadores del sistema.

Al final, siguiendo con nuestro ejemplo de la Figura 5.9, obtenemos la matriz de interacción definitiva, que se muestra en la Figura 5.12). Así, por ejemplo, las interacciones remotas potenciales para el polígono 1 del procesador 0 en el procesador 3 son los polígonos $\{26, 27, 28, 29, 30, 32\}$ (ver Figura 5.12). Tras el cálculo de la visibilidad de estas interacciones, primero en la subescena del propio procesador 3 (Figura 5.10), luego en cada uno de los subespacios intermedios correspondientes, como veremos en la subsección 5.3.4, y finalmente en la subescena del procesador solicitante, la única interacción remota de ese polígono con polígonos del procesador 3 es con el polígono 28 (como se aprecia en la matriz final de la Figura 5.12).

5.3.3. Distribución de la energía: refinado perezoso

La esencia del método de radiosidad es la distribución de la energía en un entorno, tarea que se lleva a cabo durante el proceso iterativo de la fase principal que veíamos en la Figura 5.5. Respecto al proceso iterativo del método secuencial (ver Sección 1.3 del Capítulo 1), la partición del entorno en varios fragmentos, de los cuales pueden ocuparse procesadores diferentes, nos va a permitir segmentar el proceso de distribución de la energía en el interior de cada subescena en dos etapas diferentes: por un lado, tenemos la difusión de la energía local de la subescena en el propio subespacio de la misma, que no requiere información alguna del exterior de la subescena, y, por otro lado, la distribución de la energía proveniente del resto de la escena en el interior del subentorno. Ambas etapas, como veremos, se intercalan, tanto durante el proceso iterativo como en la fase previa de cálculo de las interacciones iniciales, con el objetivo de solapar en la medida de lo posible comunicación y computación.

El cálculo de la radiosidad local de cada subescena se resuelve simplemente apli-

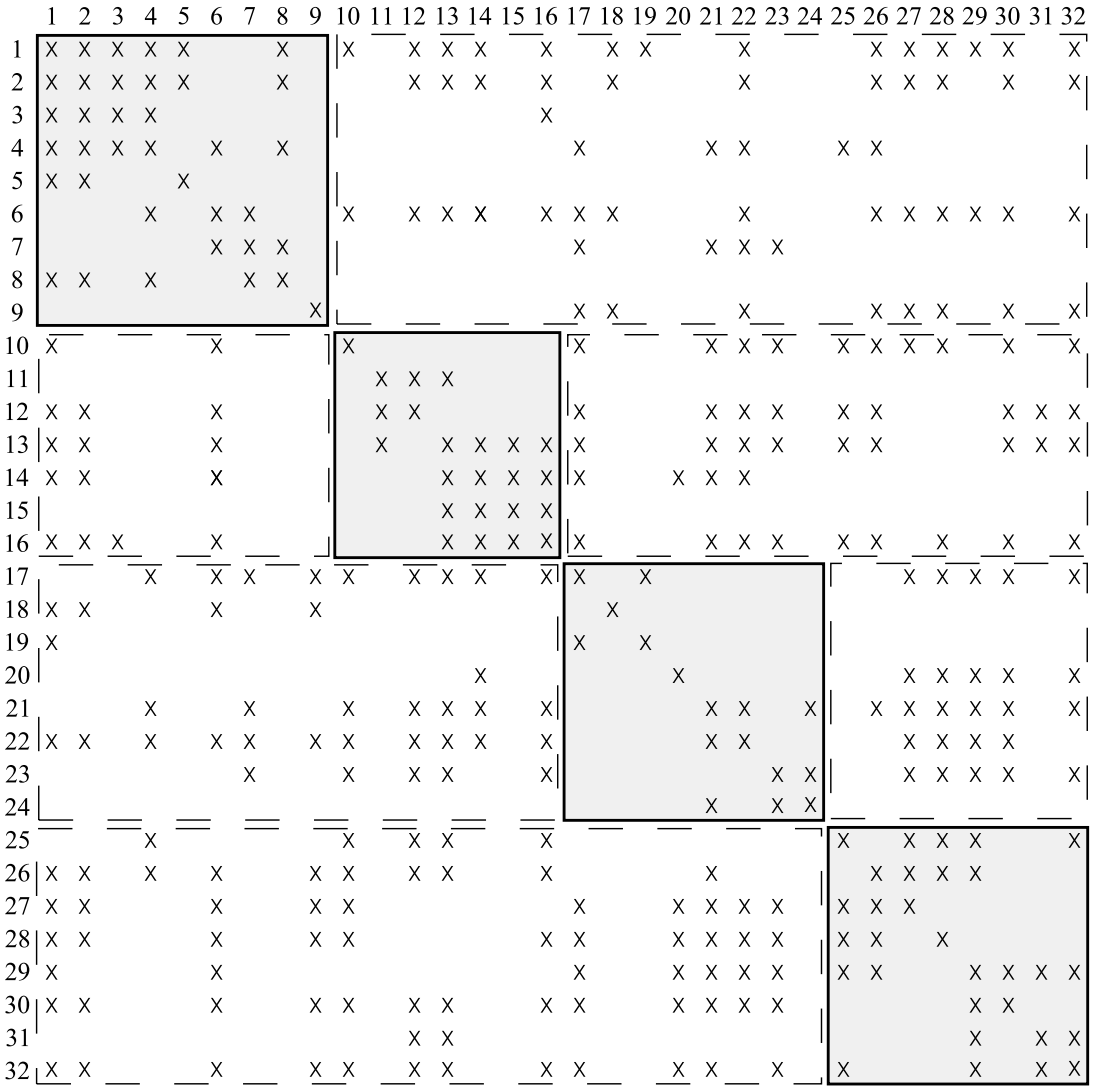


Figura 5.11: Matriz potencial de interacciones iniciales.

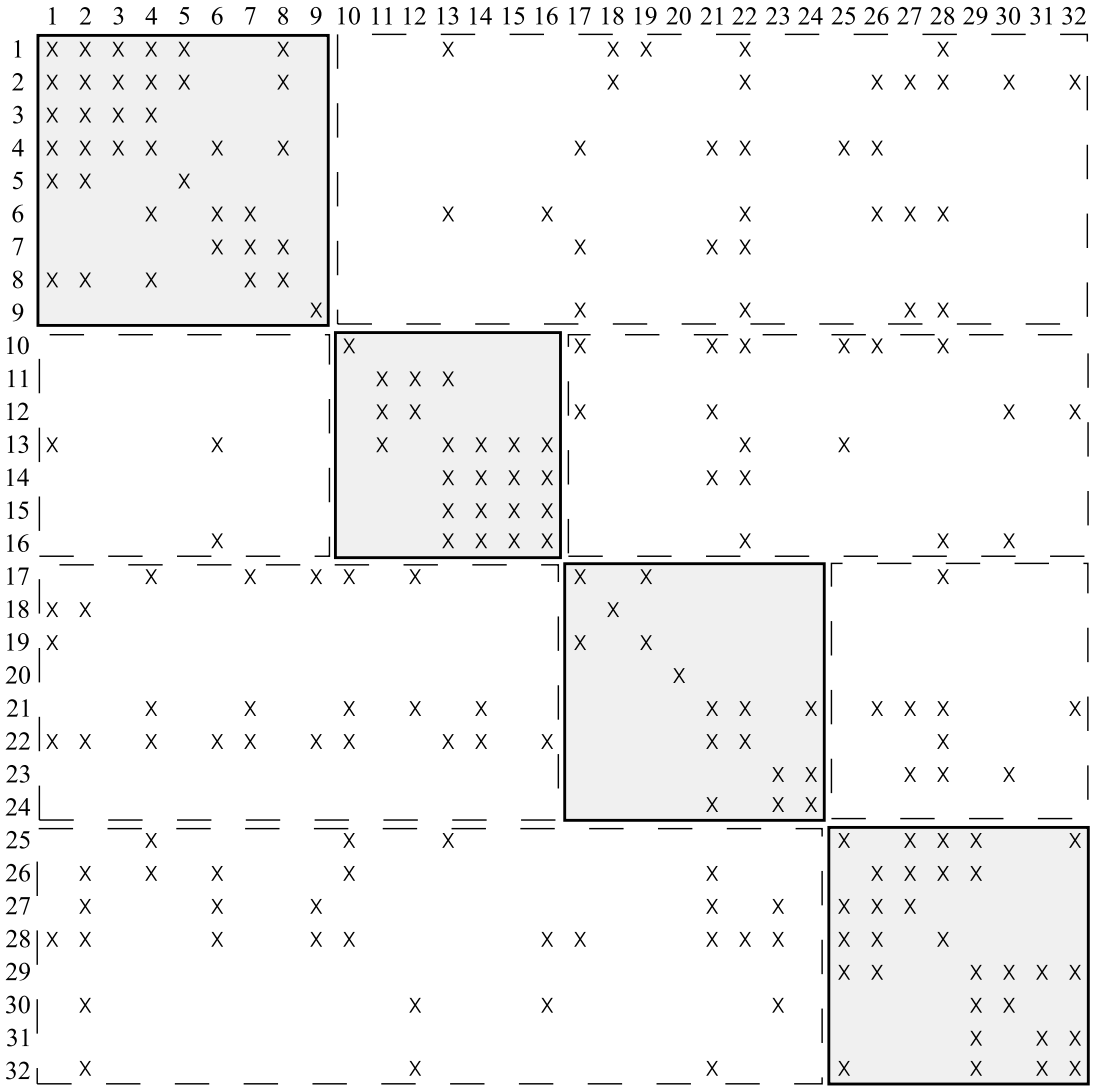


Figura 5.12: Matriz definitiva de interacciones iniciales.

cando el esquema de acumulación de energía habitual del método de radiosidad jerárquica (ver Sección 1.3 del Capítulo 1). Así, en cada iteración, a cada triángulo de la escena se le aplican, sucesivamente, los pasos de: refinado, con el que se obtiene una jerarquía de triángulos que interactúan a distintos niveles con el resto de la escena (*refinement*); cálculo de la energía recibida por el elemento en cada uno de esos niveles de refinado (*gathering*); y sincronización de la jerarquía (*sweeping*), transmitiendo la energía recibida en los diferentes niveles hasta dejar la jerarquía de elementos en un estado coherente. El método de refinado que hemos aplicado es el refinado *BF* [64], habitual en radiosidad jerárquica y ya comentado en el Capítulo 1. Los criterios bajo los cuales se decide si una interacción es o no refinada se basan tanto en la cantidad de energía transportada por la interacción como en el grado de visibilidad de la misma ($B_j \times F_{ij}$ y V_{ij}, según se detalla en la Sección 1.3 del Capítulo 1).

Al tratarse de espacios convexos y disjuntos, el cálculo del factor de forma, considerando el valor de visibilidad como una parte del mismo, entre dos polígonos cualesquiera en el interior de uno de los subespacios se realiza sin necesidad de información ajena al subentorno, por lo que no se requiere ningún tipo de comunicación entre los procesadores para el desarrollo de cada iteración de esta fase. Este hecho permite realizar, cuando son necesarios, múltiples refinados recursivos de las interacciones en cada iteración, de la misma forma que en el algoritmo secuencial, acelerando la convergencia. Como resultado de esta manera de aplicar el refinado, el grueso de las interacciones se ven refinadas en la primera iteración.

En la Figura 5.13 se representa, a modo de ejemplo, una iteración de este tipo para la interacción entre dos elementos. El primer paso consiste en el refinado exhaustivo de la interacción, que es refinada recursivamente hasta alcanzar el umbral prefijado de refinado *BF*, calculando en cada paso los valores de factor de forma y visibilidad correspondientes. Como resultado se obtienen dos jerarquías de elementos que interactúan en distintos niveles de refinado. La propagación de la energía de la interacción refinada (*gathering*) se reparte así en varios niveles de detalle, lo que hace necesaria, antes de iniciar la siguiente iteración, una fase de sincronización de la energía recibida en los diferentes niveles de la jerarquía del elemento destino de la interacción.

