Informe Técnico
DLSI 01/11/2007

Simplificación por Regiones usando Qslim

Pablo Prades, José Ribelles

Departamento de Lenguajes y Sistemas Informáticos

Correo electrónico: ribelles@lsi.uji.es

Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I
Campus de Riu Sec
E-12071 Castellón, España

Region Simplification using Qslim

Pablo Prades, José Ribelles

Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I
Campus de Riu Sec
E-12071 Castellón, España

E-mail:ribelles@lsi.uji.es

Abstract:

This technical report describes an extension for Qslim which implements a region based simplification algorithm. Qslim is an automatic polygonal surface simplification program developed by Michael Garland. It implements his iterative edge contraction based simplification method. The exposed extension allows Qslim to delete a group of edges instead of only one in each iteration. The affected group of triangles comprises a region of the object surface. Thus, the new simplification process generates an approximation in each iteration which differs from the previous one only on the region obtained in that iteration. The size of the region is defined by the number of deleted triangles. This number is given by the user as a parameter in the command line at execution time of the program and it is a constant along the whole process.

Keywords:

Polygonal surface, polygonal model, polygonal mesh, level of detail, multiresolution model, iterative simplification, edge collapse, region simplification.

Simplificación por Regiones usando Qslim

Pablo Prades, José Ribelles

Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I
Campus de Riu Sec
E-12071 Castellón, España

E-mail:ribelles@lsi.uji.es

Resumen:

Este informe describe una extensión del *Qslim* que implementa un algoritmo de simplificación basado en regiones. El *Qslim* es un programa de simplificación automática de superficies poligonales desarrollado por M. Garland, implementación de su método de simplificación basado en la contracción iterativa de aristas. La ampliación que aquí se presenta va a permitir que el *Qslim* elimine en cada iteración del proceso de simplificación no una única arista sino un conjunto de ellas de manera que el conjunto de triángulos afectados por la eliminación de dichas aristas conforman una región de la superficie del objeto. De esta manera, en cada iteración del proceso de simplificación, se obtiene una nueva aproximación donde la diferencia con su antecesora es únicamente la región obtenida para esa iteración. El tamaño de la región se define como el número de triángulos eliminados, que es constante a lo largo de todo el proceso, y es dado por el usuario como un parámetro más en la línea de comandos del *Qslim*.

Palabras clave:

Superficie poligonal, modelo poligonal, malla poligonal, nivel de detalle, modelo multirresolución, simplificación iterativa, contracción de aristas, región de simplificación.

Simplificación por Regiones usando Qslim

P. Prades, J. Ribelles

Departamento de Lenguajes y Sistemas Informáticos

Universitat Jaume I, 12071 Castellón, Spain
e − mail: {ribelles}@lsi.uji.es
Phone: +34-96-472-83-18, Fax: +34-96-472-84-35

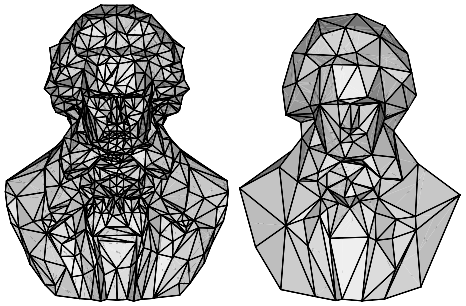
1. INTRODUCCIÓN

Hoy en día, la mayoría de las aplicaciones basadas en gráficos por ordenador y campos relacionados utilizan superficies poligonales para la simulación y visualización de objetos [2]. No es de extrañar que los algoritmos de simplificación automática de superficies poligonales se hayan hecho tan prolíferos. Éstos son idóneos para generar el nivel de detalle que más se adecue a los requerimientos visuales de una aplicación [12]. En el ámbito del modelado de objetos, se denomina nivel de detalle a la malla que representa a un objeto en una aplicación bajo unas determinadas circunstancias. Habitualmente, estas circunstancias tienen que ver con la distancia a la que se encuentra el objeto, pero también pueden ser otras como por ejemplo la velocidad del objeto en la escena.

Es interesante que una aplicación pueda disponer de distintos niveles de detalle del mismo modelo poligonal. De este modo, la aplicación muestra el nivel de detalle del objeto más adecuado a los requerimientos visuales. Éste servicio lo proporciona un modelo multiresolución. Este consiste en una estructura de datos que permite almacenar distintos niveles de detalle de un modelo poligonal y en un conjunto de métodos que facilitan el acceso a dichos niveles de detalle.

El programa *Qslim* desarrollado por M. Garland [4, 5, 6], implementa un método de simplificación que se basa en la contracción iterativa de aristas. El *Qslim* crea una lista con las aristas del modelo poligonal a simplificar y, a continuación, contrae dichas aristas (uniendo los dos vértices extremos en uno), según el orden de la lista, hasta llegar al número de polígonos deseado. La figura 1 muestra dos aproximaciones obtenidas con el *Qslim*.

El objetivo de este trabajo consiste en la implementación de una ampliación para el *Qslim* de manera que éste proporcione al usuario la posibilidad de que en cada iteración del proceso de simplificación, el nivel de detalle obtenido se



(a) 25 % (b) 5 %

Figura 1: *Dos aproximaciones distintas del modelo Ludwig. El porcentaje representa el número de caras utilizadas en comparación con el original.*

diferencie del anterior únicamente en una zona de la superficie poligonal a la que denominamos *región*. Para ello, es necesario agrupar varias simplificaciones en una sola. Esta agrupación se realiza si se cumple la condición de que exista coincidencia entre los triángulos afectados por cada una de las simplificaciones (ver figura 2). Los datos generados por este proceso se deberán almacenar con un nuevo formato de fichero de salida para que sirva como entrada de otras posibles aplicaciones, como por ejemplo un modelo multiresolución.

El *Qslim* acepta como entrada superficies *manifold* y *non-manifold* lo que significa que es capaz de tratar un amplio rango de superficies poligonales. Otro de los motivos de su eficiencia es que la métrica usada no requiere de un gran coste computacional y produce aproximaciones muy buenas de la superficie.

La construcción de modelos multirresolución a partir de los resultados producidos por la aplicación de un algoritmo

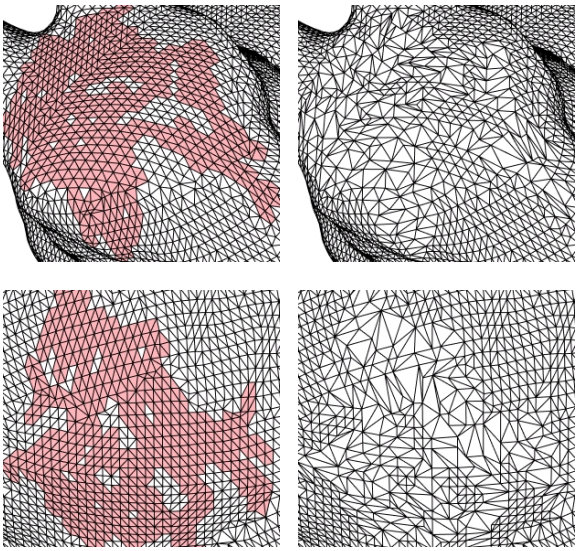


Figura 2: *Dos ejemplos de regiones simplificadas del modelo Conejo. A la izquierda aparecen los polígonos coloreados que forman una región. A la derecha, el resultado de su simplificación (sustitución de la región por otra formada por un número menor de triángulos).*

de simplificación es una de las principales aplicaciones de estos últimos. Sin embargo, las prestaciones que los modelos multirresolución alcanzan están condicionadas en parte por cómo se ha realizado dicha simplificación. Dicho de otra forma, para obtener modelos multirresolución que aprovechen al máximo las prestaciones de los procesadores gráficos más modernos será necesario estudiar cómo el proceso de simplificación influye en dichas prestaciones.

