Recuperando el detalle y evaluando la calidad visual en GPU de modelos topográficos simplificados

María V. Cifuentes[a,b], Juan P. D'Amato[a,c], Lucas Lo Vercio[a], Marcelo J. Vénere[a,d]

[a]PLADEMA, Universidad Nacional del Centro, 7000 Tandil, Argentina,
({cifuente,jpdamato }@exa.unicen.edu.ar)
[b] Comisión de Investigaciones Científicas de la Provincia de Buenos Aires
[c] Consejo Nacional de Investigaciones Científicas
[d] CNEA

Palabras Clave: Computer Graphics, Simplificación Topográfica.

Resumen. *La visualización interactiva de escenarios 3D es clave en la industria de entretenimientos, en multimedia y en otras aplicaciones de computer graphics. A la hora del rendering, el volumen y la complejidad de la información que define a cada ambiente resulta problemático. Por lo tanto, es necesario recurrir a técnicas que reduzcan los tiempos de rendering tales como la simplificación de modelos, el nivel de detalle adecuado para ciertas porciones del modelo, el uso de primitivas eficientes que aceleren el hardware gráfico, el balance entre CPU-GPU, entre otras. En este trabajo, presentamos una línea de investigación que propone un conjunto de técnicas las cuales han sido desarrolladas para ofrecer una alta performance del rendering de grandes terrenos en tiempo real.*

1 Introducción

Actualmente, para incrementar el realismo de las aplicaciones gráficas, librerías tales como OSG (http://www.openscenegraph.org/projects/osg) u OpenInventor (http://oss.sgi.com/projects/inventor/index.html), y motores de juegos como Torque o CryEngine incorporan diferentes técnicas de simplificación basadas en otorgar cierto nivel de detalle a cada parte del modelo, adaptando el número de polígonos de los objetos a su importancia dentro del escenario. Resultando modelos aproximados que son ampliamente empleados en aplicaciones de computación gráfica debido a que reducen la cantidad de información geométrica enviada al sistema gráfico. No obstante, estas estrategias tienen algunas deficiencias en la calidad visual resultante, al perderse ciertas características de las mallas tales como la apariencia, y exigen además un alto uso de CPU para su procesamiento.
Básicamente, nuestra propuesta de investigación intenta balancear la relación CPU-GPU migrando parte del procesamiento a la GPU para agilizar los tiempos de rendering consumidos en la representación de modelos topográficos simplificados.

2 Simplificación de Modelos Topográficos

La literatura brinda métodos [1, 2, 3] y métricas [4,5] para la simplificación poligonal de superficies. Algunos, muy difundidos [6, 7] usan técnicas iterativas basadas en

contracción de arcos para simplificar los modelos y mantener el error de aproximación usando métricas cuadráticas. Además, existe otra variedad de algoritmos de simplificación aplicables a casos característicos.
En particular, los algoritmos que reducen el volumen de información original de terrenos han constituido nuestra principal motivación. Básicamente, el proceso consiste en construir aproximaciones que conservan la similitud visual original del terreno aunque representado a través de unos pocos polígonos. Indicadores de error basados en la geometría del objeto, la ubicación de la cámara, la dirección de vista y máscaras binarias de visibilidad construidas mediante algoritmos de *ray-casting*, son algunas de las estrategias estudiadas e implementadas [8]. Actualmente, nos motiva la búsqueda de nuevos indicadores y/o criterios de simplificación basados en principios similares a los aplicados en el cálculo de modelos de iluminación conocidos. Por ejemplo, la simplificación guiada por la divergencia de los vectores normales asociados a cada triángulo resulta una alternativa. Por otra parte, se busca mejorar la performance aprovechando las capacidades de las unidades de procesamiento gráfico actuales.