En el caso de las interacciones remotas, al no disponer de los datos geométricos del polígono fuente, un refinado exhaustivo de la interacción supone un excesivo número de comunicaciones, no solo con el procesador encargado del polígono, sino también con los procesadores que tienen asignadas particiones intermedias. Para

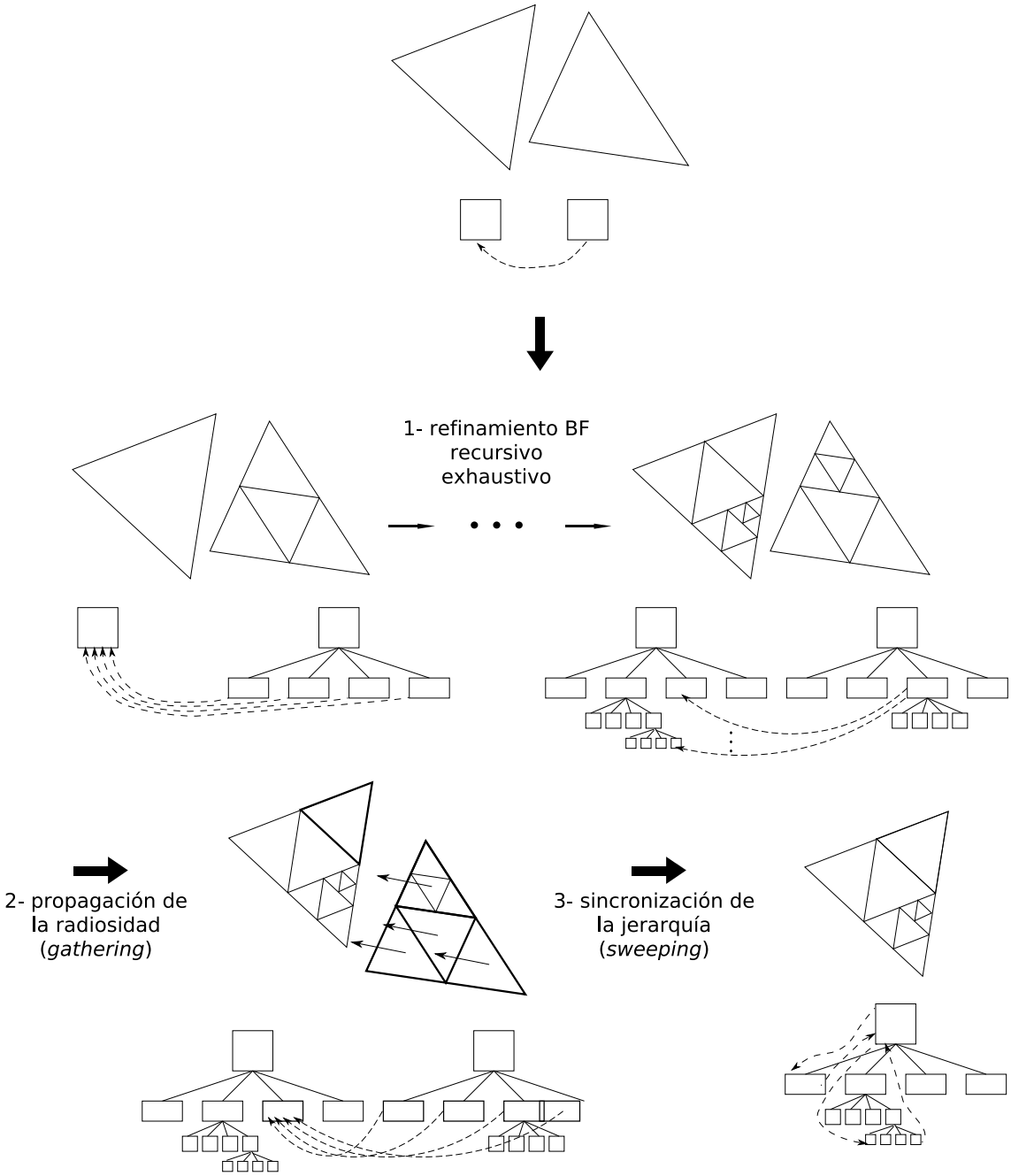


Figura 5.13: Iteración local en el refinado paralelo.

reducir los requisitos en cuanto a comunicación entre procesadores, y así poder intercalar de manera efectiva etapas locales y remotas en cada iteración, aplicamos lo que denominamos un refinado *perezoso*, de forma que en cada iteración remota las interacciones solo van a ser refinadas, si es necesario, un único nivel de profundidad (ver Figura 5.14).

La Figura 5.14 muestra, en comparación con el refinado local de la Figura 5.13, las tres primeras iteraciones de la aplicación de un refinado *BF* perezoso. Como se puede apreciar, una interacción es refinada un solo nivel en cada iteración, tras la cual se realizan las fases de propagación de la energía y sincronización de la jerarquía creada. En la figura, en la primera iteración se propaga la energía de una de las interacciones remotas iniciales. En la segunda iteración, el paso de refinado determina que la transferencia de energía para esa interacción es demasiado elevada, según los criterios del refinado BF aplicado, por lo que la interacción debe ser refinada. En este caso, concretamente, se procede subdiviciendo el polígono destino de la interacción, resultando cuatro nuevas interacciones que sustituyen a la vieja. El proceso de propagación se realiza teniendo en cuenta ahora las nuevas interacciones, y el paso final de sincronización deja las estructuras jerárquicas en un estado coherente con la energía recibida. En la tercera iteración, las interacciones fruto del refinado anterior son todavía demasiado gruesas según el criterio del refinado *BF* utilizado, por lo que todas ellas son refinadas, subdividiendo en esta ocasión, sin embargo, el polígono fuente de la interacción. Un nuevo paso de propagación de la energía y sincronización de las jerarquía finalizaría la iteración.

La utilización de un refinado perezoso nos permite resolver el procesado completo de una interacción remota en un único paso, evitando el excesivo cruce de mensajes de comunicación que necesitaría un refinado exhaustivo en cada iteración. El efecto más destacable que produce el cambio introducido con el refinado perezoso es, respecto al refinado exhaustivo utilizado en el método secuencial, un retraso en el refinado de las interacciones remotas, refinado que ahora es espaciado a lo largo de las distintas iteraciones en lugar de acumularse principalmente en la primera iteración. Al depender la convergencia final del algoritmo únicamente de la iluminación global de la escena alcanzada, la introducción del refinado perezoso no implica una pérdida significativa de la calidad de la imagen final, pudiendo, en todo caso, simplemente requerir alguna iteración más que el proceso iterativo jerárquico clásico. Sin embargo, muchas de las interacciones remotas corresponden a objetos más alejados entre sí, siendo habitualmente menos influyentes en el resultado final de la iluminación global, lo que permite alcanzar la convergencia en la mayoría de las ocasiones en el

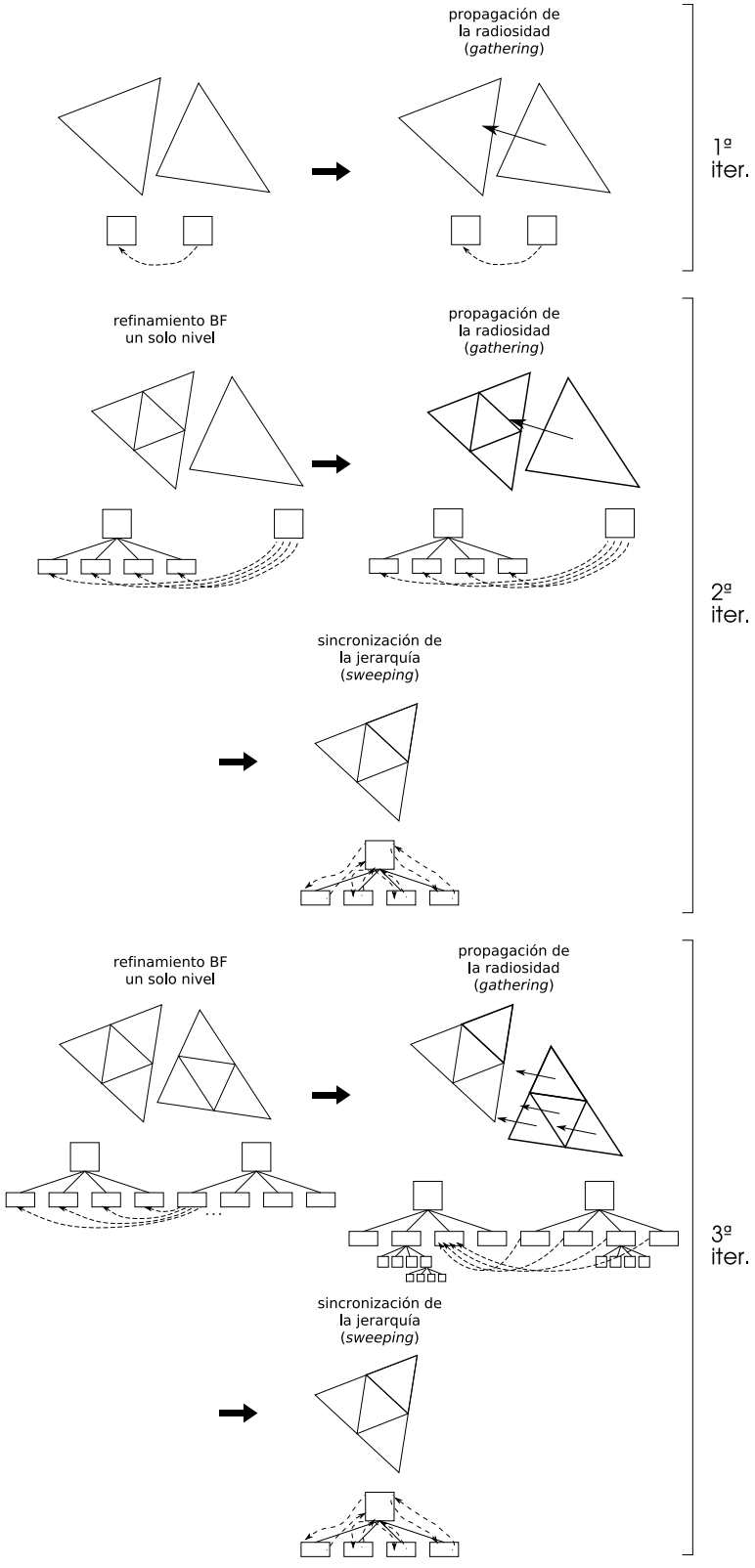


Figura 5.14: Iteración remota con un refinado perezoso.

mismo número de iteraciones que el caso secuencial, y sin la introducción de error adicional alguno.

En la Figura 5.15 vemos representado en detalle el desarrollo completo de una iteración en nuestro algoritmo paralelo, que correspondería con la fase principal del flujo de la Figura 5.5. Cada iteración remota consta de tres fases, que se intercalan con la iteración ejecutada en la subescena local, una de inicialización y dos de cálculo. En la fase de inicialización se comprueba qué interacciones remotas necesitan ser refinadas, aplicando un único nivel de refinado *BF*. Las interacciones que necesitan ser refinadas se envían al procesador que dispone del elemento fuente de la interacción, para que inicie el proceso de determinación de la visibilidad paralelo de las interacciones resultantes, como se describe en el siguiente apartado. Además, se solicitan los valores actualizados de radiosidad de todas las interacciones remotas (implícitamente se incluyen las interacciones que surgen de las que van a ser refinadas).

Tras el desarrollo de una iteración completa de radiosidad jerárquica sobre el entorno local, se completa la iteración remota iniciada. Para ello se recoge toda la información que atañe a las interacciones remotas, enviada por el resto de procesadores. Esta información nos va a permitir, tras completar la información de visibilidad remota con la visibilidad en el subentorno local, propagar la energía remota en la subescena.

5.3.4. Determinación distribuida de la visibilidad mediante mapas de bits

La interacción necesaria entre procesadores para una correcta distribución de la energía entre subescenas es la parte crítica de la paralelización, al requerir una considerable cantidad de comunicación y, por ello, una cierta coordinación entre los procesadores. El intercambio de mensajes entre procesadores se centra en dos vertientes: la transmisión de los valores de energía y el cálculo de los factores de forma y la visibilidad entre polígonos.

