Búsqueda de tiras para modelos multirresolución estáticos

Oscar E. Ripollés, Miguel Chover
Dept. Lenguajes y Sistemas Informáticos
Universidad Jaume I
12071 Castellón de la Plana
oripolle@sg.uji.es, chover@sg.uji.es

Resumen

Los modelos multirresolución se emplean habitualmente en aplicaciones gráficas para aliviar el tráfico de información entre la CPU y la GPU. Del mismo modo, la utilización de las tiras de triángulos aumenta la eficiencia de estos modelos pero presenta ciertas limitaciones. La creación dinámica de las tiras supone un coste computacional elevado, y la utilización de un único conjunto de tiras muestra un alto grado de degeneración en los niveles de bajo detalle. Presentamos un algoritmo de búsqueda de tiras para solucionar este problema, en el que éstas se construyen considerando que serán simplificadas dentro de un modelo multirresolución. Para ello, partimos de un conjunto de tiras de triángulos a mínimo nivel de detalle y obtenemos las correspondientes al máximo detalle respetando las contracciones de aristas del método de simplificación.

1. Motivación

Una forma habitual de enfrentarse al problema que representa el trabajar con los modelos poligonales actuales, con una alta y creciente complejidad, es el uso de técnicas de modelado multirresolución. Según Garland [9], un modelo multirresolución representa a un objeto mediante un conjunto de aproximaciones de diferente nivel de detalle y permite recuperar cualquiera de ellas bajo demanda. Los primeros modelos multirresolución que se desarrollaron estaban basados en un número de aproximaciones relativamente pequeño (normalmente entre 5 y 10) [8] y se les denominó modelos multirresolución discretos. Más tarde, con el fin de solucionar los problemas que presentaba este tipo de representación, aparecieron los modelos multirresolución continuos que disponen de un amplio rango de aproximaciones distintas para representar el objeto original. Los modelos multirresolución continuos se usan ampliamente porque son capaces de resolver los problemas de la visualización interactiva, la transmisión progresiva, la compresión geométrica y la resolución variable. En [14] podemos encontrar una caracterización completa de los modelos multirresolución.

El uso de estos modelos multirresolución permite reducir la cantidad de geometría que se envía al sistema gráfico, lo que se traduce en un aumento de las prestaciones. Los triángulos son la primitiva gráfica más habitual, ya que la sencillez de sus elementos básicos y su conectividad permiten que sean rápidamente manipulados por todas las librerías y sistemas gráficos [12]. Pero, por su parte, las tiras de triángulos permiten aprovechar estas características a la vez que ofrecen una representación compacta de la conectividad existente en una malla de triángulos, y una mayor rapidez en su representación ya que el número de vértices a recortar, transformar e iluminar es menor. En ocasiones, las tiras de triángulos no son suficientes por sí mismas para representar una superficie poligonal formada por triángulos, por lo que es necesario recurrir a la operación de *swap* o a la repetición de vértices, obteniéndose tiras que reciben el nombre de generalizadas. Obviamente, estas tiras ofrecen un menor ahorro respecto a las mallas de triángulos que las tiras no generalizadas o secuenciales.

El principal problema de los modelos multirresolución basados en tiras aparece cuando, al partir de un conjunto de tiras que representan la malla inicial a máximo detalle e ir aplicando las sucesivas simplificaciones, las tiras van incluyendo una gran cantidad de vértices repetidos y aristas innecesarias. Un ejemplo de este tipo de tiras se puede observar en la Figura 1, donde una de las tres tiras ha sido colapsada tras aplicar dos

pasos de la simplificación en los que las aristas *0, 3* y *1, 2* también han sido colapsadas.

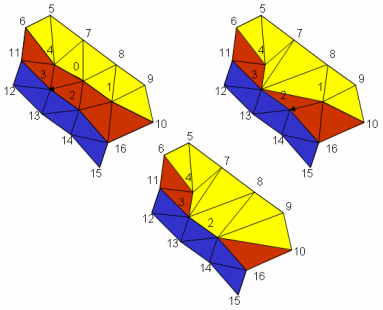

Figura 1. Colapso de una tira.