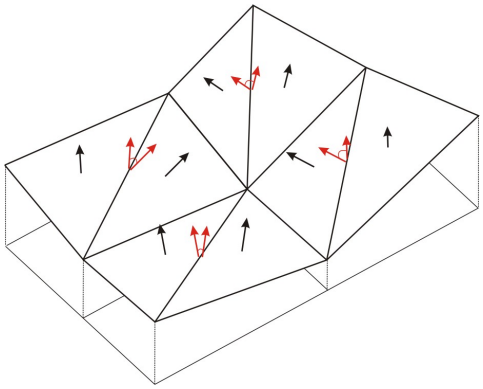

Figura 1- Normales y Divergencia correspondientes a una grilla regular de 4 elementos.

A partir del terreno original, el algoritmo de simplificación propuesto construye una jerarquía quadtree de mallas aproximadas de la que se pueden extraer aproximaciones de resolución variada. En una aplicación interactiva, la selección del nivel de detalle se realiza en tiempo de ejecución por lo que es necesario establecer un balance entre el número de polígonos con los que se representará el objeto y la cantidad de tiempo necesario para visualizarlo.

3 Criterios de similitud

Una evaluación de cuánto se parecen dos mallas, es medir el volumen encerrado entre la original y la simplificada. Para esto, se utiliza el volumen del prisma encerrado entre el centro del triángulo simplificado y el centro del triángulo original. En los casos de grillas regulares, el algoritmo es muy eficiente con un costo constante de cálculo por elemento, por acceder directamente a las posiciones de los triángulos originales. El siguiente pseudocódigo esquematiza el proceso:

para todo nodo terminal R_{aprox} ***hacer***

para todo triángulo $\hat{t} \in R_{orig}$ ***hacer***

determinar el centroide $p_c(x_c, y_c, z_c)$ de $\hat{t}$

identificar el triángulo $\hat{T} \in R_{aprox}$ que contiene al centroide p_c

calcular la componente z_{aprox} de p_c en $\hat{T} \in R_{aprox}$ (interpolación)

calcular $z_c - z_{aprox}$

acumular la diferencia de volumen para medir el error

fin para

fin para

Otra estrategia, aplicable a mallas de superficie no orientadas no regulares es la propuesta por [9] basada en una discretización del espacio mediante una grilla bidimensional y donde luego se computan intersecciones entre rayos provenientes de dicha grilla y los polígonos de las mallas comparadas. El volumen de diferencia se calcula como la suma de las distancias entre los triángulos intersecados.
Para el caso particular de mallas de altura, es posible realizar el mismo cálculo utilizando el buffer de profundidad, provisto por las GPU. Esto implica una mejora drástica de la performance, siendo el costo resultante el de recorrer los píxeles de dicho búffer. Esta estrategia, actualmente en evaluación, se encuentra limitada por la precisión de las GPU's y se requiere tener mucho cuidado en la medición de los datos.

4 Balance CPU-GPU

En cada frame, la malla aproximada extraída desde la jerarquía quadtree es enviada a la placa gráfica para su renderizado. OpenGL brinda primitivas especiales para renderizar tiras de cuadrantes, pero el número de vértices necesarios puede superar al usado por las tiras de triángulos. Debido a que la performance de las placas gráficas está supeditada a la cantidad de datos intercambiados entre CPU-GPU, se opta por triangular los cuadrantes y luego conectarlos en tiras de triángulos para reducir la cantidad de datos a transmitir, acelerando la etapa de rendering. Las tiras de triángulos mejoran notablemente los tiempos comparadas a las listas de triángulos, pero encontrar un conjunto adecuado de tiras de triángulos para el modelo aproximado [10] no es trivial y menos aún si es dinámico.
La idea es lograr un modelo aproximado con la calidad suficiente, pero utilizando la menor cantidad de polígonos, aprovechando además que las GPU trabajan eficientemente con texturas. Las GPU actuales permiten programar el pipeline de renderizado, a nivel de vértice y de textura, a través de un lenguaje gráfico denominado GLSL. Con estas capacidades, se propone una estrategia similar a un *bumpmapping* para reducir la cantidad de polígonos y al mismo tiempo mantener la apariencia de la malla.
En las mallas de superficie que analizamos, nos encontramos casos muy característicos, donde pequeñas variaciones de un vértice, ya sea en altura o en la normal, promueven la

generación en el árbol de muchos elementos, en general innecesarios. Estas pequeñas variaciones son los que dan un aspecto rugoso a la superficie, visible al aplicar el modelo de iluminación correspondiente. Es posible eliminar dichos vértices, si se permite un mayor error de volumen, pero con la consiguiente pérdida del aspecto aunque preservando las características principales como el lecho del río y los sectores más elevados (ver figura 2).