Este trabajo trata de ampliar el *Qslim* para poder generar simplificaciones de trozos de superficies que llamamos *regiones* y así poder estudiar la influencia en la variación del tamaño y localización de la región en las prestaciones finales de los modelos multirresolución.

Este informe está organizado de la siguiente forma. La sección 2 comenta la principal bibliografía y se explica el método de simplificación basado en contracción de aristas implementado en el *Qslim*. En la seción 3, se describe el *Qslim*, sus ficheros, la jerarquía de clases y su funcionamiento. En la sección 4, se presenta la principal aportación de este informe, es decir, la simplificación por regiones. Se describe el algoritmo y cómo se ha realizado la ampliación, explicando como queda dentro del organigrama del propio *Qslim*. El capítulo 5 muestra algunos resultados obtenidos utilizando para ello diversos modelos poligonales. Finalmente, en el capítulo 6 se presentan las conclusiones finales y las líneas de trabajo futuro.

2. TRABAJO RELACIONADO

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

La necesidad de utilizar métodos de simplificación aparece en muchos campos de la informática gráfica y también en otras áreas como la visión por ordenador, cartografía, análisis de elementos finitos, etc. Esto ha producido el desarrollo de una gran diversidad de métodos. Cada uno de estos métodos ha sido motivado por unas razones diferentes y, sin embargo, en algunas ocasiones llegan a aparecer coincidencias en la forma de resolver el problema [9].

Una de las grandes familias de estos métodos son los basados en la contracción iterativa. La mayor parte de este tipo de algoritmos realizan contracción iterativa de pares de vértices, aunque se han presentado métodos basados en contracción de caras [7]. De entre los que realizan contracción de pares de vértices, algunos exigen que el par elegido sea una arista [1, 8, 11, 10, 14] mientras que otros no [4, 13]. Este tipo de métodos permiten la simplificación de la topología y su aplicación a superficies *non-manifold*. Tal vez es el tipo de simplificación que más interés ha suscitado y que más estrechamente se ha relacionado con el modelado multirresolución. Probablemente el motivo es que estos métodos de simplificación crean de forma natural una jerarquía de vértices fácilmente representable mediante una estructura de árbol binario.

De entre todos estos métodos, se ha seleccionado el propuesto por Garland and Heckbert [4] debido a su velocidad, a la calidad de las aproximaciones generadas y a la implementación de dominio público existente *Qslim* [3].

En *Qslim* cada vértice lleva asociada una matriz que representa la suma de las ecuaciones de las distancias desde un punto a los planos en los que se inscriben los triángulos que contienen dicho vértice. A esta matriz se le llama *quadric* y representa el error que genera un vértice al desplazarse. El valor numérico del *quadric* se obtiene al resolver la matriz con las coordenadas del vértice asociado. En la contracción de una arista son los dos vértices implicados los que se desplazan para unirse en un único vértice y, por tanto, el error que genera este desplazamiento, esta contracción, se calcula sumando los dos *quadrics* asociados a los dos vértices implicados y resolviendo el *quadric* resultante con las coordenadas del nuevo vértice.

Se podría decir que la suma de las distancias desde el vértice resultante de una contracción a los triángulos implicados en dicha contracción y la suma de los *quadrics* implicados

en la contracción es lo mismo, sin embargo, esto no es del todo cierto. Cuando se suman los *quadrics* implicados en una contracción el *quadric* resultante asociado al nuevo vértice además de las distancias desde un punto a los triángulos que contienen dicho vértice, llevará implícitamente la carga de las ecuaciones de la distancia desde un punto a los triángulos que se han eliminado, ya que estos formaban parte de los *quadrics* de los vértices de la arista.

Esto no supone problema alguno, es más, esta sobrecarga hará que las superficies interiores se conserven mejor que las exteriores, que posiblemente tengan menor detalle por estar en los bordes de la superficie.

Asimismo, de este error depende la posición que tome el vértice resultante de la contracción. Se escogerá, dentro de la política de emplazamiento elegida, la posición que genere el menor error posible.

3. EL QSLIM

El *Qslim* se compone de varias librerías y de unos pocos ficheros que gestionan su uso en pro del proceso de simplificación. Una de esas librerías es la *libgfx* que implementa funciones generales para el tratamiento de superficies poligonales. La documentación de esta librería se puede encontrar en: http://graphics.cs.uiuc.edu/~garland/software/libgfx.html.

La otra librería usada en *Qslim* es la librería *MixKit*. Las clases de esta librería constituyen el núcleo del programa *Qslim*, ya que implementan las funciones específicas para el proceso de simplificación por contracción de aristas. En la figura 3 se muestra la relación de uso entre los ficheros más relevantes. Esta figura ayuda a visualizar la organización del código del programa, gracias a ella, resultará más fácil comprender la relación entre las clases de la librería.

Por claridad se han omitido las flechas que parten de los ficheros *MxBlock* y *MxDynBlock*, ya que implementan vectores de elementos, y sirven de soporte para el resto de ficheros.

En la tabla 1 se muestran las clases que hay implementadas en cada fichero, esta tabla sirve de apoyo a la figura 3 para comprender la relación entre las clases de la librería *MixKit*.

La jerarquía de clases de la librería *MixKit* se puede encontrar en la figura 4. Además del significado de herencia de las flechas, se ha intentado mostrar en esta figura la relación entre las clases de la librería *MixKit*. En la parte de componentes de la figura se encuentran las clases que se utilizan para construir las clases de la parte de núcleo. Las clases en componentes en la misma línea, forman la clase de la misma línea en núcleo. Asimismo, cada clase de núcleo se utiliza para la creación de la clase de la línea inmediatamente inferior. Las distintas zonas de simplificación, lectura y escritura no siguen esta última regla. De este modo se puede relacionar con más facilidad la organización del código con la jerarquía de clases. Las clases *MxBlock* y *MxDynBlock* no se

Fichero	Clases
MxBlock	MxBlock
MxDynBlock	MxDynBlock MxSizedDynBlock
MxBlockModel	MxBlockModel
MxGeoPrims	MxColor MxTexCoord MxVertex MxNormal MxEdge MxFace
MxStdModel	MxPairContraction MxStdModel
MxHeap	MxHeapable MxHeap
MxStdSlim	MxStdSlim
MxQMetric3	MxQuadric3
MxQslim	MxQslim MxQslimEdge MxEdgeQslim
MxAsp	MxAspVar MxAspStore
MxCmdParser	MxCmd MxCmdParser
MxStack	MxStack
MxSMF	MxSMFReader MxSMFWriter

Tabla 1: *Relación de los ficheros más relevantes con las clases implementadas en cada uno de ellos.*

han mantenido a nivel, por claridad de la figura, sabiendo de antemano que participan en el resto de clases de la librería.

El *Qslim* es un programa ejecutable en línea de comandos sin ningún tipo de interfaz gráfica en el que se establecen ciertos parámetros que rigen su funcionamiento.