Una posible forma de aliviar este problema es la utilización de tiras que se generen dinámicamente para cada nivel de detalle. El sobrecoste de su generación es elevado, por lo que puede llegar a resultar más interesante utilizar tiras estáticas a pesar de sus limitaciones.

El trabajo que se presenta en este artículo propone un algoritmo de búsqueda de tiras generalizadas para modelos estáticos, que evite la aparición de tiras degeneradas de mala calidad. Se propone construir las tiras desde el mínimo al máximo nivel de detalle siguiendo la secuencia de simplificaciones, evitando la aparición de cualquier tipo de repetición o arista innecesaria pero siempre manteniendo el aspecto original del modelo.

El artículo presenta la siguiente estructura. En el apartado 2 se incluye un estudio del trabajo realizado previamente sobre búsqueda de tiras y modelos multirresolución basados en esta primitiva. En el apartado 3 se detalla el método propuesto. El apartado 4 ofrece una comparativa de nuestro algoritmo frente a otras posibles soluciones, así como imágenes mostrando las tiras obtenidas como resultado de la aplicación de este algoritmo. Por último, en la sección 5 se comentan los resultados obtenidos y se esbozan las líneas de trabajo futuro.

2. Trabajo previo

2.1. Búsqueda de tiras

El uso de la primitiva tira de triángulos permite acelerar en gran medida la visualización de la geometría. Aunque encontrar un conjunto de tiras óptimo a partir de una triangulación dada es un problema NP-completo [6], existen diversas soluciones que, aunque no óptimas, maximizan sus prestaciones siguiendo diversos criterios.

Entre los numerosos estudios realizados hasta la actualidad cabe destacar los métodos propuestos en [1][7][17] para generar tiras de forma estática, así como el propuesto por Stewart [16] para la generación dinámica de tiras. Los trabajos propuestos ofrecen diferencias en la rapidez de generación y visualización, en el uso de memoria o en la cantidad de tiras generadas que los hacen más adecuados para cada uso concreto. También es importante comentar los trabajos que realizan un uso óptimo de la cache de vértices de la GPU. En este sentido, en los últimos años han aparecido métodos como los de Hoppe [11] o el realizado por Nvidia [2], que ha creado su propia librería para encontrar tiras que sacan el máximo provecho de la cache de vértices y de la localidad espacial de los buffers de vértices.

Por último, referenciar el algoritmo propuesto por Belmonte *et al.* [4] en el que, como en nuestro método, también se propone la generación de tiras siguiendo un criterio de simplificación. Pero en este caso, como ocurre en el resto de algoritmos, la generación partiendo del máximo nivel de detalle y su posterior simplificación provoca que las tiras que se obtienen en niveles de bajo detalle estén degeneradas, llegando a suponer un coste en vértices enviados a la *pipeline* mayor que enviar la propia malla de triángulos.

2.2. Métodos multirresolución basados en tiras

Uno de los primeros modelos en aprovechar la primitiva tira de triángulos fue el propuesto por Hoppe [10], conocido como Progressive Meshes e incluido en la librería DirectX de Microsoft. Este modelo utiliza triángulos durante el cambio de nivel de detalle pero construye tiras en el instante previo a su visualización. Posteriormente, El-Sana *et al.* [5] presentaron el modelo Skip Strips, que fue el primer modelo en mantener una estructura de datos para almacenar las tiras, evitando tener que calcularlas en tiempo real. Pero este modelo sigue utilizando triángulos para realizar el ajuste de la geometría en cada nivel de detalle.

El modelo MTS [3] utiliza las tiras de triángulos tanto como primitiva de almacenamiento como de visualización. Se

compone de un conjunto de tiras multirresolución, de manera que cada una de ellas representa una tira de triángulos y todos sus niveles de detalle, codificado como un grafo; sólo las que se modifican al cambiar de nivel de detalle se actualizan antes de ser dibujadas. Más adelante apareció el modelo Dstrips [15], que intenta mantener las tiras calculadas inicialmente, modificando las existentes y buscando nuevas sólo cuando una zona concreta del modelo lo requiere. También recientemente se ha presentado Lodstrips [13], que utiliza las tiras de triángulos tanto en la estructura de datos como en la visualización. Es sencillo de implementar y es eficiente y rápido al extraer el nivel de detalle, lo que permite obtener transiciones suaves entre los diferentes niveles de detalle. En la actualidad, es el modelo multirresolución que ofrece unas mejores prestaciones tanto en memoria ocupada como en rapidez de extracción y de visualización.