Mientras que la propagación de la energía entre subescenas mediante el intercambio de mensajes entre procesadores es la práctica habitual en la mayoría de los trabajos anteriormente comentados [13, 118], incluida nuestra primera aproximación paralela descrita en la Sección 5.2, el desarrollo de un esquema paralelo para la determinación de la visibilidad en un entorno distribuido sin la replicación de la

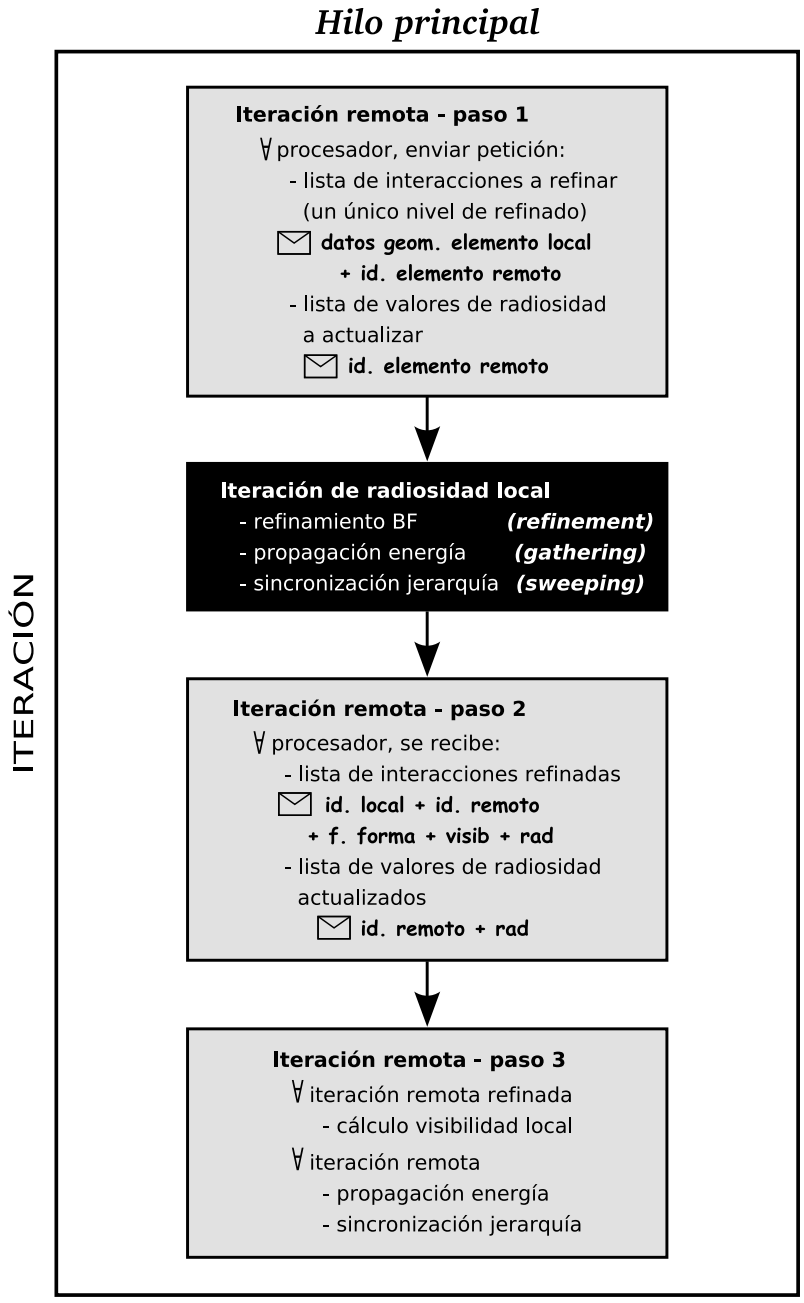


Figura 5.15: Iteración paralela.

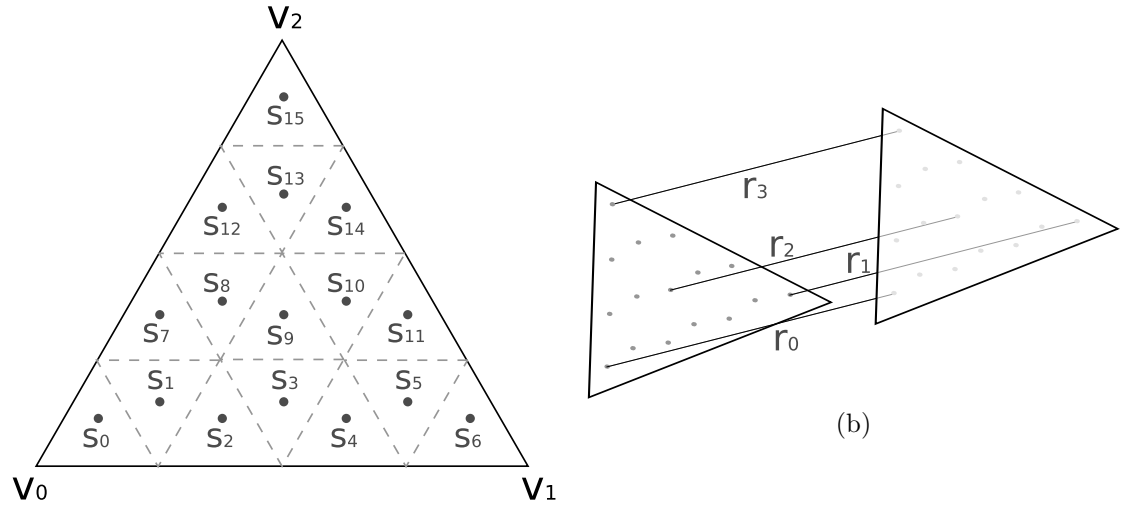
geometría completa de la escena inicial no es, ni mucho menos, tan frecuente. Específicamente, en lo que a radiosidad se refiere, lo más habitual es el empleo de técnicas basadas en el uso de fronteras y máscaras de visibilidad, como las ya comentadas en la introducción de este capítulo [5, 9]. El uso de este tipo de técnicas, sin embargo, introduce un factor de simplificación importante en el cálculo de la visibilidad, con un error de difícil cuantificación. Nuestra solución, sin embargo, se aproxima más a la idea de un lanzamiento de rayos distribuido como el utilizado en algunas implementaciones paralelas de *ray tracing* como Kilauea [76], de forma que no se introduce simplificación ni error alguno respecto al caso secuencial.

Como se ha visto a lo largo del Capítulo 2, la determinación de visibilidad entre dos polígonos se calcula habitualmente como la proporción de rayos lanzados entre ambos polígonos que alcanzan su destino sin encontrar obstáculo alguno (ver Ecuación 2.1). Por tanto, para cada rayo $\vec{r_i}$ trazado entre los polígonos A y B, el valor V_i^{AB} nos indica si la trayectoria del rayo es interceptada por algún objeto de la escena.

La codificación de la información de visibilidad, para su transmisión entre los procesadores, se realiza mediante mapas de bits, que representan el resultado del lanzamiento de rayos. Para cada rayo lanzado se considera únicamente la información binaria que indica si el rayo ha alcanzado o no su destino, información que se codifica con el bit correspondiente a 1 o a 0, respectivamente. Los rayos que encuentran algún obstáculo, y que por tanto son codificados con un 0, no vuelven a ser lanzados. Así, la visibilidad entre los polígonos A y B se puede representar mediante el siguiente mapa de bits:

$$V^{AB} = [V_1^{AB}\ V_2^{AB}\ \ldots\ V_L^{AB}]$$

donde L es el número de rayos lanzados utilizado para el cálculo de la visibilidad. La técnica utilizada para el trazado de rayos entre polígonos es de tipo determinístico, lo que permite lanzar, para cada test de visibilidad entre dos polígonos, los mismos rayos y en el mismo orden a cualquier procesador, sin necesidad de aportar información adicional sobre los rayos lanzados. En la Figura 5.16a se muestra un esquema de este tipo de lanzamiento de rayos: se utiliza un número predeterminado de muestras fijas por polígono, relativas a los vértices del mismo, y se lanza el número de rayos deseado entre ellas, siguiendo un orden y configuración también previamente fijado. En la Tabla 5.2 se muestran los ocho primeros rayos de una lista predeterminada de configuraciones para rayos entre dos triángulos. En el ejemplo de la Figura 5.16b,



(a) (b)

Figura 5.16: Lanzamiento determinístico de rayos: (a) Muestras predeterminadas (b) Lanzamiento de cuatro rayos.

para el lanzamiento de cuatro rayos entre dos triángulos, se escogieron los primeros cuatro rayos de la Tabla 5.2, por lo que los rayos se trazarán, el primero, desde la muestra s_o del triángulo origen a la muestra s_6 del destino, el segundo, desde la s_6 a la s_0, y así sucesivamente.

En nuestro caso, al estar la escena distribuida entre las memorias de los diferentes procesadores, es necesario comprobar la información de los objetos de los diferentes subespacios que atraviesa cada rayo. Para ello, la resolución de una consulta de visibilidad pasa ahora por que cada procesador realice, para un mismo conjunto de rayos, el test de intersección de cada rayo con su subescena local. Para cada rayo lanzado, $\vec{r_i}$, obtendríamos su visibilidad como el producto del resultado de la evaluación de su lanzamiento en cada una de las particiones de la escena, P_j, por las que pase $\vec{r_i}$. De esta forma:

$$V_i^{AB} = [V_i^{AB}]^{P_0}\ [V_i^{AB}]^{P_1}\ [V_i^{AB}]^{P_2}\ \ldots\ [V_i^{AB}]^{P_j} \tag{5.1}$$

donde

$$[V_i^{AB}]^{P_j} = \begin{cases} 0 & \text{si } \vec{r_i} \text{ es interceptado por algún objeto de la partición } P_j \\ 1 & \text{en caso contrario} \end{cases}$$

Tabla 5.2: Lanzamiento determinístico de rayos: ocho primeros elementos de una tabla de rayos.

rayo	origen	destino
0	s_0	s_6
1	s_6	s_0
2	s_{15}	s_{15}
3	s_9	s_9
4	s_2	s_{12}
5	s_4	s_{14}
6	s_7	s_{11}
7	s_3	s_{13}
...		

Así, en el ejemplo de la Figura 5.17, donde se indica con líneas discontinuas la partición uniforme y con trazo continuo el resultado de la aplicación del ajuste a contenido, para el rayo r_1 se obtendría:

$$\begin{aligned} V_1^{AB} &= [V_1^{AB}]^{P_0}\ [V_1^{AB}]^{P_1}\ [V_1^{AB}]^{P_3} \\ &= 1\ 0\ 1 = 0 \end{aligned}$$

De manera general, y utilizando como notación para cada subespacio las coordenadas en las tres dimensiones (x, y, z) de la posición que ocupa en la rejilla uniforme (según veíamos en el apartado *Partición geométrica uniforme adaptada al contenido*), representaríamos el conjunto de subespacios a atravesar para una consulta de visibilidad entre las particiones origen (o_x, o_y, o_z) y destino (d_x, d_y, d_z) con la siguiente expresión:

$$\prod_{i_x=o_x}^{d_x} \prod_{i_y=o_y}^{d_y} \prod_{i_z=o_z}^{d_z} i_x i_y i_z \tag{5.2}$$

con lo que la visibilidad para un rayo $\vec{r_i}$ sería

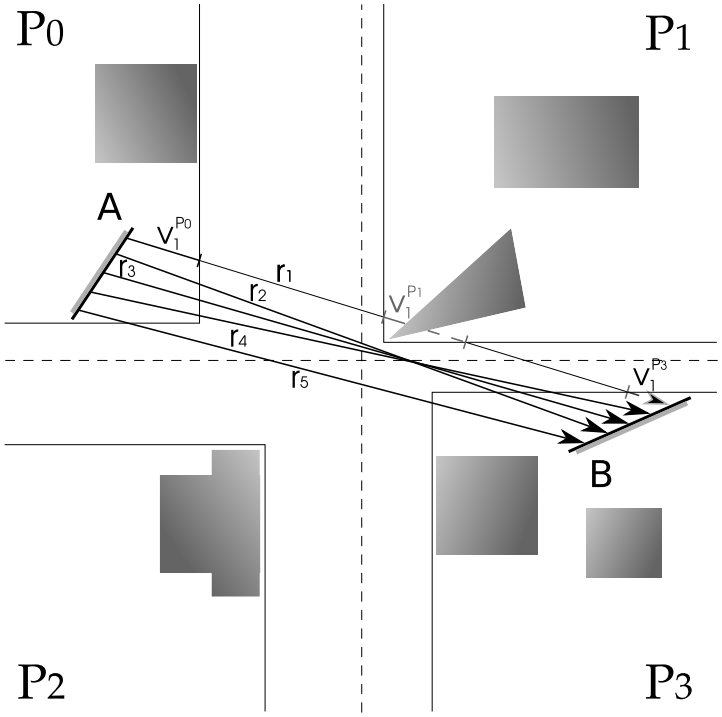


Figura 5.17: Lanzamiento de rayos entre polígonos de particiones diferentes.

$$V_i^{AB} = \prod_{i_x=o_x}^{d_x} \prod_{i_y=o_y}^{d_y} \prod_{i_z=o_z}^{d_z} [V_i^{AB}]^{i_x i_y i_z} \tag{5.3}$$

teniendo en cuenta que el ajuste a contenido aplicado sobre la partición uniforme minimiza el número de particiones que es necesario atravesar. Así, durante la fase de precálculo inicial (ver Figura 5.5) cada procesador calcula y almacena en una tabla, para cada combinación de particiones origen-destino, las particiones que es necesario atravesar para las consultas de visibilidad. Para p particiones, esta tabla tiene un total de $p(p-1)/2$ entradas, teniendo en cuenta su simetría Cada entrada de la tabla es un mapa de bits en el que se indica, con un valor 1 en la posición correspondiente, qué procesadores hay que atravesar para esa combinación origen-destino. Así, para p procesadores las entradas estarían formadas por palabras de p bits de la forma:

$$W_{ij} = b_0 b_1 b_2 \dots b_{p-1}$$

con valor uno en el bit b_k si el procesador p_k tiene que ser visitado en el proceso de

Tabla 5.3: Tabla con los procesadores a visitar para resolver las consultas de visibilidad entre cuatro procesadores.

	P_0	P_1	P_2	P_3
P_0		W_{01}	W_{02}	W_{03}
P_1			W_{12}	W_{13}
P_2				W_{23}
P_3				

determinación de visibilidad entre elementos de las subescenas de los procesadores p_i y p_j, y cero en caso contrario. Para cuatro procesadores con la partición de la Figura 5.17 tendríamos una tabla como la de la Tabla 5.3, con palabras de 4 bits como entradas, con lo que la información relativa a los procesadores a recorrer para las consultas de visibilidad se guardaría, en cada procesador, en un vector de 6 palabras de 4 bits de la forma:

$$W = \underbrace{1100}_{W_{01}}\ \underbrace{1010}_{W_{02}}\ \underbrace{1101}_{W_{03}}\ \underbrace{0111}_{W_{12}}\ \underbrace{0101}_{W_{13}}\ \underbrace{0011}_{W_{23}}$$

Por tanto, la determinación de visibilidad para las interacciones remotas se resuelve mediante la propagación, a través de los procesadores indicados en el vector W, de los rayos lanzados para esas interacciones.