Para la ejecución del *Qslim* se necesita:

- el fichero con el modelo poligonal a simplificar,
- el número de triángulos que deben definir al modelo simplificado,
- el nombre del fichero destino,
- la elección del formato de salida, es decir, el modelo poligonal simplificado se puede escribir en el fichero destino en formato VRML, IV, MMF, LOG y SMF. En los formatos VRML y IV se escriben solamente los vértices y los triángulos del modelo resultante. El formato MMF escribe el modelo poligonal simplificado en formato SMF y a continuación, escribe en orden de ejecución los dos vértices de cada contracción realizada junto con la distancia que se ha movido cada vértice hasta el destino y los triángulos que han desaparecido en dicha contracción. En el formato LOG se escribe el modelo poligonal en SMF y seguidamente, se escribe en orden de ejecución toda la información contenida en cada contracción.

Otros parámetros de entrada en *Qslim* se refieren a la po-

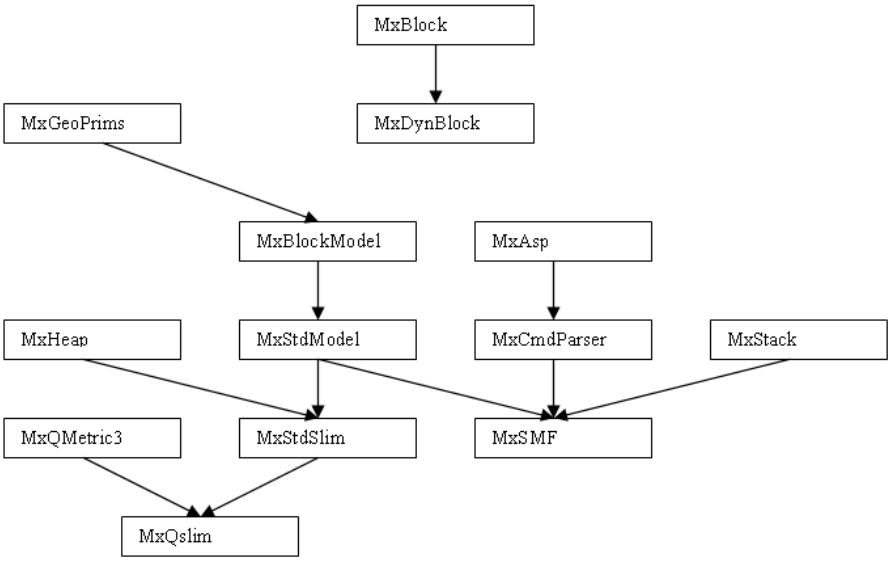


Figura 3: *Relación entre los ficheros más importantes de la librería MixKit.*

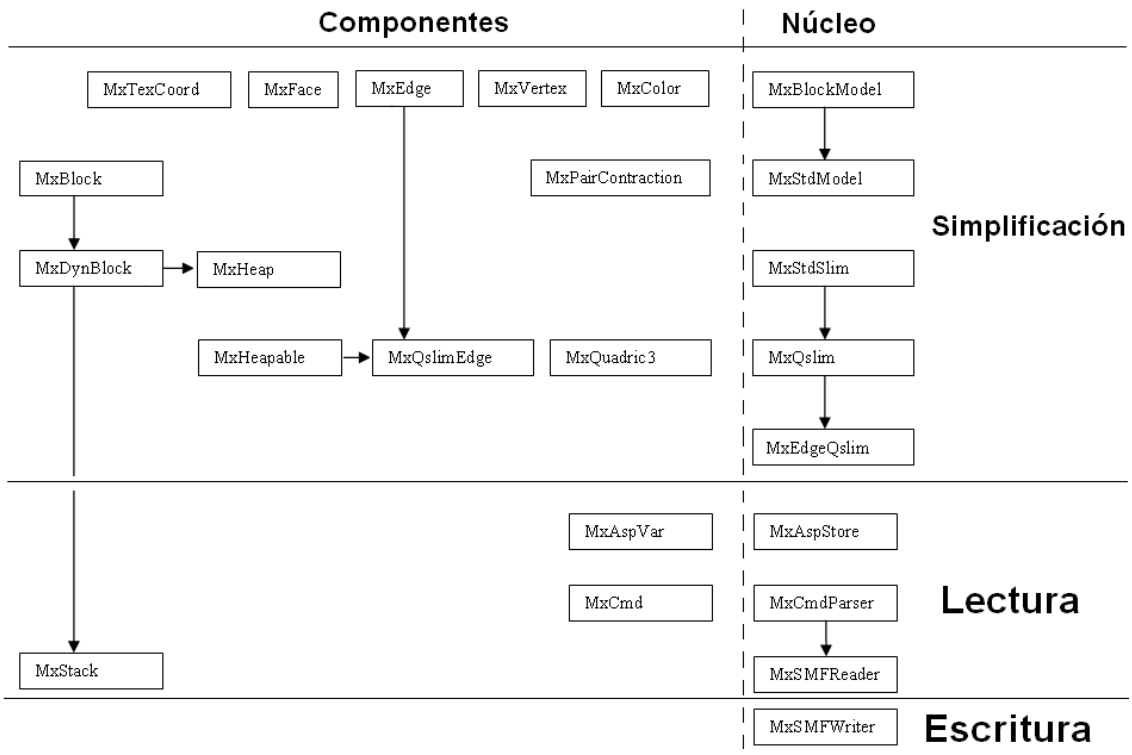


Figura 4: *Jerarquía de las clases del Qslim.*

lítica de emplazamiento, la política de ponderación y las restricciones de malla que se usan para el cálculo del error.

Cuando se ejecuta el *Qslim* con los parámetros correspondientes, dichos parámetros se guardan en variables globales del programa para facilitar su posterior acceso. A continuación, se crea el objeto *MxSMFReader* que leerá el fichero SMF con el modelo poligonal y llenará un objeto *MxStdModel* con la información de ese modelo.

Una vez se tiene el modelo poligonal cargado en un objeto *MxStdModel*, se crea un objeto de la clase *MxQslimEdge* y se le pasa la información almacenada en los parámetros globales mencionados anteriormente. Al inicializar este objeto se computan todos los objetos (MxEdgeQslim) correspondientes a las aristas del modelo poligonal junto con el error que generarían al contraerse y se insertan en el montículo en orden decreciente error. Seguidamente, si el formato de salida así lo requiere, se establece la función de registro. Esta función se ejecutará una vez por cada contracción almacenando en el vector de registro la información correspondiente a dicha contracción.

En este momento dará comienzo el proceso de simplificación de la malla. Este proceso comienza extrayendo un arista, un objeto *MxQslimEdge*, del montículo. Por definición, este objeto se extrae de la raíz del montículo, posición donde siempre estará la arista que genere menos error al contraerse. A partir este objeto, se calcula la contracción correspondiente, un objeto *MxPairContraction*. Esta contracción se almacena en un vector que se encarga de llevar a cabo un registro de las contracciones realizadas para su posterior escritura el fichero de salida, si así se ha dispuesto. A continuación, se aplica la contracción calculada sobre el modelo poligonal. Se recalculan los errores de las aristas de los triángulos afectados y se actualizan los correspondientes nodos del montículo con esos valores. De esta manera se consiguen mantener los nodos del montículo en orden decreciente de error durante todo el proceso de simplificación.

Este proceso se repite hasta que el modelo poligonal alcanza un número de triángulos menor o igual que el número deseado, introducido por el usuario como parámetro en la línea de órdenes.

Finalmente se escribe el modelo simplificado en un fichero con el formato elegido. También se escribe por pantalla, si así se ha establecido, el tiempo que ha tardado cada fase del programa: lectura y carga, simplificación y escritura del modelo poligonal.