3. Algoritmo propuesto

El objetivo del algoritmo que presentamos es encontrar, a partir de una superficie poligonal triangularizada, tiras de triángulos que sean óptimas para modelos multirresolución que utilizan la operación de contracción de aristas en el proceso de simplificación de la malla. Con el método que proponemos se pretende conseguir que las tiras encontradas, al simplificarse, no sean cortadas ya que las simplificaciones se realizarán siempre a lo largo de aristas frontera entre dos tiras.

Con este propósito, partiremos de la malla simplificada al mínimo nivel de detalle utilizando un algoritmo de simplificación por contracción de aristas sin inserción de vértices, de forma que cada uno de los triángulos de la malla conformará una única tira. Asimismo, necesitaremos disponer del historial de simplificaciones en el que, para cada paso, se conozcan los dos triángulos que desaparecen, los vértices que se colapsan y los triángulos que se ven modificados. En nuestro caso, dado que seguiremos esta información en orden inverso, cada uno de los pasos representará la expansión de un vértice y la aparición de dos nuevos triángulos, además de la modificación de un conjunto de triángulos ya existentes. De esta forma, en contraposición con la contracción de aristas con la que trabaja el método de simplificación, nosotros trabajaremos con el concepto de expansión de aristas.

En general, podemos considerar que existen dos casos principales. El primero de ellos, mostrado en la Figura 2, representa un refinamiento a lo largo de una arista frontera entre las dos tiras que se muestran. En este paso, dos triángulos se verán modificados y se insertarán un total de cuatro nuevos vértices para dos nuevos triángulos, ya que en cada una de las tiras aparecerá un swap. De esta forma, las tiras estarán inicialmente formadas por los vértices *1, 2, 3, 4* y *1, 5, 3, 6*. Tras la expansión, quedarán formadas por los vértices *1, 2, 7, 2, 3, 4* y *1, 5, 7, 5, 3, 6*. Comentar que también es posible, aunque poco probable, encontrar este mismo caso pero de forma que sólo se modifique una única tira, siendo esto únicamente viable si la tira presenta un abanico.

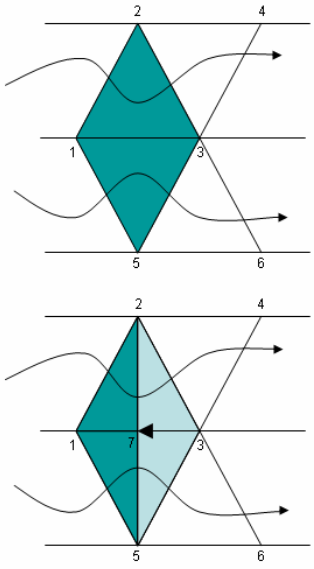

Figura 2. Expansión de aristas a lo largo de una arista frontera entre dos tiras.

El segundo caso se puede observar en la Figura 3. Presenta una expansión a lo largo de una arista que no es frontera. Esto provoca que sea imposible la inserción de los dos nuevos triángulos en la tira existente sin recurrir a triángulos degenerados que incrementarían el número de vértices enviados. En este caso es necesaria la creación de una nueva tira, ya que

sólo podremos insertar uno de los nuevos triángulos en la tira existente. La inserción de estos dos nuevos triángulos supondrá un incremento de cinco unidades en el total de vértices. Así, la tira que inicialmente estaba formada por los vértices *1, 2, 3, 4, 5, 6* pasará a contener las aristas *1, 2, 3, 2, 7, 4, 5, 6* y aparecerá una nueva tira con los vértices *3, 7, 5*.