El cálculo de la visibilidad realizado mediante la evaluación distribuida del lanzamiento de rayos puede implementarse, al menos, de dos maneras diferentes. En ambos casos, el proceso de propagación del resultado del lanzamiento de rayos comienza con un procesador que necesita calcular el valor de visibilidad para las interacciones remotas de sus polígonos locales. Una aproximación posible, consiste en, una vez calculada la visibilidad en el subentorno local de ese procesador, propagar el resultado del lanzamiento de rayos de forma encadenada, uno tras otro, por los diferentes procesadores a cargo de las subescenas que es necesario atravesar (Ecuación 5.2), de manera que en cada salto de procesador se van reduciendo tanto el número de rayos que es necesario lanzar como el número de interacciones que se comunican (las interacciones que son totalmente obstaculizadas se van descartando). Al final de la cadena, el procesador destinatario de las interacciones recibirá un único mensaje con la visibilidad de esas interacciones remotas. La otra opción consiste en la evaluación simultánea del lanzamiento de rayos en todos los procesadores intermedios, a los que el procesador que inicia el proceso envía el resultado de su test de visibilidad local.

De esta forma, el procesador receptor final tiene que recibir un mensaje de cada procesador intermedio, con los que compondrá la visibilidad final de las interacciones remotas. Esta aproximación presenta el inconveniente de realizar y transmitir cálculos de visibilidad innecesarios, al evaluarse un mismo rayo simultáneamente en varios procesadores, lo que nos ha hecho decantarnos por la primera solución.

El proceso de cálculo de la visibilidad se inicia con un primer paso que corresponde a la determinación de la visibilidad local, que supone el trazado de todos los rayos contemplados para el cálculo de la visibilidad entre polígonos y el consiguiente test de intersección rayo-subescena para cada uno de ellos. El lanzamiento de rayos nos permite obtener el mapa de bits de partida para cada interacción, que será enviado, junto con la información geométrica necesaria, al siguiente procesador cuya subescena se encuentre en la ruta de los rayos lanzados hasta alcanzar el procesador destino de las interacciones a procesar, donde finalmente se dispondría de la información de visibilidad completa que sería transmitida al procesador que había iniciado el proceso.

Tras esto, la información geométrica de los polígonos, y los campos de bits con los rayos todavía no interceptados para las distintas interacciones es enviada al siguiente procesador en la cadena de visibilidad que es atravesado por alguno de los rayos lanzados. Por supuesto, si un rayo no atraviesa el volumen de contorno de la subescena de un procesador, el rayo no es lanzado.

El proceso de propagación de la visibilidad terminaría cuando el siguiente procesador a recibir la información de visibilidad es el origen de las interacciones que están siendo comprobadas. En este caso la información geométrica enviada es únicamente la correspondiente a los polígonos remotos con los que interactúan sus polígonos locales, de los cuales se envía únicamente un identificador.

En la Figura 5.18 se muestra un ejemplo bidimensional, en el que el procesador P_5 quiere obtener el valor de visibilidad para las interacciones, i_0 y i_1, de dos de sus elementos con elementos del procesador P_0. Para ello P_5 envía a P_0 una petición, que incluye el envío de la geometría (tres vértices por elemento, si se trata de triángulos) de los elementos afectados (como hemos visto en las Figuras 5.8 y 5.15). Suponiendo que se lanzan cuatro rayos para la determinación de la visibilidad de una interacción, el lanzamiento distribuido de los rayos en este caso se iniciaría en P_0 (Figura 5.18b). Así, el procesador P_0 realiza, para cada uno de los cuatro rayos lanzados, el test de intersección rayo-escena con su subescena local. Como resultado del mismo, son obstaculizados un rayo de la interacción i_0, y dos rayos de la interacción i_1, con lo que

las máscaras iniciales obtenidas tendrán el valor de 0111 y 0011, respectivamente, que indicarán a los siguiente procesadores que el primero de los rayos lanzados para la interacción i_0 ha sido ya previamente interceptado, igual que el primer y segundo rayo de i_1, por lo que no tienen que volver a ser lanzados en el resto de subentornos de la escena. P_0 enviará la información de visibilidad obtenida a P_1, que es el siguiente procesador en la cadena de visibilidad $P_0 - P_5$. En este procesador no se lanza ninguno de los rayos de i_0, al no atravesar ninguno de ellos la subescena (Figura 5.18c). Los dos rayos activos de i_1 sí son lanzados, no encontrando obstáculo alguno en la subescena. El siguiente procesador en la cadena de visibilidad sería P2, de acuerdo con la configuración de la rejilla, sin embargo, ninguno de los rayos lanzados atraviesa el subespacio de P2. Se para, por tanto, directamente al siguiente procesador relevante, en este caso P_3, que recibiría de P_1 la información necesaria para continuar con el proceso (Figura 5.18d).

La máscara de visibilidad final, tras el lanzamiento de los rayos que quedan activos cuando el proceso llega al procesador que inició la petición, P_6, sería 0100 para i_0, y 0011 para i_1, lo que indica que únicamente el rayo 1 de la interacción i_0 y los rayos 2 y 3 de i_1 no han sido obstaculizados. El porcentaje de visibilidad que se obtendría para ambas interacciones sería de 0.25 (1/4) y 0.5 (2/4), respectivamente.

5.3.5. Resultados experimentales

Al igual que en el caso del método paralelo de la Sección 5.2, la implementación se ha desarrollado íntegramente en el lenguaje de programación C, con la librería de paso de mensajes *MPI* (*MPI-1.2*) para las tareas de comunicación. Además, se ha utilizado la librería *POSIX thread* (*pthreads*), API estándar de C y C++ para la programación multi-hilo.

Como la obtención de una solución genérica para distintos tipos de arquitecturas paralelas, tanto con memoria distribuida como compartida, era uno de los objetivos principales de nuestro diseño, se ha tratado de probar la implementación realizada sobre un conjunto lo más variado posible de sistemas. En este caso, las plataformas de prueba utilizadas difieren de las empleadas para el primer método paralelo presentado en este trabajo (Sección 5.2), debido a la diferencia temporal entre ambas implementaciones. Concretamente, las máquinas que han proporcionado los resultados expuestos en esta sección son:

- Un sistema *ccNUMA* formado por dos servidores HP Integrity Superdome con

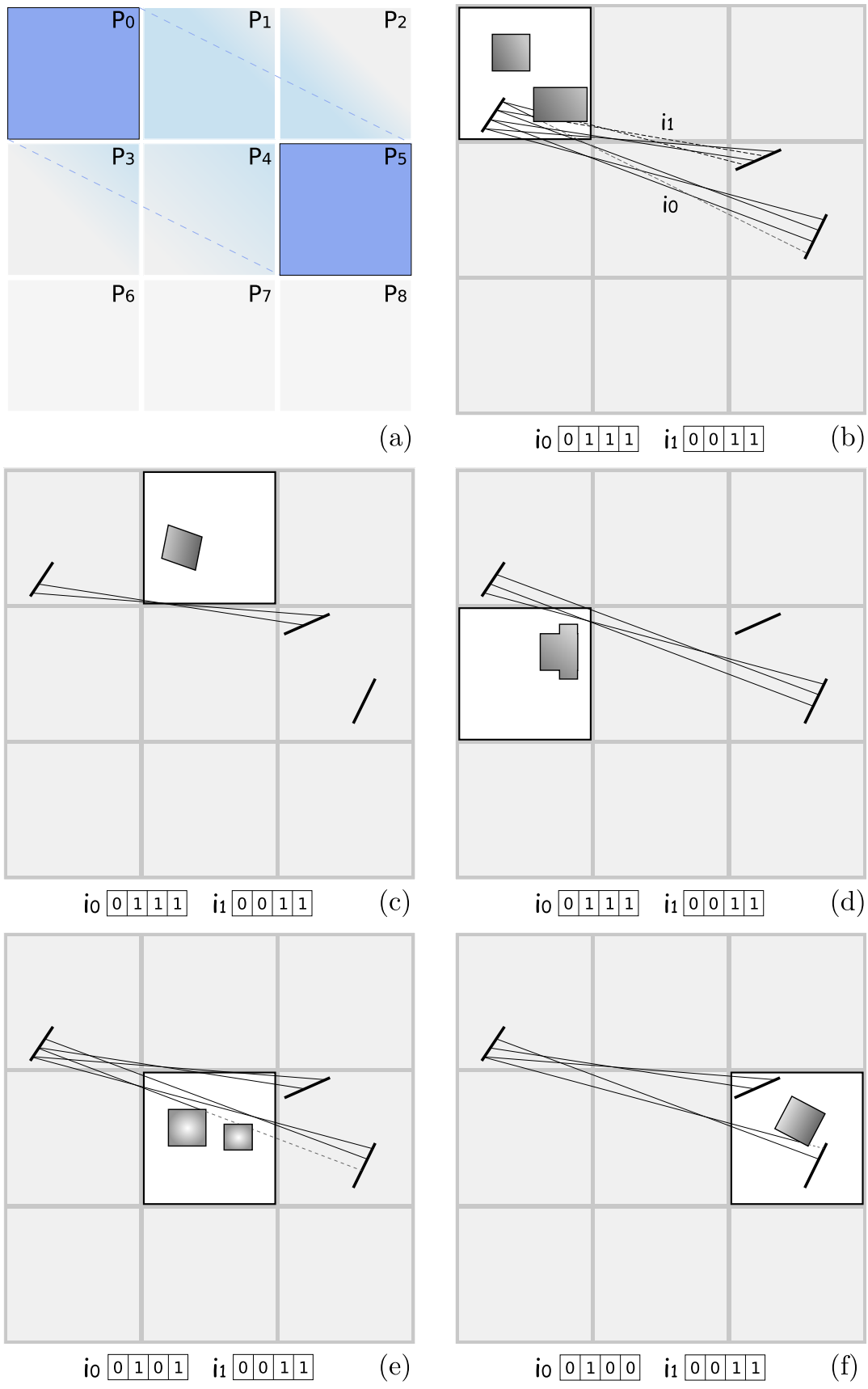


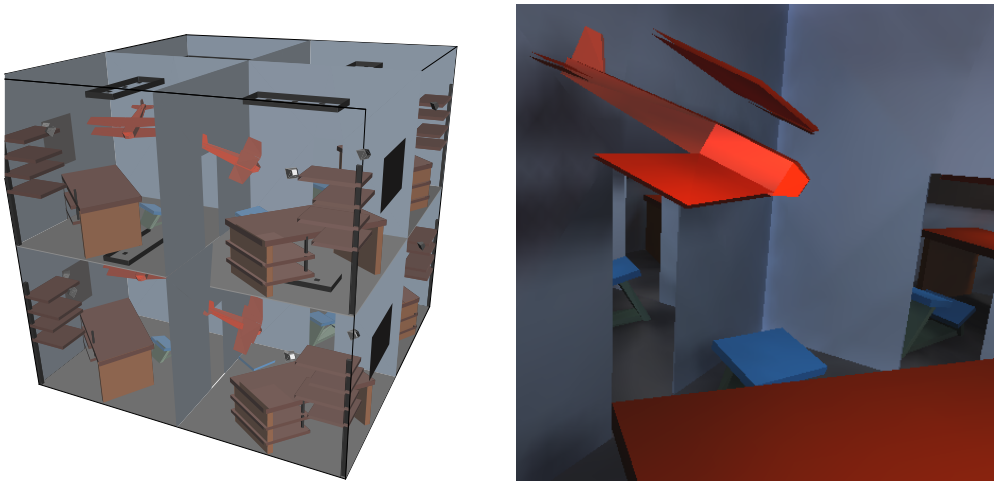
Figura 5.18: Ejemplo del cálculo distribuido de visibilidad.

64 procesadores Itanium2 a 1,5 GHz cada uno, conectados mediante una red Infiniband. Cada uno de los dos servidores que lo componen constituye, por tanto, un sistema de memoria compartida. En nuestra pruebas no se ha hecho uso de la red de comunicación entre servidores, al haber utilizado únicamente procesadores de uno de los dos nodos que lo componen (*superdome*).