4. SIMPLIFICACIÓN POR REGIONES

Se ha modificado el comportamiento del *Qslim* para que también pueda simplificar la superficie de un modelo poligonal mediante la simplificación por regiones. En cada iteración de este nuevo proceso solo se simplifica una región de la superficie del modelo poligonal. Una región creada de esta manera es una zona de la superficie formada por el conjunto de triángulos conexos implicados en una serie de contracciones de aristas. La región más simple, por tanto, es aquella formada por los triángulos que se verán afectados por la contracción de una única arista. Los triángulos implicados en una nueva contracción pasan a formar parte de una región ya existente si alguno de ellos ya pertenecía a esa región. Las regiones se van construyendo a medida que las contracciones van sucediendo.

En *Qslim*, el orden en que las aristas se contraen viene dado por el error que cada una de estas genera al contraerse. Con este razonamiento, el orden de simplificación de las regiones vendrá dado por el error que cada una de estas genere al simplificarse. Esto es así debido a que las regiones acumulan implícitamente el error de las contracciones realizadas en la misma y, por tanto, también siguen un orden de simplificación.

Dependiendo del formato de fichero de salida elegido por el usuario en la ejecución del *Qslim*, éste lleva un registro de las contracciones realizadas durante el proceso de simplificación para posteriormente escribirlas en el fichero de salida correspondiente. Este comportamiento es el que se desea del nuevo proceso de simplificación por regiones pero con la salvedad de que en lugar de contracciones se guarden regiones de simplificación.

Existe una función en *Qslim* que dada una contracción la almacena en un vector que actúa como registro. En la clase *MxEdgeQslim* existe un puntero a funciones que reciben como parámetro un objeto de la clase *MxPairContraction* y que no devuelven nada. Este puntero se establece que apunte a dicha función durante la inicialización de los parámetros globales del programa dependiendo del formato de fichero de salida elegido. Durante el proceso de simplificación se llama a esta función, por medio del puntero, antes de la ejecución de cada contracción. Análogamente se creará una función que dada una contracción averiguará si los triángulos implicados en dicha contracción deben formar parte de alguna de las regiones existentes (ver figura 5). En caso afirmativo estos triángulos se adherirán a la región correspondiente (ver figura 6). Puede suceder que estos triángulos deban formar parte de más de una región simultáneamente, en cuyo caso, éstos y los triángulos de las regiones implicadas se unirían para formar una única región. En caso negativo los triángulos implicados en dicha contracción formarán una nueva región de simplificación. En la figura 7 se muestra el pseudocódigo correspondiente a esta función. También se creará un vector que almacene regiones de simplificación de manera similar al vector que almacena contracciones en *Qslim* y, un puntero análogo al anterior pero que apunte a la función recientemente comentada. Al igual que en el proceso original, se llamará a dicha función antes de la ejecución de cada contracción. De esta forma se consigue una gestión de regiones de manera similar a como lo hace el *Qslim* original con las contracciones de aristas.

```
función Región::PertenceARegión (Contracción) {

  para (Triángulo ∈ (Contracción.TriángulosEliminados ∪ Contracción.TriángulosModificados))
    si (Triángulo ∈ Región.TriángulosNuevos)
      devuelve cierto
    sino
      devuelve falso
}
```

Figura 5: *Pseudocódigo del método de la clase MxRegion que indicará si los triángulos implicados en una contracción pertenecen a la región que un objeto de esta clase representa.*

```
función Región::AñadeARegión (Contracción) {

  para (Triángulo ∈ Contracción.TriángulosEliminados)
    si (Triángulo ∈ Región.TriángulosNuevos)
      Región.TriángulosNuevos = Región.TriángulosNuevos - Triángulo
    sino
      Región.TriángulosEliminados = Región.TriángulosEliminados + Triángulo

  /* Nótese que aunque al eliminar un triángulo de la lista de triángulos nuevos de la región
  debería añadirse éste a la lista de triángulos eliminados de dicha región, no se añade porque
  no se desean almacenar los triángulos intermedios eliminados. Eliminar un triángulo de la
  lista de triángulos nuevos de la región implica que ese triángulo ya ha sido eliminado
  anteriormente. Un triángulo modificado por una contracción entra a formar parte de una región
  como dos triángulos, uno que se elimina, y pasa a formar parte de la lista de triángulos
  eliminados de la región, y, otro que se crea, y pasa a formar parte de la lista de triángulos
  nuevos de la región. */

  para (Triángulo ∈ Contracción.TriángulosModificados) {
    si (Triángulo ∈ Región.TriángulosNuevos)
      Región.TriángulosNuevos = Región.TriángulosNuevos - Triángulo
    sino
      Región.TriángulosEliminados = Región.TriángulosEliminados + Triángulo

  /* Nótese que hasta aquí sucede exactamente lo mismo en este bucle que en el anterior */
    Triángulo2 = NuevoTriángulo(Triángulo.v1,Triángulo.v2,Triángulo.v3)
    Región.TriángulosNuevos = Región.TriángulosNuevos + Triángulo2
  }

  /* Al ser un triángulo modificado lo que se está añadiendo a la región, además de suceder
  exactamente lo mismo que en el bucle anterior, se debe añadir ese triángulo a la lista de
  triángulos nuevos de la región como un nuevo triángulo. Esta es precisamente la diferencia
  entre ambos bucles. */

}
```

Figura 6: *Pseudocódigo del método de la clase MxRegion que añadirá los triángulos implicados en una contracción a la región que un objeto de esta clase representa.*

```
función GestionaRegiones (Contracción) {
  si (VectorRegiones == 0){
    Región = NuevaRegión
    Región.AñadeARegion(Contracción)
    si (contar(Región.TriángulosEliminados) >TriángulosDeseados){
      VectorRegionesSolución = VectorRegionesSolución + Región
      VectorRegiones = VectorRegiones - Región
    }
  } sino {
    contracción_añadida=falso
    para (Región ∈ VectorRegiones) {
      si (Contracción ∈ Región) {
        para ((Region2!=Región) ∈ VectorRegiones) {
          si (Contracción ∈ Región2){
                        Región= Región ∪ Región2
                        VectorRegiones = VectorRegiones - Región2
        } }
        Región.AñadeARegión(Contracción)
        contracción_añadida=cierto
        si (contar(Región.TriángulosEliminados) >TriángulosDeseados){
          VectorRegionesSolución = VectorRegionesSolución + Región
          VectorRegiones = VectorRegiones - Región
    } } }
    si (contraccion_añadida==falso) {
      Región = NuevaRegión
      Región.AñadeARegion(Contracción)
      si (contar(Región.TriángulosEliminados) >TriángulosDeseados){
        VectorRegionesSolución = VectorRegionesSolución + Región
        VectorRegiones = VectorRegiones - Región
  } } }
  si (TriángulosAproximación <= TriángulosDeseados) {
    Región = NuevaRegión
    para (Región2 ∈ VectorRegiones) }
      Región = Región ∪ Región2
      VectorRegiones = VectorRegiones - Región2
    }
    VectorRegionesSolución = VectorRegionesSolución + Región
  }
}
```

Figura 7: *Pseudocódigo de la función principal de gestión de regiones.*

En *Qslim* la información almacenada por un objeto de la clase *MxPairContraction* es muy similar a la que deberá contener una región de simplificación, la nueva clase *MxRegion* será la encargada de esto. Un objeto *MxPairContraction* guarda una lista con los índices de los triángulos del modelo poligonal eliminados por una contracción y un objeto de la clase *MxRegion* guardará los índices de los triángulos del modelo poligonal eliminados por una serie de contracciones. Pero, mientras que un objeto *MxPairContraction* guarda una lista con los índices de los triángulos del modelo poligonal modificados por una contracción, un objeto de la clase *MxRegion* guardará los nuevos índices de los nuevos triángulos creados en modelo poligonal para llenar el hueco dejado por los triángulos eliminados por una serie de contracciones.