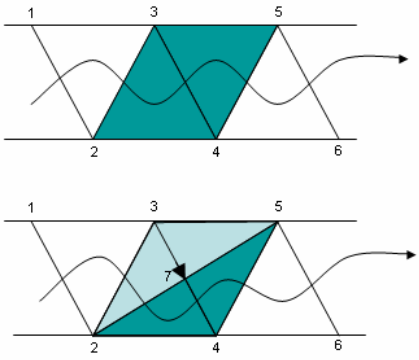

Figura 3. Expansión de aristas a lo largo de una arista no frontera dentro de una tira.

Con los dos casos generales comentados, el método que utilizaremos para encontrar las tiras será similar al presentado en el Algoritmo 1. Los dos nuevos triángulos comparten una arista con uno de los triángulos que se modificarán, por lo que a través de esta arista localizaremos el triángulo o triángulos donde podremos insertarlos. Una vez localizada la arista, simplemente insertaremos el nuevo triángulo con la precaución de escoger el lado correcto de la arista que nos había servido de guía. Si no encontramos esa arista, nos veremos obligados a crear una nueva tira. Por último, siempre deberemos comprobar que todos los triángulos modificados han sido correctamente cambiados. Debemos respetar todos los cambios que supone cada uno de los pasos de la simplificación, ya que de otra manera no obtendremos el modelo poligonal correcto al llegar al máximo nivel de detalle.

```
si encontramos_arista() hacer
   elegir_lado();
   insertar_triángulo();
sino hacer
   crear_tira();
comprobar_modificaciones();
```

Algoritmo 1. Algoritmo de generación de tiras.

3.1. Optimizaciones

Con el algoritmo propuesto hasta este momento, la mayor parte de los pasos suponen la inserción de cuatro nuevos vértices para los dos nuevos triángulos. Estas cuatro inserciones permiten un ahorro de un 30% respecto a los tres vértices por triángulo que serían necesarios si representáramos el modelo con una malla de triángulos. Para mejorar estos resultados, se ha completado el algoritmo de forma que, en cada uno de los pasos, se tiene la precaución de no insertar vértices o aristas repetidas ni aristas innecesarias. Además, es posible mejorar aún más los resultados si, cada vez que nos vemos en la obligación de crear una nueva tira, intentamos insertar el nuevo triángulo en el extremo de alguna tira ya existente.

En muchas ocasiones nos vemos en la necesidad de insertar un nuevo triángulo como una nueva tira. En sucesivas iteraciones puede que tengamos que añadir un nuevo triángulo al lado de éste pero, en función de cómo lo hayamos insertado, podremos hacerlo o no. Esto se debe a que al insertar un nuevo triángulo habrá una de sus tres aristas que no quede explícitamente reflejada en la tira y, por lo tanto, no la podremos encontrar en la búsqueda de aristas de nuestro algoritmo. Para intentar evitar este problema, se ha desarrollado una función encargada de hacer una predicción de la utilidad que cada una de las tres aristas del triángulo pueda llegar a tener, siguiendo su evolución a lo largo del refinamiento restante. Esta información nos ayudará a decidir cuál de las tres aristas es más prescindible a la hora de insertar el nuevo triángulo como una nueva tira. En este punto, debemos decidir si es mejor eliminar la arista que se va a utilizar más pronto, o la que se utilizará más tarde. Nuestros experimentos han demostrado que penalizar la arista que utilizaremos más pronto ofrece mejores resultados, al permitir que los nuevos triángulos se puedan insertar en tiras ya existentes en los últimos pasos del proceso.

Así, con todas las mejoras presentadas y como veremos en los resultados, podremos aumentar el porcentaje de ahorro hasta cerca del 50% respecto a enviar al hardware gráfico la información de la correspondiente malla de triángulos.

4. Resultados

Para analizar la tiras generadas como resultado de nuestro algoritmo se ha realizado un estudio de los vértices enviados a la tarjeta gráfica para diferentes niveles de detalle. Se han comparado estos datos con los obtenidos utilizando una representación de cada uno de los niveles de detalle mediante una simple malla de triángulos y con un modelo multirresolución de tiras basado en la utilización de un método de simplificación por contracción de aristas, como podría ser Skip-Strips [5], MTS [3] o LodStrips [13]. Los experimentos se han realizado utilizando el sistema operativo Windows XP sobre una máquina Dell con procesador a 2.8Ghz., 1 GB de RAM y una tarjeta gráfica Nvidia GeForce 6600 con 256 MB de RAM.