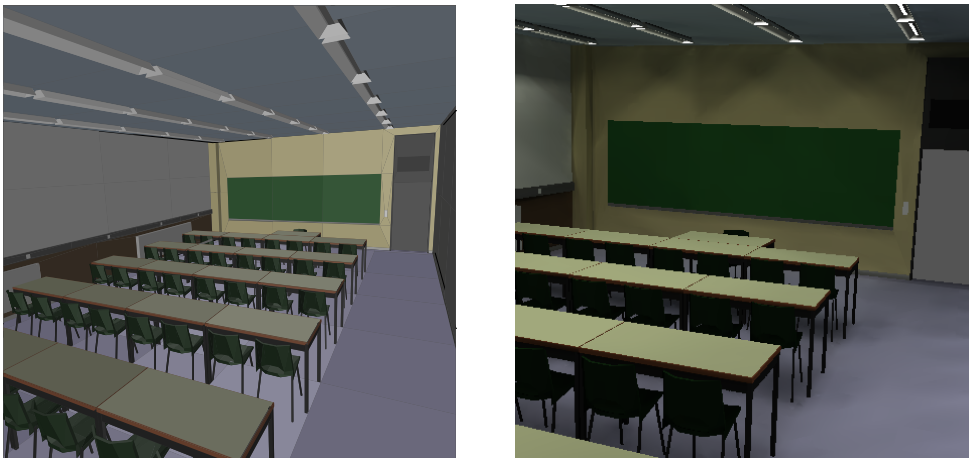
- Un multicomputador formado por un *cluster* de 16 nodos monoprocesador, con procesadores Pentium III a 800 MHz, conectados con una red Myrinet (*svg*).
- Otro multicomputador, en esta ocasión un *cluster* de 8 nodos biprocesador Intel Xeon IV a 1.8 GHz conectados mediante una red SCI (*muxia*).

Las tres escenas de prueba utilizadas para testear el rendimiento de nuestro método se muestran en la Figura 5.19. La primera de ellas, *Habitación* (Figura 5.19a), dispone la habitación clásica de Hanrahan en una configuración de 8 habitaciones, todas ellas idénticas. Se han añadido huecos a modo de puerta en algunas de las paredes entre las habitaciones, de forma que las diferentes estancias no queden por completo aisladas entre sí. El número de triángulos iniciales de la escena es de 3.172, obteniéndose, tras la aplicación del método de radiosidad, un total de 117.184 triángulos. La segunda escena, *Clase* (Figura 5.19b), consiste en un aula con, básicamente, pupitres y sillas, para un total de 13.438 triángulos de partida. El cálculo de la iluminación produce una escena de salida con 44.953 triángulos. En cuanto a la tercera escena, *Armadillos* (Figura 5.19c), presenta una disposición totalmente regular de figuras complejas, concretamente 16, en una habitación muy simple (solo se añaden los puntos de luz correspondientes, junto con las paredes). El número de triángulos de la escena de entrada es de 14.881, y el de la escena iluminada de 198.385.

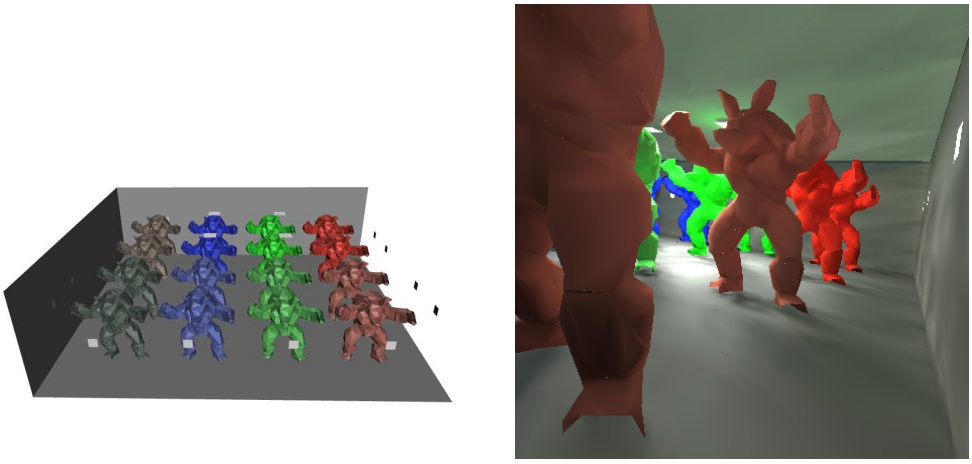
En las Tablas 5.4, 5.5 y 5.6 se muestran todos los tiempos de ejecución, en segundos, obtenidos para las tres escenas sobre las plataformas de prueba. La primera columna de cada tabla muestra el tiempo puro del método secuencial para el cálculo de la radiosidad en la escena, sin la introducción de ningún tipo de sobrecarga adicional debido a la paralelización. El resto de las columnas indica los tiempos para cada una de las combinaciones de procesadores probadas en cada máquina, con hasta 8 procesadores en *muxia* y hasta 16 procesadores en *superdome* y *svg*. Los resultados en forma de aceleración (*speed-up*), por su parte, se presentan en las gráficas de la Figura 5.20. En ambos casos, tablas de tiempos y gráficas de aceleración, aparece dos trazas de ejecución diferentes, etiquetadas como *mínimo* y *máximo* (*min* y *MAX* en las tablas de tiempos). Como vemos, los valores mínimos proporcionan



(a)



(b)



(c)

Figura 5.19: Escenas de prueba utilizadas: (a) *Habitación* (b) *Clase* (c) *Armadillos*.

una aceleración que se mantiene aproximadamente en la mitad del rendimiento ideal para cada combinación de procesadores. Estos valores se han obtenido sin aplicar ningún tipo de planificación a los dos hilos utilizados en cada procesador, lo que provoca que el porcentaje de CPU asignado por el sistema al hilo auxiliar sea equivalente al utilizado por el hilo principal, no correspondiéndose con el valor real de CPU que realmente el hilo auxiliar ha aprovechado. Las diversas pruebas realizadas nos confirman que hay un alto grado de CPU desaprovechada, y que obligaría a la implementación de algún tipo de planificación a nivel de hilo que nos permita acercarnos al máximo teórico que presentamos en las gráficas etiquetado como *máximo*. Este valor corresponde a la estimación para una ejecución en la que los dos hilos de cada proceso actuarían de forma totalmente concurrente, aprovechándose del paralelismo que es posible conseguirse a nivel de unidades funcionales en una única CPU [89], con lo que el tiempo empleado por el hilo auxiliar para resolver peticiones de visibilidad remotas se solaparía prácticamente por completo con el tiempo del hilo principal de ejecución.

En cualquier caso, los resultados muestran que el método paralelo desarrollado es perfectamente escalable y apropiado para distintos tipos de plataformas, al requerir un número pequeño de comunicaciones, aunque de tamaño considerable cada una de ellas, lo que lo hace menos sensible a la latencia de la red de interconexión utilizada por el sistema sobre el que se ejecuta. Además, la separación de tareas a nivel de hilo dentro de un mismo procesador nos permite controlar mejor la granularidad de las mismas que con el método de encuesta normalmente empleado. El método también se comporta de forma estable para las tres escenas de prueba utilizadas.

Como trabajo futuro a considerar sobre esta aproximación paralela al método de radiosidad jerárquica, estaría, por un lado, el estudio y mejora de la concurrencia de los distintos hilos de ejecución, de forma que se aprovechen al máximo los recursos de cada CPU, maximizando la ejecución en paralelo que la granularidad a nivel de hilo nos permite. Queda también pendiente, por otro lado, la aplicación de técnicas de *clustering* a nuestra implementación paralela, lo que permitirá reducir el número de comunicaciones entre los procesadores correspondientes al cálculo de las interacciones iniciales, aumentando de forma significativa el tamaño de las escenas de entrada sobre las que el algoritmo podría trabajar.

Tabla 5.4: Tiempos de ejecución para la escena *Habitación* (segundos).

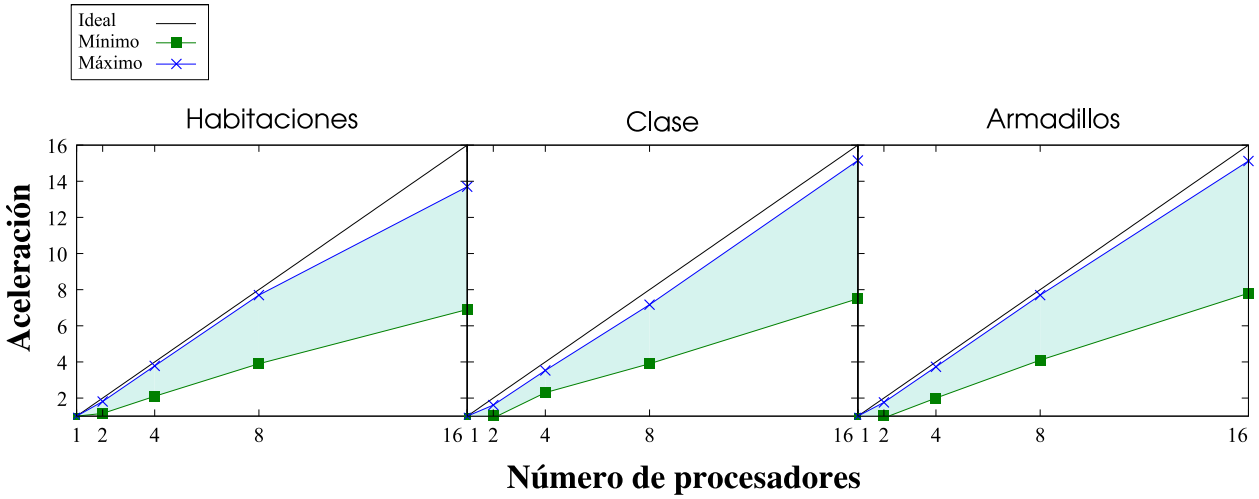
	Sec	2 Proc		4 Proc		8 Proc		16 Proc	
		min	MAX	min	MAX	min	MAX	min	MAX
superdome	280	243.5	154	133.3	74	72	36.5	40.5	20.5
svg	295	244	183	164	81	79	40	45.5	23
muxia	195	177	114	102.5	53.5	52	27	-	-

Tabla 5.5: Tiempos de ejecución para la escena *Clase* (segundos).

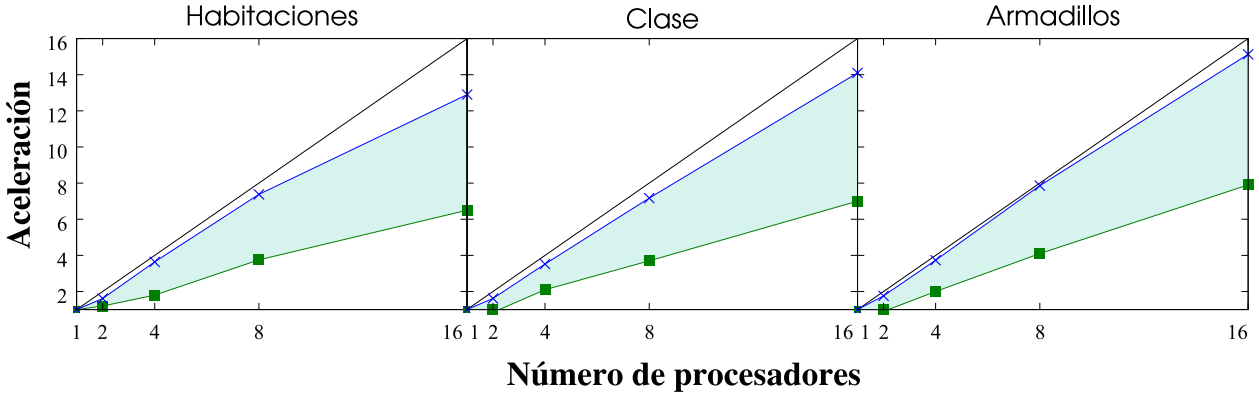
	Sec	2 Proc		4 Proc		8 Proc		16 Proc	
		min	MAX	min	MAX	min	MAX	min	MAX
superdome	6078	5936	3760	2642	1719	1558	848	810	401
svg	6375	6018	3945	3035	1809	1722	888	910	453
muxia	4435	4215	2744	2463	1255	1199	618	-	-

Tabla 5.6: Tiempos de ejecución para la escena *Armadillos* (segundos).

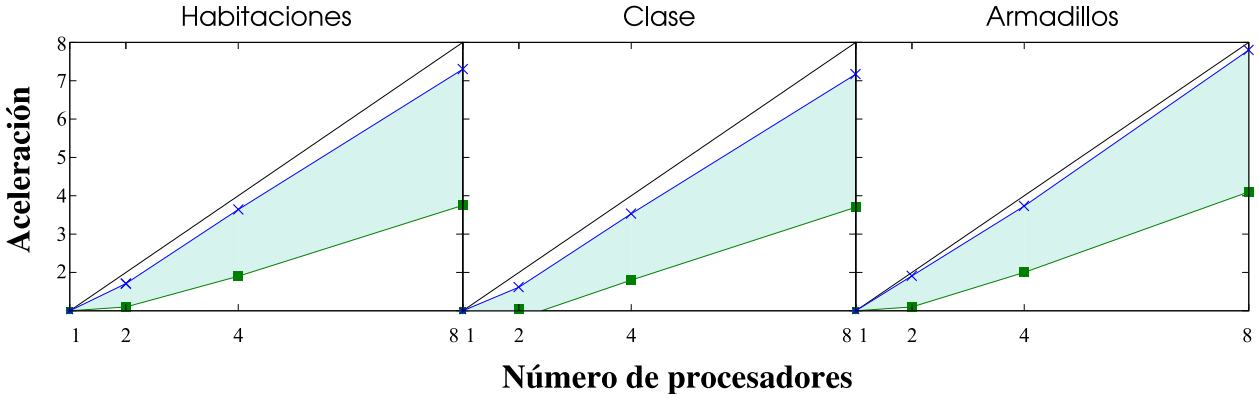
	Sec	2 Proc		4 Proc		8 Proc		16 Proc	
		min	MAX	min	MAX	min	MAX	min	MAX
superdome	7140	7079	4040	3570	1910	1741	930	916	472
svg	7490	7124	4240	3745	2005	1826	952	950	495
muxia	5160	4690	2690	2580	1382	1258	655	-	-



(a)



(b)



(c)

Figura 5.20: Aceleración obtenida para las escenas de entrada sobre: (a) sistema superdome (b) *cluster* svg (c) *cluster* muxía.