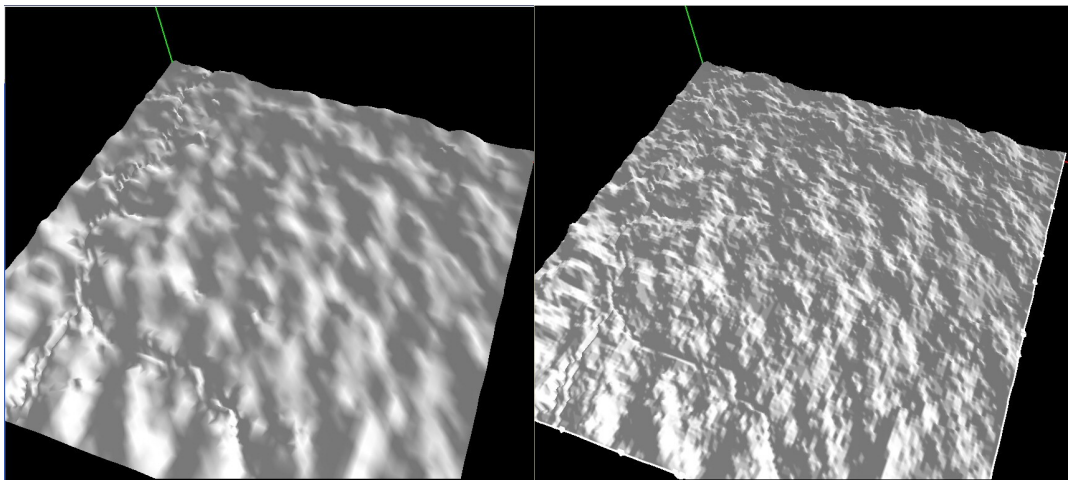

Figura 2- Terreno con baja y alta resolución. Se mantienen , izquierda y derecha respectivamente.

La propuesta para estos casos, es mantener el aspecto visual de la malla utilizando una aproximación de baja resolución combinada con la información asociada a los vectores normales del modelo original, permitir una cierta pérdida de volumen sin desmerecer la calidad visual de su apariencia. El proceso implica almacenar la información de las normales de la malla original codificada en una textura, sabiendo que esta información es inalterable se reduce la transferencia CPU-GPU. Al momento de la simplificación, se relaja la condición de volumen; es decir, se aplica un criterio de planaridad adecuado a la superficie que elimina las irregularidades de la malla (picos o serruchos). Finalmente al renderizar, en la etapa por píxel, el modelo de iluminación se calcula en base a las normales de la malla original. En la figura 3 se puede observar como aparecen los detalles sobre la superficie cuando se aplica el mapa de normales (ver figura 3). Esta estrategia brinda la facilidad adicional de no recalcular las normales en el modelo simplificado.

5 Conclusiones y Trabajos Futuros

La idea es continuar ahondado en la búsqueda de técnicas y estrategias de simplificación y algoritmos eficientes de rendering. Por un lado, construir tiras de triángulos que reduzcan la cantidad de vértices que se envían a la placa y que eviten la existencia de triángulos degenerados. Sumado a esto, se está estudiando un criterio unificado para medir la similitud de la malla resultante, basado en la medición de volumen y en el cambio visual.