Las Figuras 4, 5 y 6 ofrecen resultados para tres modelos poligonales diferentes. En ellas se ofrece el número de vértices enviados al procesador gráfico para cada nivel de detalle, considerando el 100% como el máximo detalle y el 0% como el mínimo, aunque en las pruebas se ha utilizado el 10% como mínimo ya que con un nivel de detalle menor el modelo pierde totalmente su forma original. Comentar que la información señalada en las figuras con el nombre de *Triángulos* hace referencia a un modelo que utilice mallas de triángulos, *MMT* es una abreviatura de modelo multirresolución de tiras y que las *Tiras* son el resultado de el algoritmo propuesto en este artículo.

Como era de esperar, nuestro algoritmo envía menos vértices que una malla de triángulos. Se puede observar también cómo para niveles altos de detalle el modelo multirresolución de ejemplo envía menos vértices. Pero si consideramos el total de vértices necesarios para pasar por todos los niveles de detalle, desde el mínimo hasta el máximo, nuestro método envía menos vértices, como se muestra en la Tabla 1.

En la Figura 7 se muestran las tiras obtenidas como resultado de la aplicación del algoritmo presentado al modelo vaca. Además del conjunto de tiras que representan al modelo a máximo nivel de detalle, se ofrecen dos imágenes de las tiras obtenidas durante el proceso para un 33% y un 66% de detalle.

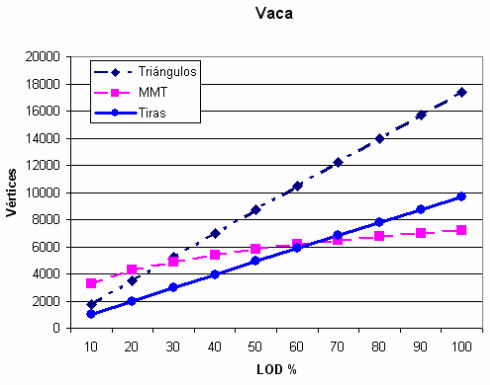

Figura 4. Resultados para el modelo vaca.

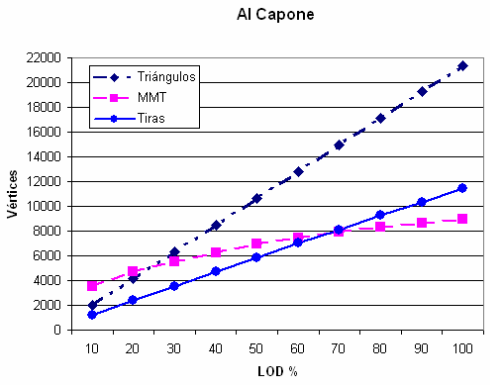

Figura 5. Resultados para el modelo Al Capone.

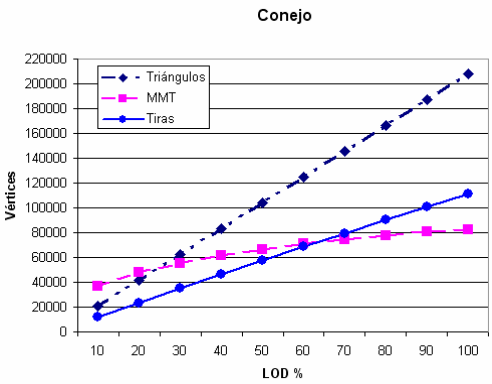

Figura 6. Resultados para el modelo conejo.

	Triángulos	MMT	Tiras
Vaca (5804 triángulos)	25.272.930 (100%)	15.752.211 (62.3%)	13.777.800 (54.5%)
Al Capone (7124 triángulos)	38.127.687 (100%)	23.158.015 (60.7%)	20.883.500 (54.7%)
Conejo (69451 triángulos)	3.619.981.407 (100%)	2.163.219.828 (59.7%)	1.976.880.000 (54.6%)

Tabla 1. Resultados en número total de vértices enviados pasando del mínimo al máximo nivel de detalle.