Esto último no quiere decir otra cosa que, en la ampliación del *Qslim*, cada triángulo que se modifica por una contracción, se considerará que son dos triángulos, uno que se elimina, que tiene un índice existente en el modelo, y, otro que se crea, que tiene un índice nuevo. De esta manera se consigue que una región almacene en su lista de triángulos eliminados, los triángulos originarios de la región, iniciales, y, en su lista de triángulos nuevos, los triángulos que han sustituido a los triángulos eliminados, finales. Esto es así porque se desea que las aproximaciones se almacenen de forma explícita. En *Qslim* una aproximación se almacena de forma implícita, esto es, el fichero de salida guarda el modelo original junto con la serie de contracciones realizadas. Con un almacenamiento implícito, para obtener la aproximación correspondiente

```
función EscribeRegiones (VectorRegiones) {
  para (Región ∈ VectorRegiones) {
    escribir(contar(Región.TriángulosEliminados), contar(Región.TriángulosNuevos))
    ordenar(Región.TriángulosEliminados)
    ordenar(Región.TriángulosNuevos)
    para  (Triángulo ∈ Región.TriángulosEliminados) escribir(Triángulo)
    para  (Triángulo ∈ Región.TriángulosNuevos) {
      escribir(Triángulo)
      para  (Vértice ∈ Triángulo) escribir(Vértice)
  } }
  ordenar(Triángulos restantes del modelo)
  para  (Triángulo ∈ Triángulos restantes del modelo) escribir(Triángulo)
}
```

Figura 8: *Pseudocódigo de la función que escribe el resultado del proceso de simplificación en formato REG.*

a una iteración cualquiera, es necesario calcular todas las contracciones realizadas hasta esa iteración obteniendo, de esta manera, la malla correspondiente a dicha aproximación. Con el almacenamiento explícito, para obtener la aproximación correspondiente a una iteración cualquiera, basta con sustituir en la superficie del modelo los triángulos correspondientes a los triángulos eliminados de la región obtenida en dicha iteración por los triángulos nuevos de esa misma región. Un almacenamiento explícito es más rápido cargando las aproximaciones que uno implícito pero necesita más memoria para almacenarlas que éste. Asimismo proporciona cierta independencia del proceso de simplificación ya que no necesita conocer cómo se han calculado las contracciones para poder obtener una aproximación.

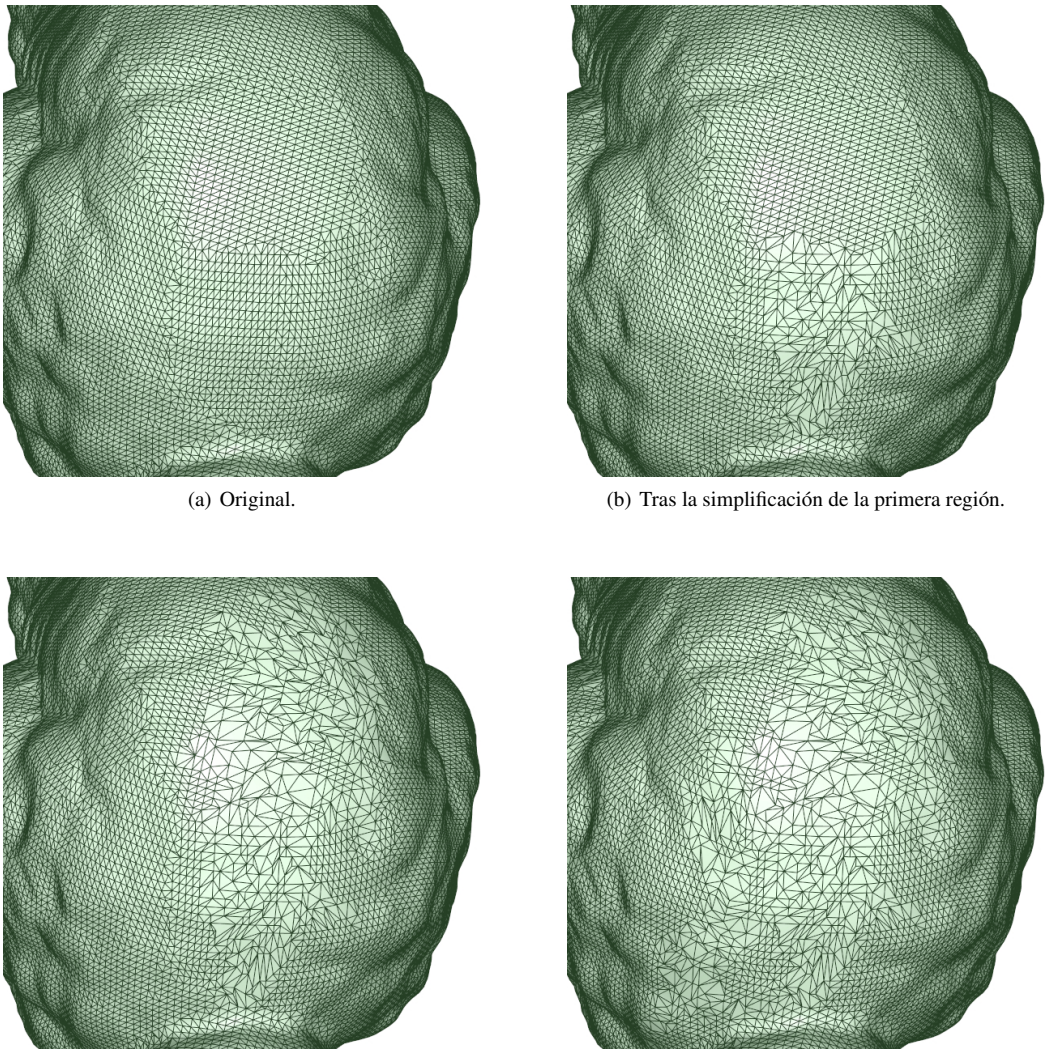
El interpretar que un triángulo modificado por una contracción sean dos, uno que se elimina y otro que se crea, hace que durante la formación de una región la lista de triángulos eliminados contenga más triángulos de los necesarios. Por ejemplo supongamos que un triángulo ha sido modificado por tres contracciones distintas. Únicamente se requeriría un triángulo en la lista de triángulos eliminados para representar el triángulo original de la superficie, el inicial, y, un triángulo en la lista de triángulos nuevos para representar el triángulo que ha sustituido al original, el final. Sin embargo, en la primera contracción la ampliación genera un triángulo eliminado y un triángulo nuevo. En la segunda, el triángulo creado en por la contracción anterior se elimina y se crea otro nuevo, ya hay dos triángulos eliminados correspondientes al mismo triángulo. En la tercera contracción serán tres los triángulos eliminados cuando únicamente se requiere uno. Por este motivo, los triángulos intermedios que se eliminan durante la formación de una región no se almacenan en la lista de triángulos eliminados.

La clase *MxRenamingAid*, por un lado, ayudará a simular el nuevo comportamiento explícito del *Qslim*, es decir, ayudará a interpretar un triángulo modificado como dos, uno que se elimina y otro que se crea. Para ello, dispondrá de un vector de longitud igual al número de triángulos del modelo original de manera que, cada posición del vector representará al triángulo correspondiente en el modelo poligonal y contendrá, antes de una contracción, el índice del triángulo que se eliminará y, después de dicha contracción, el índice del triángulo nuevo que se creará. Por otro lado, esta clase dispondrá de un vector de longitud igual al número de vértices del modelo original de manera que cada posición del vector representará al vértice correspondiente en el modelo poligonal y contendrá el índice del vértice que ha prevalecido en la contracción, ya que en *Qslim* es siempre el segundo vértice de la arista a contraer el que desaparece y, el primero, que guardará la posición resultante de la contracción de dicha arista, el que prevalece.