Conclusiones y principales aportaciones

Los modelos de iluminación global basados en radiosidad permiten obtener representaciones extraordinariamente realistas de escenas sintéticas, al simular de forma fiel el comportamiento físico real de la luz. Concretamente, la simulación efectiva de la componente difusa del reflejo de la luz en su interacción con una superficie difícilmente puede ser obtenida en la actualidad, con un coste razonable, mediante ningún otro modelo de iluminación global. El objetivo de este trabajo ha sido el desarrollo de distintas alternativas que den solución a los cuellos de botella más habituales en el cálculo de la iluminación global de una escena mediante el método de radiosidad jerárquica, según eran enumerados en la Sección 1.3.1.

Las principales aportaciones de este trabajo se pueden resumir en los siguientes puntos:

- Se han analizado las principales técnicas utilizadas para la aceleración del proceso de determinación del grado de visibilidad entre polígonos durante el cálculo de la iluminación global de una escena. Se han implementado y contrastado los algoritmos más relevantes, utilizando diferentes tipos de escenas de entrada que nos han permitido compararlos, analizando las ventajas y desventajas de cada uno de ellos.

- Tras comprobar que los métodos basados en la subdivisión del espacio ofrecen, en general, los mejores resultados, se ha propuesto un nuevo método, Método de Coherencia Angular de los Rayos (*ACR*). Nuestro método combina la subdivisión espacial con el aprovechamiento de los principios de coherencia y localidad en el espacio de direcciones de los rayos lanzados para comprobar la visibilidad entre dos objetos. Se presentan dos variantes del método ACR, una con un solo nivel de subdivisión del espacio de direcciones y otra adaptativa

de dos niveles. El método adaptativo permite lograr los mismos resultados que el primero reduciendo el consumo en memoria del método, permitiendo así la aplicación de particiones angulares más finas. El rendimiento y el error introducidos en ambos casos es analizado y comparado respecto a la simple aplicación de un método de subdivisión de superficies tradicional, logrando una mejora importante en tiempo de ejecución con un error introducido bastante pequeño.

- Se han descrito dos de los métodos de subdivisión recursiva más utilizados para mallas de triángulos, *Loop* y *Butterfly*, proponiendo una paralelización que permite el aprovechamiento de sistemas multiprocesador para la aceleración del proceso de subdivisión de objetos complejos. El diseño propuesto parte de la creación de grupos de triángulos mediante la técnica de agrupamiento *grouping*, ya existente. Así, la granularidad con la que la carga computacional es repartida entre los distintos procesadores del sistema se realiza a nivel de grupo. Primero se realiza un reparto estático inicial de los grupos de triángulos entre los procesadores. Durante el proceso iterativo de subdivisión, para mantener equilibrada la carga computacional de los diferentes procesadores, evitando esperas y minimizando el tiempo que los procesadores permanecen desocupados, se lleva a cabo un esquema de reparto dinámico de la carga. Los resultados obtenidos muestran un gran aprovechamiento del número de procesadores utilizado para la subdivisión.

- Enlazando con el punto anterior, hemos desarrollado dos métodos de radiosidad jerárquica basados en la subdivisión de superficies como aproximación al uso de mallas multiresolución. Ambos métodos se basan en el hecho de que la utilización de modelos geométricamente complejos en el proceso de cálculo de la radiosidad de una escena supone un coste insostenible respecto a la aportación real de ese nivel de detalle en la iluminación obtenida:

  - Nuestro primer método utiliza modelos sencillos que son iluminados mediante el método de radiosidad jerárquica. A continuación, estos modelos son substituidos por modelos detallados aplicando un esquema de subdivisión de superficies hasta el nivel deseado. Los colores obtenidos con la iluminación del objeto sencillo son asignados a la malla detallada, obteniendo un acabado muy semejante al que se lograría con el objeto detallado con una reducción drástica del tiempo necesario.
  - El segundo método de radiosidad basado en multiresolución utiliza una

selección de distintos vectores normales junto a los modelos gruesos, permitiendo aproximar más, con un pequeño coste adicional en el cálculo de la iluminación, los colores obtenidos al caso en el que se utilizaran los modelos complejos directamente. El resultado es un error todavía menor al del caso anterior con apenas un pequeño coste adicional en el método de radiosidad, derivado del cálculo de las transferencias de energía en las direcciones de las distintas normales utilizadas.

- La ejecución concurrente del proceso de iluminación de una escena en varias máquinas a la vez también ha sido tratada en este trabajo. Así, se han diseñado e implementado dos método paralelos de radiosidad jerárquica diferentes, ambos orientados a permitir su ejecución sobre un número lo más variado posible de plataformas paralelas. Con este fin, se ha utilizado un paradigma de memoria distribuida, con la utilización de una librería de paso de mensajes para la comunicación entre los procesadores del sistema.

  - La primera de nuestras paralelizaciones permite mantener equilibrada la carga computacional entre los procesadores, resolviendo el problema intrínseco que plantea el refinado adaptativo aplicado en el método de radiosidad jerárquico, que impide un conocimiento previo a su cálculo del nivel de detalle con el que será refinada cada transferencia de energía entre dos elementos de la escena. Con este método obtenemos buenos resultados en términos de aceleración para escenas de un tamaño relativamente pequeño, pero tiene el gran inconveniente de mantener réplicas de la escena de entrada inicial en la memoria de cada procesador, lo que impide el procesado de escenas de tamaño mayor a la memoria de uno de los procesadores.
  - La segunda implementación paralela supera la limitación del primer planteamiento, al ser la escena totalmente repartida entre todos los procesadores. Esta aproximación es utilizada a menudo en la paralelización del método de radiosidad progresiva, pero no es frecuente su aplicación en el caso del refinado jerárquico. La jerarquía adaptativa construida por el método de radiosidad jerárquica complica la tarea de mantener y distribuir la información de iluminación entre objetos presentes en diferentes procesadores, sobre todo en cuanto al número de comunicaciones necesarias si no se quiere aproximar en exceso la determinación de visibilidad entre polígonos en distintos procesadores.

- Se ha planteado un nuevo método para el cálculo de la visibilidad entre objetos en un entorno distribuido, utilizando mapas de bits. En nuestra método, el nivel de precisión con el que se determina la visibilidad entre polígonos de diferentes subespacios es el mismo que en el caso de interacciones entre polígonos locales a un mismo subespacio.
- Para minimizar el número de consultas de visibilidad entre procesadores se ha implementado una variante del refinado jerárquico tradicional, a la que hemos denominado refinado perezoso.
- En la implementación paralela con distribución de la escena se ha utilizado, además, una aproximación multi-hilo, que permite un aumento y mejor control de la concurrencia entre cálculo local y atención de peticiones de visibilidad remotas en cada procesador. Los resultados obtenidos demuestran que esta aproximación es válida tanto para su ejecución sobre distintas arquitecturas multiprocesador, como para el procesado de escenas de entrada de distinto tipo y tamaño.

Bibliografía

[1] M. Abrash. *Michael Abrash's graphics programming black book, special edition.* The Coriolis Group, Inc., 1997. Citado en p. 31

[2] M. Amor, M. Bóo, J. Hirche, M. Doggett, and W. Straßer. A meshing scheme for efficient hardware implementation of butterfly subdivision surfaces using displacement mapping. *IEEE Computer Graphics and Applications*, 25(2), 2005. Citado en p. 58, 61, 67, 70, 94, 96

[3] M. Amor, M. Bóo, E. J. Padrón, and D. Bartz. Hardware oriented algorithms for rendering order-independent transparency. *The Computer Journal (Aceptado, pendiente de publicación)*, 2006. Citado en p. 21

[4] M. Amor, E. J. Padrón, J. Touriño, and R. Doallo. Scheduling of a hierarchical radiosity algorithm on a distributed–memory multiprocessor. In *Proc. of the 4th International Meeting on Vector and Parallel Processing (VECPAR 2000)*, pages 581–591, 2000. Citado en p. 117, 121, 129

[5] M. Amor, J. R. Sanjurjo, E. J. Padrón, and R. Doallo. Progressive radiosity method on clusters using a new clipping algorithm. *Int. J. High Performance Computing and Networking (IJHPCN)*, 1(1/2/3):55–63, 2004. ISSN 1740-0562. Citado en p. 47, 115, 129, 148

[6] M. Amor, J. R. Sanjurjo, E. J. Padrón, A. P. Guerra, and R. Doallo. A scene partitioning method for global illumination simulations on clusters. In *Proc. of the 2nd Workshop on Hardware/Software Support for High Performance Scientific and Engineering Computing (SHPSEC 2003)*, pages S3/1–S3/17, 2003. Citado en p. 115

[7] S. Ar, G. Montag, and A. Tal. Self-organizing bsp trees. In *Computer Graphics Forum (Eurographics'02 Proc.)*, volume 21, pages 269–278, 2002. Citado en p. 30

[8] C. Armstrong, A. Bowyer, S. Cameron, J. Corney, G. Jared, R. Martin, A. Middleditch, M. Sabin, and J. Salmon. *Djinn: A geometric interface for solid modelling*. Information Geometers, 2000. Citado en p. 26

[9] B. Arnaldi, T. Priol, L. Renambot, and X. Pueyo. Visibility masks for solving complex radiosity computations on multiprocessors. In *Proc. First Eurographics Workshop on Parallel Graphics and Visualisation*, pages 219–232, 1996. Citado en p. 114, 148

[10] J. Arvo and D. Kirk. Fast ray tracing by ray classification. In *Computer Graphics (SIGGRAPH'87 Proc.)*, volume 21, pages 55–64. ACM Press, 1987. Citado en p. 38

[11] J. Arvo and D. Kirk. *An introduction to ray tracing*, chapter A survey of ray tracing acceleration techniques, pages 201–262. Academic Press Ltd., 1989. Citado en p. 21, 23, 45

[12] P. Bekaert, L. Neumann, A. Neumann, M. Sbert, and Y. D. Willems. Hierarchical monte carlo radiosity. In G. Drettakis and N. Max, editors, *Rendering Techniques'98 (Proc. Eurographics Rendering Workshop '98)*, pages 259–268. Springer Wien, 1998. Citado en p. 13

[13] J. I. Benavides, G. Cerruela, G. P. Trabado, and E. L. Zapata. Fast scalable solution for the parallel hierarchical radiosity problem in distributed memory architectures. In *Proc. Second Eurographics Workshop on Parallel Graphics and Visualisation*, pages 49–60, 1998. Citado en p. 116, 146

[14] S. Bischoff, L. P. Kobbelt, and H.-P. Seidel. Towards hardware implementation of loop subdivision. In *SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, pages 41–50, 2000. Citado en p. 61

[15] J. Bittner and P. Wonka. Visibility in computer graphics. *Journal of Environment and Planning B: Planning and Design*, pages 729–756, September 2003. Citado en p. 21

[16] M. R. Bolin and G. W. Meyer. An error metric for monte carlo ray tracing. In J. Dorsey and P. Slusallek, editors, *Rendering Techniques'97 (Proc. Eighth Eurographics Workshop on Rendering)*, pages 57–68. Springer Wien, 1997. Citado en p. 24

[17] M. Bóo, M. Amor, M. Doggett, J. Hirche, and W. Straßer. Hardware support for adaptive subdivision surface rendering. In *HWWS '01: Proc. ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 33–40, 2001. Citado en p. 58, 61, 67, 70, 94, 96

[18] A. Bowyer and J. Woodwark. *Introduction to Computing with Geometry.* Information Geometers Ltd, 1993. Citado en p. 26

[19] I. Buck and T. Purcell. *GPUGems: Programming techniques, tips and tricks for real-time graphics*, chapter 37, A toolkit for computation on GPUs. Addison-Wesley, 2004. Citado en p. 14

[20] M. Cammarano and H. W. Jensen. Time dependent photon mapping. In *Proc. 13th Eurographics Workshop on Rendering*, 2002. Citado en p. 8

[21] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, 1978. Citado en p. 60

[22] E. E. Catmull. *A subdivision algorithm for computer display of curved surfaces.* PhD thesis, University of Utah, 1974. Citado en p. 21