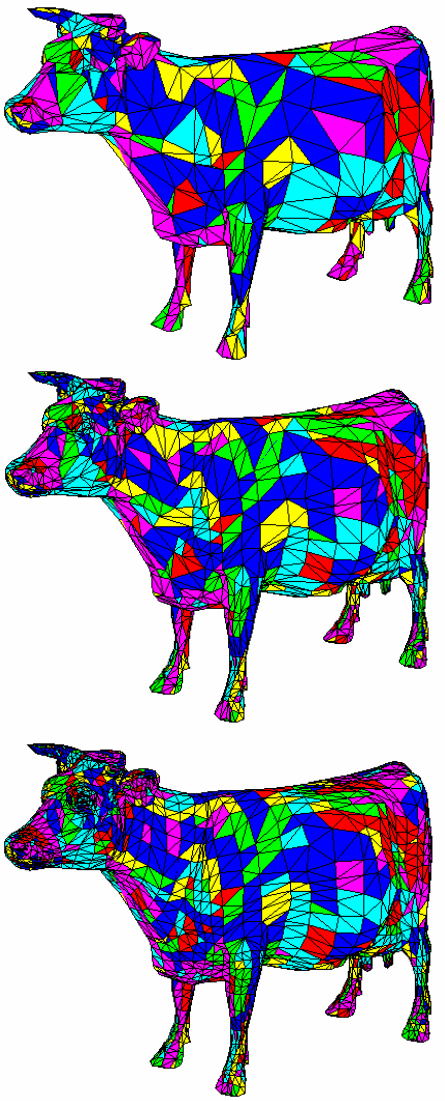

Figura 7. Tiras encontradas en el modelo vaca para un 33%, un 66% y un 100% de detalle respectivamente.

5. Conclusiones y trabajo futuro

Se ha propuesto un método de búsqueda de tiras en el que se consigue un conjunto de tiras de triángulos que no serán cortadas durante el proceso de simplificación, ya que únicamente se contraerán aristas frontera. Como era de esperar, conseguimos mejorar los resultados ofrecidos por los algoritmos de búsqueda de tiras de triángulos existentes, ya que todos ofrecen una baja calidad para niveles de poco detalle. En contraposición, nuestro algoritmo necesita un mayor número de tiras de triángulos en niveles de alto detalle. Pero, en general, la cantidad total de vértices enviados pasando del mínimo al máximo nivel de detalle es de un 15% menos que el modelo multirresolución con el que nos hemos comparado y supone un ahorro en torno a un 50% respecto a la malla de triángulos original. También a través de los resultados se puede observar cómo, a medida que disminuimos el nivel de detalle, las tiras que utiliza el modelo multirresolución con el que nos comparamos van empeorando llegando un punto, cercano al 20% de detalle, dónde incluso es mejor utilizar la malla de triángulos en lugar de las tiras que el modelo ofrece. Esto viene a probar la razón que ha impulsado la investigación de este nuevo algoritmo de búsqueda de tiras.

De los resultados obtenidos se puede desprender además que nuestra propuesta pierde calidad a medida que los modelos aumentan su tamaño. En muchas aplicaciones como los juegos, se suele trabajar con modelos no tan complejos como los analizados, donde es sencillo suplir esta baja complejidad poligonal con un correcto tratamiento de iluminación u otros aspectos de la visualización de la geometría.

Como dirección en nuestro trabajo futuro, resultaría interesante enfocar un posterior estudio a la disminución de las tiras generadas para altos niveles de detalle. De todas maneras, también se ha de tener en cuenta que no siempre las tiras más largas son mejores, como demostró Hoppe [11]. Así, la gestión correcta de la cache de vértices es

una cuestión muy importante para posibles mejoras del algoritmo. De la misma manera, consideramos que se puede conseguir una importante mejora utilizando un sistema de predicción más ajustado a la evolución de la simplificación, ya que con este sencillo método de predicción ya se consigue una mejora de un 5% en el número de vértices enviados. Por último, resultaría interesante combinar en un mismo método de búsqueda tanto la optimización de niveles de detalle altos como de niveles bajos. Para ello, sería aconsejable en futuros estudios considerar la búsqueda de tiras a partir de un nivel de detalle intermedio.