El formato de fichero que se introducirá para la salida de la ampliación del *Qslim* será el formato REG. En éste se escribirán las regiones en el orden de simplificación de las mismas. Además, para cada región se escribirá, por una parte, el número de triángulos eliminados y sus correspondientes índices y, por otra, el número de triángulos nuevos y sus correspondientes índices junto con los índices de los vértices que los forman. En la figura 8 se muestra el pseudocódigo de la función que se encargará de escribir en este formato.

5. RESULTADOS

En este sección se muestran algunas aproximaciones resultantes de aplicar la ampliación desarrollada en este proyecto sobre tres modelos poligonales. Estos modelos pertenecen al banco de pruebas de los programadores de algoritmos para el tratamiento de gráficos, es decir, son ampliamente usados por otras personas para realizar baterías de tests con sus aplicaciones. El Conejo esta compuesto de 69.451 triángulos y 35.947 vértices, el Armadillo contiene 345.944 triángulos y 172.974 vértices y el Dragón esta formado por 715.933 triángulos y 359.173 vértices. Se han escogido estos modelos por su diferencia en número de triángulos y de vértices para demostrar que la ampliación implementada es capaz de manejar cualquier superficie.



(a) Original.

(b) Tras la simplificación de la primera región.

(c) Tras la simplificación de la primera y la segunda región.

(d) Tras la simplificación de las tres primeras regiones.

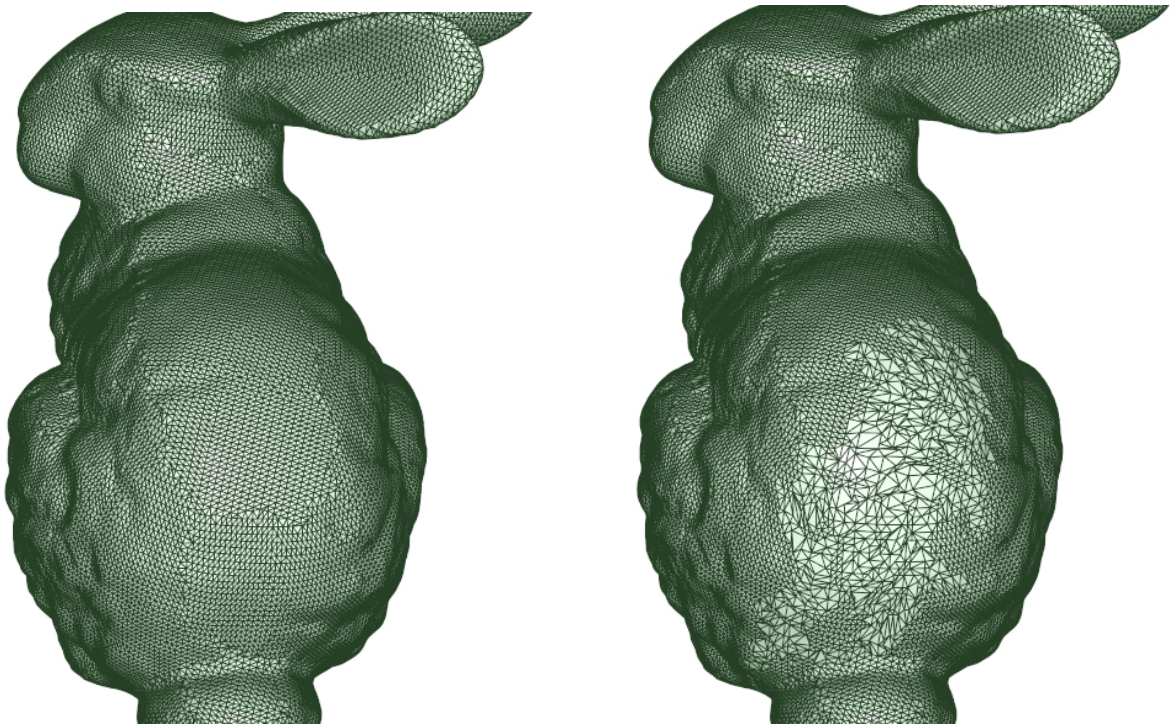
Figura 9: *Aproximaciones resultantes de la simplificación por regiones con tamaño 700 sobre el modelo poligonal Conejo*

Las aproximaciones resultantes de aplicar la simplificación por regiones al modelo Conejo que se muestrán en la figura 9 son ejemplos de como se conforman las regiones de simplificación. Estas aproximaciones corresponden a los tres primeros niveles de detalle obtenidos con un tamaño de región de 700 triángulos. Se puede observar como un nivel de detalle se diferencia del anterior en una única zona de la superficie, la región de simplificación.

En la figura 10 se compara la malla del modelo Conejo original con el tercer nivel de detalle resultante de la simplificación por regiones de 700 triángulos, es decir, con las tres primeras regiones obtenidas simplificadas. En la figura 11 se muestran varios ejemplos más de niveles de detalle obtenidos mediante el proceso de simplificación por regiones aplicado al modelo Conejo.

A continuación se muestran los niveles de detalle correspondientes al modelo poligonal Armadillo (ver figura 12) y Dragón (ver figura 13). La elevada densidad de polígonos de estos dos modelos impide mostrar toda su superficie en estas figuras, por ello, se ha escogido la zona de la cara de cada uno de ellos para representar los distintos niveles de detalle.

La tabla 2 muestra los tiempos (y entre paréntesis el tamaño de la región) empleados en simplificar los modelos Conejo, Armadillo y Dragón con el *Qslim* y con el nuevo método basado en regiones para diferentes tamaños de región obte-



(a) Antes. (b) Después.

Figura 10: *Vista del modelo Conejo antes y después de la simplificación de las tres primeras regiones.*

Modelo	Qslim (seg)	80 LoDs	140 LoDs	300 LoDs	1000 LoDs
Conejo	0.8	4 (1260)	2 (700)	1 (335)	1 (102)
Armadillo	4.8	195 (5900)	91 (3350)	32 (1595)	16 (490)
Dragón	9.6	1622 (12328)	648 (7000)	193 (3330)	67 (1015)

Tabla 2: *Tiempos de simplificación en segundos del Qslim y del método basado en regiones para los modelos Conejo, Armadillo y Dragón. Entre paréntesis figura el tamaño de la región.*

niendo distintos números de niveles de detalles (LoDs en la tabla). Como es lógico, el tiempo de la ampliación es claramente más lento sobre todo al aumentar el tamaño de la región. Sin embargo, ya que el proceso de simplificación es un proceso realizado fuera de línea, la importancia de este hecho es relativamente pequeña.