[23] G. Cerruela. *Radiosidad en multiprocesadores.* PhD thesis, Univ. Málaga, 1999. Citado en p. 113

[24] A. G. Chalmers and D. J. Paddon. Implementing a radiosity method using a parallel adaptive system. In *Proc. First International Conference on Applications of Transputers*, 1989. Citado en p. 113

[25] S. E. Chen. A progressive radiosity method and its implementation in a distributed processing environment. Master's thesis, Program of Computer Graphics, Cornell University, Ithaca, NY, 1989. Citado en p. 113

[26] P. H. Christensen, D. Lischinski, E. J. Stollnitz, and D. H. Salesin. Clustering for glossy global illumination. *ACM Transactions on Graphics*, 16(1):3–33, 1997. Citado en p. 27, 58, 85, 88

[27] J. H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, 1976. Citado en p. 25

[28] M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A progressive refinement approach to fast radiosity image generation. In *Computer Graphics (SIGGRAPH'88 Proc.)*, volume 22, pages 75–84, 1988. Citado en p. 12

[29] R. L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, 1986. Citado en p. 8

[30] G. Coombe and M. Harris. *GPUGems 2: Programming techniques for high-performance graphics and general-purpose computation*, chapter 39, Global illumination using progressive refinement radiosity. Addison-Wesley, 2005. Citado en p. 13, 14

[31] G. Coombe, M. Harris, and A. Lastra. Radiosity on graphics hardware. In *Proc. 2004 Conference on Graphics Interface*, pages 161–168, 2004. Citado en p. 13, 14

[32] N. D. Cuong. An exploration of coherence-based acceleration methods using the ray tracing kernel g/gx. Technical report, TU-Dresden, Germany, 1997. Citado en p. 23, 38

[33] C. Damez, N. Holzschuch, and F. X. Sillion. Space-time hierarchical radiosity with clustering and higher-order wavelets. *Computer Graphics Forum*, 23(2), 2004. Citado en p. 86

[34] A. del Río Fernández. Análisis de la implementación del algoritmo de loop para la subdivisión de mallas de triángulos. Master's thesis, Universidade de Santiago de Compostela, 2001. Citado en p. 60

[35] D. Doo and M. Sabin. Behavior of recursive subdivision surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, 1978. Citado en p. 60

[36] R. Dumont and K. Bouatouch. Combining hierarchical radiosity and LODs. In *Proc. IASTED International Conference on Computer Graphics and Imaging*, 2000. Citado en p. 86, 89, 90

[37] N. Dyn, D. Levin, and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, 1990. Citado en p. 60

[38] F. J. Espino, M. Bóo, M. Amor, and J. D. Bruguera. Adaptive tessellation of NURBS surfaces. In *Journal of WSCG (Proc. 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'03))*, volume 1, 2003. Citado en p. 96

[39] M. Feda and W. Purgathofer. Progressive refinement radiosity on a transputer network. In P. Brunet and F. W. Jansen, editors, *Photorealistic Rendering in Computer Graphics (Proc. Second Eurographics Workshop on Rendering*, pages 139–148. Springer-Verlag, 1994. Citado en p. 114

[40] C.-C. Feng and S.-N. Yang. A parallel hierarchical radiosity algorithm for complex scenes. In *Proc. Third Parallel Rendering Symposium (IEEE-ACM/SIGGRAPH) (PRS'97)*, pages 71–78, 1997. Citado en p. 116

[41] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer graphics: Principles and practice in C (2nd edition).* Addison-Wesley, 1995. Citado en p. 5

[42] H. Fuchs, Z. M. Kedem, and B. F. Naylor. Predetermining visibility priority in 3-d scenes (preliminary report). In *Computer Graphics (SIGGRAPH'79 Proc.)*, pages 175–181. ACM Press, 1979. Citado en p. 30, 31

[43] H. Fuchs, Z. M. Kedem, and B. F. Naylor. On visible surface generation by a priori tree structures. In *Computer Graphics (SIGGRAPH'80 Proc.)*, volume 14, pages 124–133. ACM Press, 1980. Citado en p. 29, 30, 31, 34, 35

[44] A. Fujimoto and T. Tanaka. Accelerated ray tracing. In T. Kunii, editor, *Computer Graphics: Visual Technology and Art (Proc. Computer Graphics Tokyo '85)*, pages 41–65. Springer-Verlag, 1985. Citado en p. 29

[45] T. A. Funkhouser. Coarse-grained parallelism for hierarchical radiosity using group iterative methods. In *Computer Graphics (SIGGRAPH'96 Proc.)*, volume 30, pages 343–352, 1996. Citado en p. 116

[46] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Computer Graphics (SIGGRAPH'97 Proc.)*, volume 31, 1997. Citado en p. 86, 90

[47] M. Garland, A. Willmot, and P. H. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proc. ACM Symposium on Interactive 3D Graphics*, 2001. Citado en p. 58, 86, 90

[48] M. Garland and Y. Zhou. Quadric-based simplification in any dimension. *ACM Transactions on Graphics*, 24(2), 2005. Citado en p. 86

[49] S. Gibson and R. J. Hubbold. Efficient hierarchical refinement and clustering for radiosity in complex environment. In *Computer Graphics Forum (Eurographics'96 Proc.)*, volume 15, pages 297–310, 1996. Citado en p. 19, 88

[50] S. Gibson and R. J. Hubbold. A perceptually-driven parallel algorithm for efficient radiosity simulation. *IEEE Transactions on Visualization and Computer Graphics*, 6(3):220–235, 2000. Citado en p. 114, 115

[51] A. S. Glassner. Space subdivision for fast ray tracing. *IEEE Computer Graphics and Applications*, 4(10):15–22, 1984. Citado en p. 29, 31

[52] A. S. Glassner, editor. *An introduction to ray tracing.* Academic Press Ltd., 1989. Citado en p. 7

[53] E. Gobbetti, L. Spanò, and M. Agus. Hierarchical higher order face cluster radiosity for global illumination walkthroughs of complex non-diffuse environments. *Computer Graphics Forum*, 22(3):563–572, 2003. Citado en p. 88

[54] J. Goldsmith and J. Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications*, 7(5):14–20, 1987. Citado en p. 27

[55] O. Good and Z. Taylor. Optimized photon tracing using spherical harmonic light maps. ACM SIGGRAPH'05 Sketch, 2005. Citado en p. 9

[56] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces. In *Computer Graphics (SIGGRAPH'84 Proc.)*, volume 18, pages 213–222, 1984. Citado en p. 11

[57] S. Gortler, M. F. Cohen, and P. Slusallek. Radiosity and relaxation methods. *IEEE Computer Graphics and Applications*, 14(6):48–58, 1994. Citado en p. 12

[58] S. J. Gortler, P. Schröder, M. F. Cohen, and P. Hanrahan. Wavelet radiosity. In *Computer Graphics (SIGGRAPH'93 Proc.)*, pages 221–230, 1993. Citado en p. 13

[59] G. Greger, P. Shirley, P. M. Hubbard, and D. P. Greenberg. The irradiance volume. *IEEE Computer Graphics and Applications*, 18(2):32–43, 1998. Citado en p. 89

[60] A. P. Guerra, M. Amor, J. Eiroa, E. J. Padrón, J. R. Sanjurjo, and R. Doallo. Método de radiosidad progresiva de alto rendimiento basado en el particionamiento de la escena. In *Actas de las XIII Jornadas de Paralelismo*, pages 251–256, 2002. Citado en p. 130

[61] A. P. Guerra, M. Amor, E. J. Padrón, and R. Doallo. A high–performance progressive radiosity method based on scene partitioning. In J. M. L. M. Palma, J. Dongarra, V. Hernández, and A. A. Sousa, editors, *High Performance Computing for Computational Science - VECPAR 2002*, volume 2565 of *Lecture Notes in Computer Science*, pages 537–548. Springer, 2003. Citado en p. 114, 130

[62] P. Guitton, J. Roman, and G. Subrenat. Implementation results and analysis of a parallel progressive radiosity. In *Proc. Second Parallel Rendering Symposium (IEEE-ACM/SIGGRAPH) (PRS'95)*, pages 31–38, 1995. Citado en p. 114

[63] E. A. Haines and J. R. Wallace. Shaft culling for efficient ray-traced radiosity. In *Photorealistic Rendering in Computer Graphics (Proc. Second Eurographics Workshop on Rendering)*, pages 122–138. Springer-Verlag, 1991. Citado en p. 38

[64] P. Hanrahan, D. Saltzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. In *Computer Graphics (SIGGRAPH'91 Proc.)*, pages 197–206, 1991. Citado en p. 14, 18, 47, 121, 142

[65] J.-M. Hasenfratz, C. Damez, F. Sillion, and G. Drettakis. A practical analysis of clustering strategies for hierarchical radiosity. In *Computer Graphics Forum (Eurographics'99 Proc.)*, volume 18, pages 221–232, 1999. Citado en p. 19, 26, 47, 58, 86, 88

[66] V. Havran. *Heuristic ray shooting algorithms*. PhD thesis, Faculty of Electrical Engineering, Czech Technical University, Praha, Czech Republic, 2000. Citado en p. 21, 23, 30

[67] P. S. Heckbert and P. Hanrahan. Beam tracing polygonal objects. In H. Christiansen, editor, *Computer Graphics (SIGGRAPH'84 Proc.)*, volume 18, pages 119–127, 1984. Citado en p. 24

[68] P. M. Hubbard. Interactive collision detection. In *Proc. IEEE Symposium on Research Frontiers in Virtual Reality*, pages 24–31, 1993. Citado en p. 26

[69] H. Igehy. Tracing ray differentials. In A. Rockwood, editor, *Computer Graphics (SIGGRAPH'99 Proc.)*, volume 18, pages 179–186. ACM Press/Addison-Wesley Publishing Co., 1999. Citado en p. 24

[70] A. James. *Binary space partitioning for accelerated hidden surface removal and rendering of static environments*. PhD thesis, University of East Anglia, 1999. Citado en p. 33, 35

[71] A. James and A. M. Day. Conflict neutralization on binary space partitioning. In *Computer Graphics Forum (Eurographics'00 Proc.)*, volume 19, pages 153–163, 2000. Citado en p. 33, 35

[72] F. W. Jansen. Data structures for ray tracing. In L. R. A. Kessener, F. J. Peters, and M. L. P. van Lierop, editors, *Data Structures for Raster Graphics (Eurographics Seminars)*, pages 57–73. Springer-Verlag, 1986. Citado en p. 31

[73] H. W. Jensen. *Realistic image synthesis using photon mapping*. AK Peters, 2001. Citado en p. 8

[74] J. T. Kajiya. The rendering equation. In *Computer Graphics (SIGGRAPH'86 Proc.)*, volume 20, pages 143–150, 1986. Citado en p. 5

[75] M. R. Kaplan. Space tracing: A constant time ray tracer. In *Computer Graphics (Tutorial on ray tracing, SIGGRAPH '85 Proc.)*, volume 19, 1985. Citado en p. 30, 31

[76] T. Kato. "kilauea": parallel global illumination renderer. *Parallel Comput.*, 29(3), 2003. Citado en p. 148

[77] T. L. Kay and J. T. Kajiya. Ray tracing complex scenes. In *Computer Graphics (SIGGRAPH'86 Proc.)*, volume 20, pages 269–278. ACM Press, 1986. Citado en p. 27

[78] A. Keller. Instant radiosity. In *Computer Graphics (SIGGRAPH'97 Proc.)*, pages 49–56, 1997. Citado en p. 13

[79] A. Keller. Strictly deterministic sampling methods in computer graphics. Technical report, mental images, 2001. Also in SIGGRAPH 2003 Monte Carlo Course Notes. Citado en p. 24

[80] S. Kircher and M. Garland. Progressive multiresolution meshes for deforming surfaces. In *ACM/Eurographics Symposium on Computer Animation*, pages 191–200, 2005. Citado en p. 86

[81] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998. Citado en p. 26

[82] A. J. F. Kok. Grouping of patches in progressive radiosity. In *Proc. Fourth Eurographics Workshop on Rendering*, pages 221–231, 1993. Citado en p. 26

[83] E. P. Lafortune. *Mathematical models and monte carlo algorithms for physically based rendering*. PhD thesis, Katholieke Universiteit Leuven, 1996. Citado en p. 8

[84] E. P. Lafortune and Y. D. Willems. Rendering participating media with bidirectional path tracing. In *Proc. 7th Eurographics Workshop on Rendering*, pages 92–101, 1996. Citado en p. 8

[85] M. E. Lee, R. A. Redner, and S. P. Uselton. Statistically optimized sampling for distributed ray tracing. In *Computer Graphics (SIGGRAPH'85 Proc.)*, volume 19, pages 61–68. ACM Press, 1985. Citado en p. 24

[86] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, 1987. Citado en p. 58, 60, 62

[87] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of detail for 3D graphics*. Morgan Kaufmann, 2002. Citado en p. 26

[88] V. C. H. Ma and M. D. McCool. Low latency photon mapping using block hashing. In *SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, pages 89–99, 2002. Citado en p. 9