6. Agradecimientos

Este trabajo ha sido financiado por el gobierno español (TIN2004-07451-C03-03 y FIT- 350101-2004-15) y la Unión Europea (IST-2-004363 y fondos FEDER).

Referencias

[1] Akeley, K., Haeberli, P., Burns, D. tomesh.c: C Program on SGI Developer's Toolbox CD, 1990.

[2] Beeson, C., Demer, J. Nvtristrip, library version. Software available via Internet website. http://developer.nvidia.com/view.asp?IO=nvtristrip_library. January 2002.

[3] Belmonte, O., Remolar, I., Ribelles, J., Chover, M., Fernández, M., Rebollo, C. Multiresolution Triangle Strips. Proc. of Visualization, Imaging and Image Processing (VIIP 2001), ISBN/ISSN 0-88986-309-1, Marbella (Spain), pp. 182-187.

[4] Belmonte, O., Ribelles, J., Remolar, I., Chover, M. Búsqueda de tiras de triángulos guiadas por un criterio de simplificación. Actas del X Congreso Español de Informática Gráfica (CEIG 2000), ISBN/ISSN 84-8021-314-0, Castellón (Spain), pp. 51-64.

[5] El-sana, J. et al. Skip Strips: Maintaining Triangle Strips for View-dependent Rendering. Proceedings of Visualization 99, pp 131-137.

[6] Evans, F., Skiena, S., Varshney, A. Efficiently generating triangle strips for fast rendering. Technical report, Department of Computer Science, State University of New York at Stonyt Brook, Stony Brook, NY 11794-4400, USA, March 1997.

[7] Evans, F., Skiena, S., Varshney, A. Optimising Triangle Strips for Fast Rendering, IEEE Visualization'96, pp. 319-326,1996. http://www.cs.sunysb.edu/~stripe

[8] Funkhauser et al. Management of large amounts of data in interactive building walkthroughs. Proceedings of the 1992 symposium on Interactive 3D graphics, pp. 11 - 20.

[9] Garland, M. Multiresolution modeling: Survey & Future opportunities. State of the Art Reports of EUROGRAPHICS'99, pp 111-131, 1999.

[10] Hoppe, H. Progressive messhes. ACM SIGGRAPH 1996, pp 99-108.

[11] Hoppe, H. Optimization of Mesh Locality for Transparent Vertex Caching. ACM SIGGRAPH 1999, pp. 269-276.

[12] Puppo, E., Scopigno, R. Simplification, LOD, and Multiresolution - Principles and Applications. EUROGRAPHICS'97 Tutorial Notes (ISSN 1017-4656). Eurographics Association, Aire-la-Ville (CH), 1997 (PS97 TN4).

[13] Ramos, F., Chover, M. LodStrips. Lecture notes in Computer Science, Proc. of Computational Science ICCS 2004, Springer, ISBN/ISSN 3-540-22129-8, Krakow (Poland), vol. 3039, pp. 107-114.

[14] Ribelles, J., López, A., Belmonte, O., Remolar, I., Chover, M. Multiresolution modeling of arbitrary polygonal surfaces: a characterization. Computers & Graphics, vol. 26:3, pp. 449-462, ISSN 0097-8493. June 2002.

[15] Shafae, M., Pajarola, R. Dstrips: Dynamic Triangle Strips for Real-Time Mesh Simplification and Rendering. Proceedings Pacific Graphics Conference 2003.

[16] Stewart, J. Tunneling for Triangle Strips in Continuous Level-of-Detail Meshes. Graphics Interface, 2001, pp. 91-100.

[17] Xiang, X., Held, M., Mitchell, P. Fast and Effective Stripification of Polygonal Surface Models. SODA: ACM-SIAM Symposium on Discrete Algorithms, 1999.