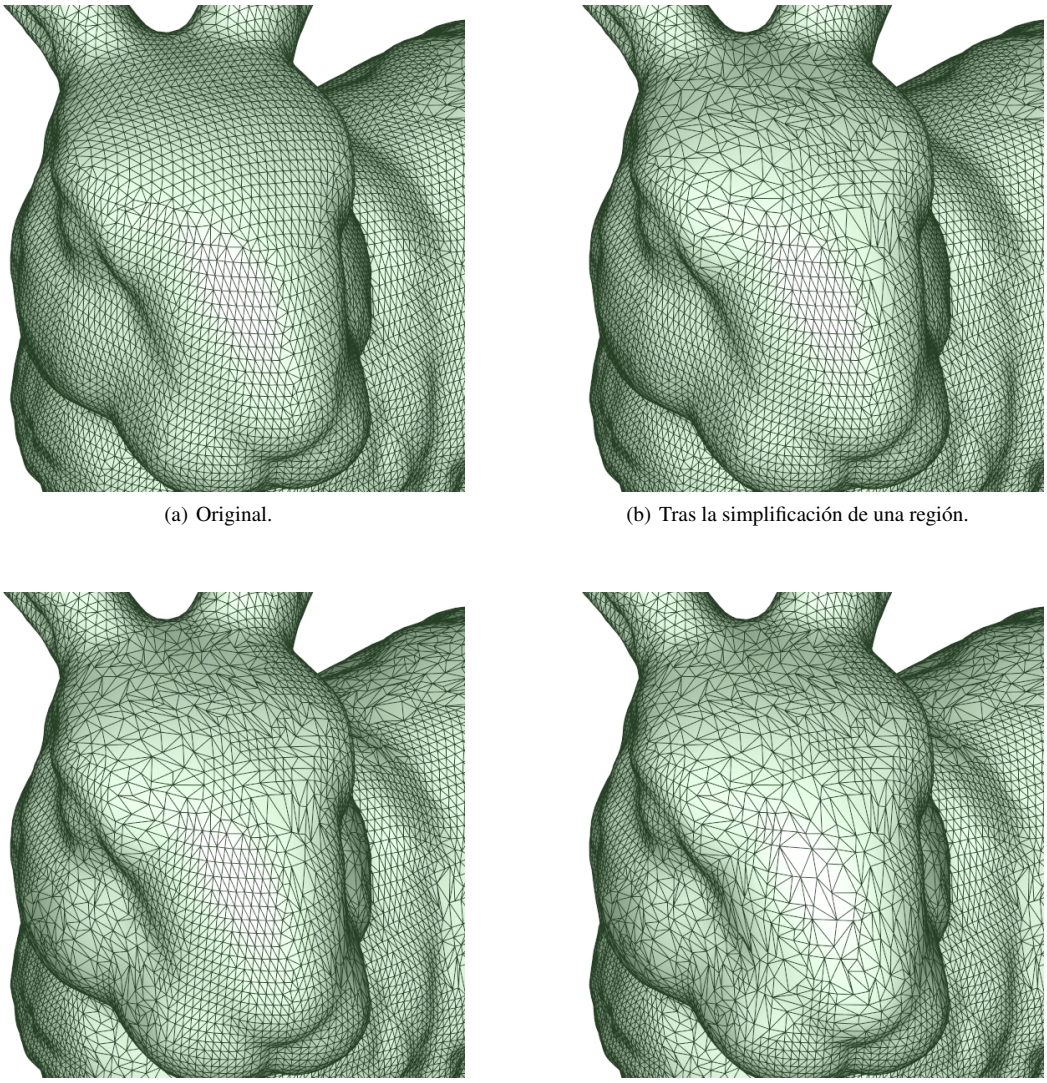
6. CONCLUSIONES

La modificación que se ha aplicado al programa de simplificación *Qslim* permite obtener distintos niveles de detalle del modelo que se esta simplificando de manera que cada nivel de detalle se diferencia del anterior únicamente en una zona de la superficie de dicho modelo (ver figura 10). Los datos obtenidos de este modo son idóneos para ser almacenados en un modelo multiresolución.

Los resultados que se obtienen respecto a la calidad de la simplificación son comparables a los obtenidos con el *Qslim*, ya que la métrica utilizada para realizar la simplificación es la misma. Asimismo se obtienen muy buenos tiempos de simplificación a pesar de las búsquedas que se han de realizar en las largas listas de triángulos durante el crecimiento de las regiones.

Referencias

1. Maria-Elena Algorri and Francis Schmitt. Mesh simplification. *Computer Graphics Forum*, 15(3):C77–C86, September 1996.
2. Foley D. *Computer graphics principles and practice*. Addison-Wesley, second edition, 1990.
3. Garland. Qslim 2.1. http://graphics.cs.uiuc.edu/~garland/software/qslim.html.
4. Michael Garland and Paul Heckbert. Surface simplification using quadric error metrics. *SIGGRAPH 97*, 1997.
5. Michael Garland and Paul Heckbert. Simplifying surfaces with color and texture using quadric error metrics. *IEEE Visualization 98*, 1998.



(a) Original.

(b) Tras la simplificación de una región.

(c) Tras la simplificación de dos regiones.

(d) Tras la simplificación de tres regiones.

Figura 11: *Regiones de simplificación de tamaño 700 sobre la cabeza del modelo poligonal Conejo.*

6. Michael Garland and Paul Heckbert. Optimal triangulation and quadric-based surface simplification. *Journal of Computational Geometry: Theory and Applications*, 14(1-3):49–65, 1999.

7. Tran S. Gieng, Bernd Hamann, Kenneth I. Joy, Greg Schussman, and Isaac J. Trotts. Constructing hierarchies for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):145–161, April 1998.

8. André Guéziec. Surface simplification inside a tolerance volume. Technical report. IBM Research Report RC 20440, MONTH=Mar., YEAR=1996, INSTITUTION=Yorktown Heights, NY 10598,.

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

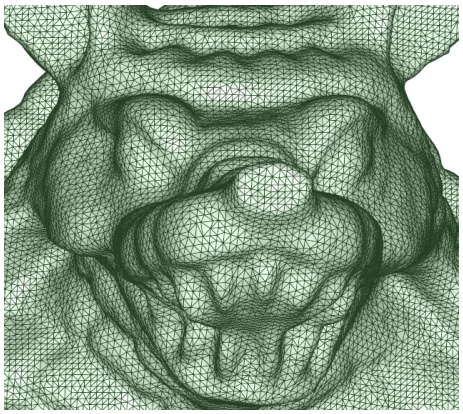
10. Hugues Hoppe. Progressive meshes. *Computer Graphics*, 30(Annual Conference Series):99–108, 1996.

11. Peter Lindstrom and Greg Turk. Fast and memory efficient polygonal simplification. In *IEEE Visualization*, pages 279–286, 1998.

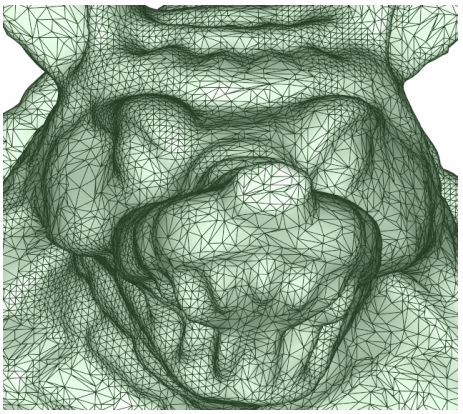
12. David Luebke. *Level of detail for 3D graphics*. Morgan Kaufmann, 2003.

13. Jovan Popovic and Hugues Hoppe. Progressive simplicial complexes. In *SIGGRAPH*, pages 217–224, 1997.

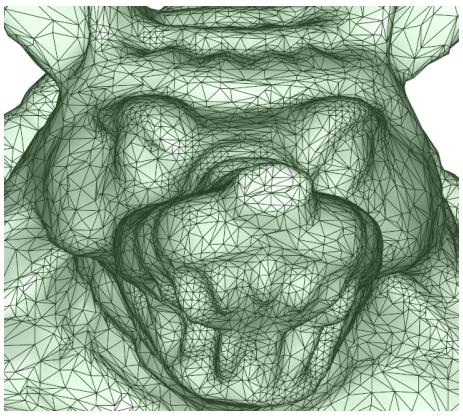
14. Remi Ronfard, Jarek Rossignac, and Jarek Rossignac. Full-range approximation of triangulated polyhedra. In Jarek Rossignac and Francois Sillon, editors, *Proceeding of Eurographics, Computer Graphics Forum*, volume 15(3), pages C67–C76. Eurographics, Blackwell, August 1996.