[89] P. Marcuello and A. González. Thread partitioning and value prediction for exploiting speculative thread-level parallelism. *IEEE Transactions on Computer*, 53(2), 2004. Citado en p. 158

[90] I. Martín, X. Pueyo, and D. Tost. A two-pass hardware-based method for hierarchical radiosity. In *Computer Graphics Forum (Eurographics'98 Proc.)*, volume 17, pages 159–164, 1998. Citado en p. 13

[91] R. Martínez, M. Sbert, and L. Szirmay-Kalos. Parallel multipath with bundles of lines. Universitat de Girona, Spain, 2000. Citado en p. 117

[92] R. Martínez, L. Szirmay-Kalos, M. Sbert, and A. M. Abbas. Parallel implementation of stochastic iteration algorithms. In *Proc. Ninth International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'01)*, 2001. Citado en p. 117

[93] S. Melax. Dynamic plane shifting bsp traversal. In *Proc. 2000 Conference on Graphics Interface*, pages 213–220, 2000. Citado en p. 29, 30

[94] D. Meneveaux and K. Bouatouch. Synchronization and load balancing for parallel hierarchical radiosity of a complex scene on a heterogeneus computer network. In *Computer Graphics Forum (Eurographics'99 Proc.)*, volume 18, pages 201–212, 1999. Citado en p. 117

[95] D. Meneveaux, K. Bouatouch, G. Subrenat, and P. Blasi. Efficient clustering and visibility calculation for global illumination. In *Proc. 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa (AFRIGRAPH 2003)*, pages 87–94, 2003. Citado en p. 58, 85

[96] T. Möller and B. Trumbore. Fast, minimum storage ray-triangle intersection. *Journal of graphics tools*, 2(1):21–28, 1997. Citado en p. 23

[97] P. Morer, A. M. García-Alonso, and J. Flaquer. Optimization of a priority list algorithm for 3-d rendering of buildings. In *Computer Graphics Forum (Eurographics'95 Proc.)*, volume 14, pages 217–228, 1995. Citado en p. 33, 34

[98] G. Müller, S. Schäfer, and D. Fellner. Automatic creation of object hierarchies for radiosity clustering. *Computer Graphics Forum*, 19(4):213–221, 2000. Citado en p. 88

[99] D. Nalasco. Truform white paper. Technical report, ATI Technologies, 2001. Citado en p. 61

[100] B. Naylor, J. Amanatides, and W. Thibault. Merging bsp trees yields polyhedral set operations. In *Computer Graphics (SIGGRAPH'90 Proc.)*, volume 24, pages 115–124. ACM Press, 1990. Citado en p. 29, 30

[101] A.Ñ. Netravali and B. G. Haskell. *Digital pictures: Representation, compression and standards (2nd ed)*. Springer, 1995. Citado en p. 52, 109

[102] E. J. Padrón, M. Amor, M. Bóo, M. Arenaz, and R. Doallo. Subdivisión de superficies en tiempo real de síntesis de imágenes 3d. In *Actas de las XII Jornadas de Paralelismo*, pages 295–300, 2001. Citado en p. 65

[103] E. J. Padrón, M. Amor, M. Bóo, and R. Doallo. Efficient parallel implementations for surface subdivision. In *Proc. of the Fourth Eurographics Workshop on Parallel Graphics and Visualization*, pages 113–121, 2002. Citado en p. 65

[104] E. J. Padrón, M. Amor, M. Bóo, and R. Doallo. A meshing scheme for real time surface subdivision. *Journal of WSCG (Proc. 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'02))*, 10(2):349–356, 2002. Citado en p. 65

[105] E. J. Padrón, M. Amor, and R. Doallo. Improving the performance of visibility determination in global illumination methods. In *Proc. of the 2005 International Conference on Imaging Science, Systems, and Technology: Computer Graphics (CISST'05)*, pages 265–270, 2005. Citado en p. 22, 40

[106] E. J. Padrón, M. Amor, J. Touriño, and R. Doallo. Hierarchical radiosity on multicomputers: a load–balanced approach. In *Proc. of the 10th SIAM Conference on Parallel Processing for Scientific Computing*, pages 379–390, 2001. Citado en p. 117, 121, 129

[107] E. J. Padrón, M. Amor, J. Touriño, B. B. Fraguela, and R. Doallo. Implementación del método de radiosidad jerárquica sobre un multicomputador. In *Actas de las XI Jornadas de Paralelismo*, pages 317–322, 2000. Citado en p. 117, 121, 129

[108] E. J. Padrón, R. Troncoso, M. Amor, J. R. Sanjurjo, and R. Doallo. Estudio comparativo de algoritmos de visibilidad en los modelos de iluminación global. In *Actas del XIII Congreso Español de Informática Gráfica (CEIG 2003)*, pages 141–154, 2003. Citado en p. 22

[109] M. Price and G. Truman. Radiosity in parallel. In *Proc. First International Conference on Applications of Transputers*, pages 40–47, 1989. Citado en p. 113

[110] T. J. Purcell, C. Donner, M. Cammarano, H. W. Jensen, and P. Hanrahan. Photon mapping on programmable graphics hardware. In *SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, pages 41–50, 2003. Citado en p. 13

[111] R. E. M. Ramírez. *Adaptive and depth buffer solutions with bundles of parallel rays for global line monte carlo radiosity.* PhD thesis, Universitat Politècnica de Catalunya, 2004. Citado en p. 13

[112] E. Reinhard and F. W. Jansen. Hybrid scheduling for efficient ray tracing of complex images. In *High Performance Computing for Computer Graphics and Visualisation*, pages 78–87. Springer-Verlag, 1995. Citado en p. 8

[113] L. Renambot, B. Arnaldi, T. Priol, and X. Pueyo. Towards efficient parallel radiosity for dsm-based parallel computers using virtual interfaces. In *Proc. Third Parallel Rendering Symposium (IEEE-ACM/SIGGRAPH) (PRS'97)*, pages 79–86, 1997. Citado en p. 113

[114] J. Revelles and C. Ureña. A formalization of ray casting optimization techniques. In *XI Congreso Español de Informática Gráfica (CEIG'2001)*, pages 85–98, 2001. Citado en p. 24

[115] S. Roth. Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18(2):109–144, 1982. Citado en p. 21

[116] S. M. Rubin and T. Whitted. A 3-dimensional representation for fast rendering of complex scenes. In *Computer Graphics (SIGGRAPH'80 Proc.)*, volume 14, pages 110–116, 1980. Citado en p. 25

[117] H. E. Rushmeier, C. Patterson, and A. Veerasamy. Geometric simplification for indirect illumination calculations. In *Proc. Graphics Interface '93*, pages 227–236, 1993. Citado en p. 89

[118] J. R. Sanjurjo, M. Amor, M. Bóo, and R. Doallo. Parallel global illumination method based on a non-uniform partitioning of the scene. In *Proc. 13th Euromicro Conference on Parallel, Distributed and Network Based Processing (PDP'05)*, pages 251–257, 2005. Citado en p. 115, 129, 146

[119] M. Sbert, X. Pueyo, L. Neumann, and W. Purgathofer. Global multipath monte carlo algorithms for radiosity. *The Visual Computer*, 12(2):47–61, 1996. Citado en p. 13

[120] K. Schloegel, G. Karypis, and V. Kumar. *Sourcebook of parallel computing*, chapter Graph partitioning for high-performance scientific simulations, pages 491–541. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000. Citado en p. 128

[121] P. Schröder and D. Zorin. Subdivision for modeling and animation. In *SIGGRAPH 2000 Course Notes*, 2000. Citado en p. 62

[122] R. A. Schumacker, P. Brand, M. Gilliland, and W. Sharp. Study for applying computer-generated images to visual simulation. Technical Report AFHRL-TR-69-14, U.S. Air Force Human Resources Laboratory, 1969. Citado en p. 30

[123] A. Sharpe, M. Hampton, S. Nirenstein, J. Gain, and E. Blake. Accelerating ray shooting through aggressive 5d visibility preprocessing. In *Proc. 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa (AFRIGRAPH 2003)*, volume 30, pages 95–100, 2003. Citado en p. 38

[124] P. Shirley. Hybrid radiosity/monte carlo methods. In *SIGGRAPH'94 Advanced Radiosity Course*, 1994. Citado en p. 8

[125] L.-J. Shiue, I. Jones, and J. Peters. A realtime GPU subdivision kernel. In *Computer Graphics (SIGGRAPH'05 Proc.)*, volume 24, pages 1010–1015, 2005. Citado en p. 61, 96

[126] F. Sillion and J.-M. Hasenfratz. Efficient parallel refinement for hierarchical radiosity on a dsm computer. In *Proc. Third Eurographics Workshop on Parallel Graphics and Visualization*, pages 61–74, 2000. Citado en p. 116

[127] F. X. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):240–254, 1995. Citado en p. 19, 27, 58, 85, 86, 88

[128] G. Simiakakis. *Accelerating ray tracing with directional subdivision and parallel processing*. PhD thesis, University of East Anglia, 1995. Citado en p. 38

[129] J. P. Singh. *Parallel hierarchical N-body methods and their implications for multiprocessors*. PhD thesis, Stanford University, 1993. Citado en p. 13, 86

[130] J. P. Singh, A. Gupta, and M. Levoy. Parallel visualization algorithms: Performance and architectural implications. *IEEE Computer Graphics and Applications*, pages 45–55, 1994. Citado en p. 116

[131] B. Smits, J. Arvo, and D. Greenberg. A clustering algorithm for radiosity in complex environments. In *Computer Graphics (SIGGRAPH'94 Proc.)*, pages 435–442, 1994. Citado en p. 19, 26, 27, 58, 86, 88

[132] J. Stamp and C. Loop. Quad/triangle subdivision. *Computer Graphics Forum (Eurographics'03 Proc.)*, 22(1):1–7, 2003. Citado en p. 60

[133] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*. Springer-Verlag, 1980. Citado en p. 12

[134] W. Straßer. *Schnelle Kurven- und Flächendarstellung auf graphischen Sichtgeräten*. PhD thesis, Technische Universität Berlin, 1974. Citado en p. 21

[135] W. Stüuerzlinger and C. Wild. Parallel progressive radiosity with parallel visibility computations. In *Proc. Winter School of Computer Graphics and CAD Systems '94 (WSCG'94)*, pages 66–74, 1994. Citado en p. 113

[136] E. E. Sutherland, R. F. Sproull, and R. A. Schumacker. A characterization of ten hidden-surface algorithms. *ACM Computing Surveys (CSUR)*, 6(1):1–55, 1974. Citado en p. 30

[137] W. C. Thibault and B. F. Naylor. Set operations on polyhedra using binary space partitioning trees. In *Computer Graphics (SIGGRAPH'87 Proc.)*, volume 21, pages 153–162. ACM Press, 1987. Citado en p. 30

[138] T. Ulrich. *Game programming gems*, chapter Loose octrees, pages 444–453. Charles River Media, 2000. Citado en p. 29

[139] G. van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools: JGT*, 2(4):1–14, 1997. Citado en p. 26

[140] A. Vlachos, J. Peters, C. Boyd, and J. L. Mitchell. Curved pn triangles. In *SI3D '01: Proc. 2001 symposium on Interactive 3D graphics*, pages 159–166. ACM Press, 2001. Citado en p. 73

[141] J. R. Wallace, M. F. Cohen, and D. P. Greenberg. A two-pass solution to the rendering equation: a synthesis of ray tracing and radiosity methods. In *Computer Graphics (SIGGRAPH'87 Proc.)*, volume 21, pages 311–320. ACM Press, 1987. Citado en p. 8

[142] J. Warren and S. Schaefer. A factored approach to subdivision surfaces. *IEEE Computer Graphics and Applications*, 24(3):74–81, 2004. Citado en p. 60, 62, 92

[143] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, 1980. Citado en p. 21, 23, 25

[144] A. Willmott, P. Heckbert, and M. Garland. Face cluster radiosity. In *Proc. Tenth Eurographics Workshop on Rendering*, 1999. Citado en p. 58, 86, 89, 90

[145] C. M. Wittenbrink. R-Buffer: a pointerless A-Buffer hardware architecture. In *HWWS '01: Proc. ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics hardware*, pages 73–80, 2001. Citado en p. 21

[146] D. Zareski, B. Wade, P. Hubbard, and P. Shirley. Efficient parallel global illumination using density estimation. In *Proc. Second Parallel Rendering Symposium (IEEE-ACM/SIGGRAPH) (PRS'95)*, pages 47–54, 1995. Citado en p. 116

[147] H. R. Zatz. Galerkin radiosity: a higher order solution method for global illumination. In *Computer Graphics (SIGGRAPH'93 Proc.)*, pages 213–220, 1993. Citado en p. 13

[148] D. Zorin. *Stationary subdivision and multiresolution surface representations*. PhD thesis, California Institute of Technology, 1997. Citado en p. 60

[149] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Computer Graphics (SIGGRAPH'96 Proc.)*, pages 189–192, 1996. Citado en p. 58, 60, 63