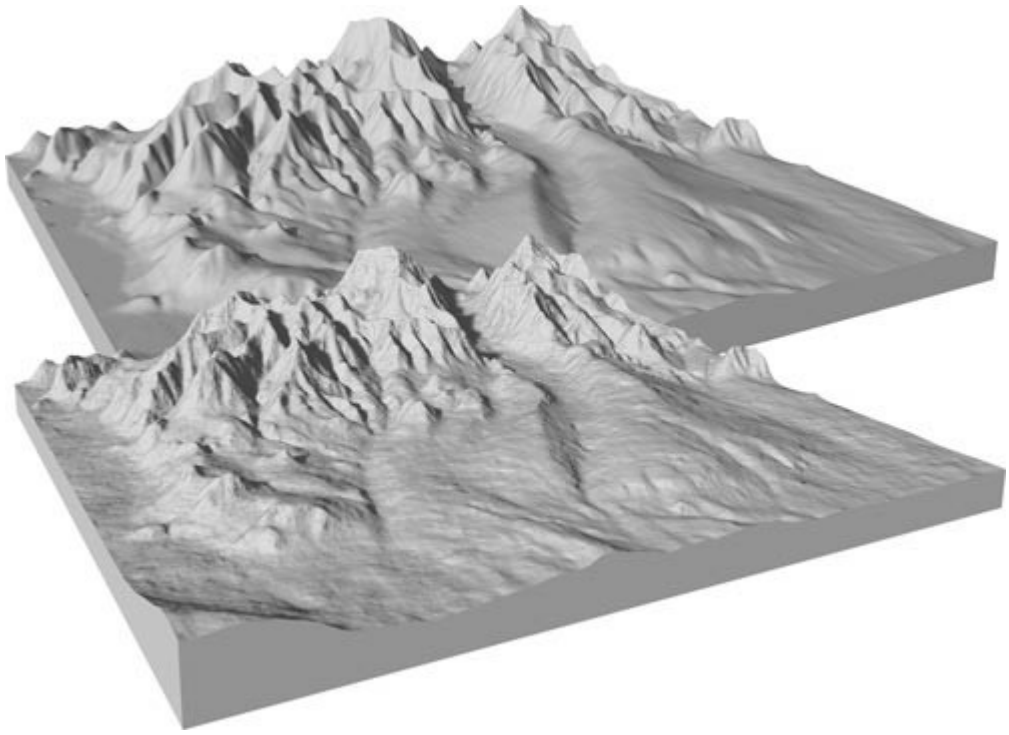

Figura 3 – Rendering de un terreno de baja resolución. Abajo se aplica el mapa de normales.

Referencias

[1] D.P. Luebke. A developer's survey of polygonal simplification algorithms. IEEE Computer Graphics and Applications, 3(24):24–35, 2001.

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

[3] Pai-Feng Lee, Bin-Shyan Jong, Point-based Simplification Algorithm, WSEAS Transactions on Computer Research, vol 1(3), pp. 61-66, 2008.

[4] P. Castelló, M. Sbert, M. Chover, M. Feixas, Techniques for Computing Viewpoint Entropy of a 3D Scene, ICCS 2006, University of Reading, UK, May, 2006.

[5] R. Alvarez, L. Tortosa, J. F. Vicent, A. Zamora, Error measurements and parameters choice in the GNG3D model for mesh simplification, WSEAS Transactions on Information Science & Applications, vol. 5 (5), pp. 579-588, 2008.

[6] E. Puppo and R. Scopigno. Simplification, lod and multiresolution - principles and applications. Tutorial Notes of EUROGRAPHICS'99, 16(3), 1997.

[7] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pp. 209–216. ACM Press/Addison- Wesley Publishing Co., 1997.

[8] M.V. Cifuentes, J.P. D'Amato, C. Garcia Bauza, M.J. Vénere, Simplificación de modelos topográficos, IX Workshop de Investigadores en Ciencias de la Computación, Trelew, Argentina, 2007.

[9] Juan D'Amato, Mariana del Fresno, Marcelo Javier Vénere, Un indicador de calidad para evaluar superficies segmentadas", ENIEF, 2008.

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