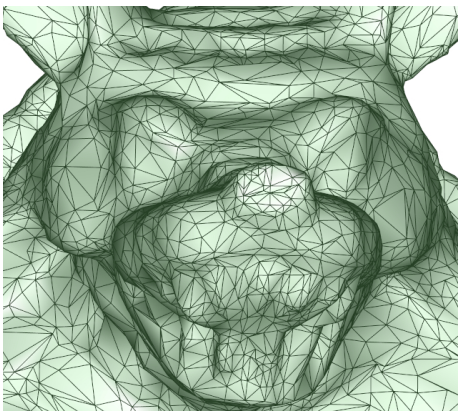
(a) Modelo original, 100 % de los triángulos iniciales.



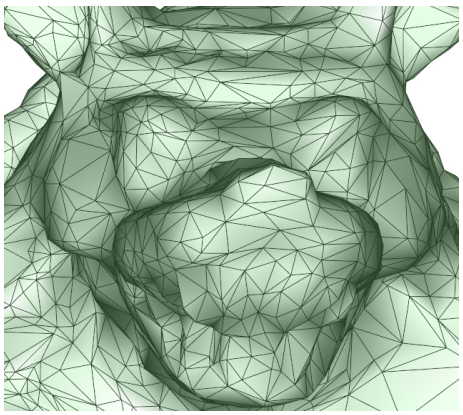
(b) Primera aproximación, 50 % de los triángulos iniciales, tras simplificar 70 regiones.



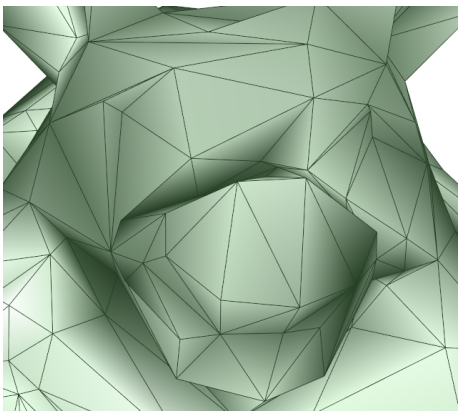
(c) Segunda aproximación, 25 % de los triángulos iniciales, tras simplificar 105 regiones.



(d) Tercera aproximación, 10 % de los triángulos iniciales, tras simplificar 126 regiones.

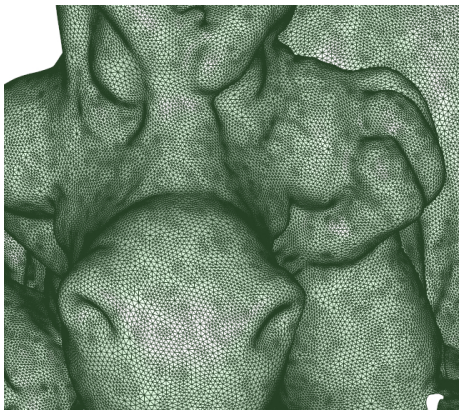


(e) Cuarta aproximación, 5 % de los triángulos iniciales, tras simplificar 133 regiones.

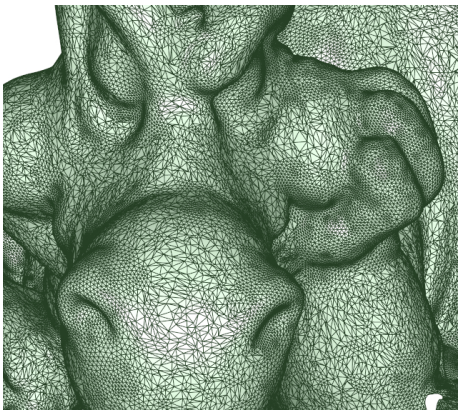


(f) Quinta aproximación, 0.5 % de los triángulos iniciales, tras simplificar 140 regiones.

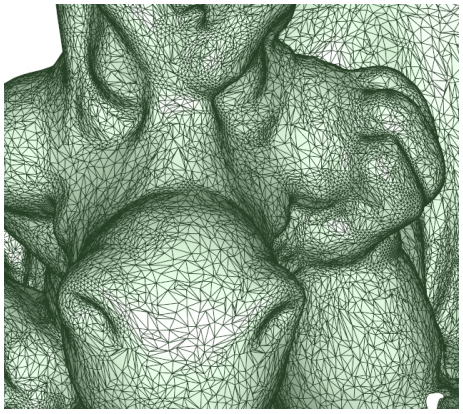
Figura 12: *Aproximaciones obtenidas tras la simplificación por regiones de tamaño 3.350 del modelo poligonal Armadillo.*



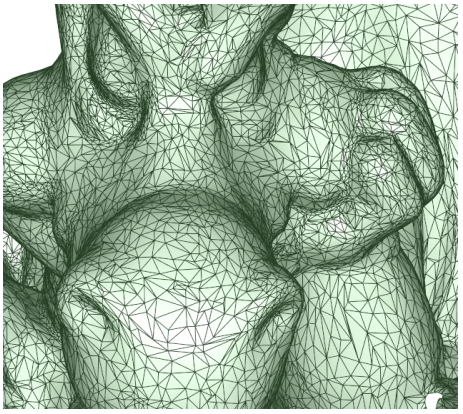
(a) Modelo original, 100 % de los triángulos iniciales.



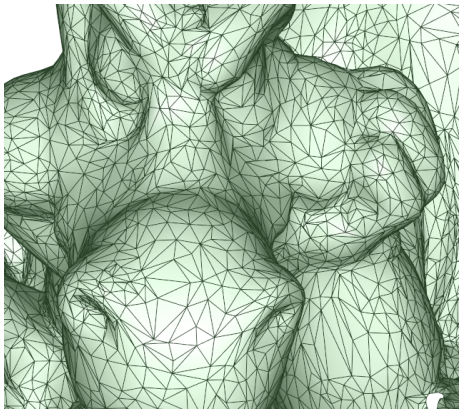
(b) Primera aproximación, 50 % de los triángulos iniciales, tras simplificar 50 regiones.



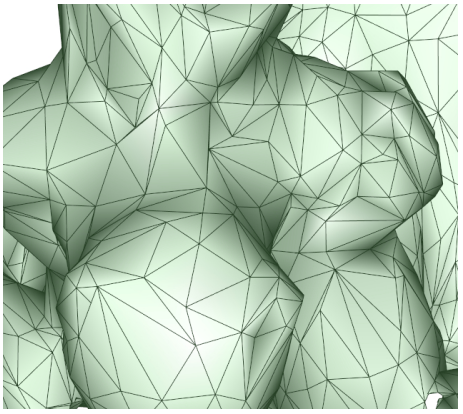
(c) Segunda aproximación, 25 % de los triángulos iniciales, tras simplificar 75 regiones.



(d) Tercera aproximación, 10 % de los triángulos iniciales, tras simplificar 90 regiones.



(e) Cuarta aproximación, 5 % de los triángulos iniciales, tras simplificar 95 regiones.



(f) Quinta aproximación, 1 % de los triángulos iniciales, tras simplificar 100 regiones.

Figura 13: *Aproximaciones obtenidas tras la simplificación por regiones de tamaño 7000 del modelo poligonal Dragón.*