Cálculo de la triangulación de Delaunay en la GPU *

Jon Valdés Furriel **

Resumen

En este trabajo presentamos una técnica simple y rápida para obtener triangulaciones Delaunay de nubes de puntos 2D, aprovechando las capacidades de las tarjetas gráficas actuales. El método se basa en el cálculo gráfico del diagrama de Voronoi y la posterior extracción de la triangulación a partir de éste.

1 Introducción

Las triangulaciones de Delaunay son una herramienta muy útil para generar mallas irregulares a partir de un conjunto de puntos. Existen buenos algoritmos para su cálculo que han sido implementados con éxito (véase, por ejemplo, la librería CGAL, `www.cgal.org`). Sin embargo, en general, las implementaciones que se emplean no utilizan la potencia y capacidad de las tarjetas gráficas. En el desarrollo de una aplicación informática para modelizar la evolución de bosques (*Vorest* [1]) se calculan diagramas de Voronoi de forma gráfica como medio de solventar el problema del cálculo de diagramas de Voronoi generalizados con pesos tanto aditivos como multiplicativos. Por esta razón, se decidió en el desarollo del proyecto emplear esa misma herramienta gráfica para calcular las mallas que dan soporte a los modelos digitales de los terrenos.

En este trabajo presentamos una técnica simple y rápida para obtener triangulaciones Delaunay-correctas de nubes de puntos 2D, aprovechando las capacidades de las tarjetas gráficas actuales. Esta técnica da buenos resultados en la generación de mallas irregulares de terrenos obtenidas, como es frecuente, de nubes de puntos densas y homogéneas.

El método consiste en el cálculo gráfico del diagrama de Voronoi para los puntos y la posterior extracción de la triangulación a partir de este, aprovechando que la triangulación de Delaunay es el grafo dual del diagrama de Voronoi. Existen sin embargo varios puntos donde el proceso no resulta trivial, para los que veremos a continuación las soluciones adoptadas. Se indican así mismo los puntos débiles del método y las líneas de trabajo futuras.

2 Construcción gráfica del diagrama de Voronoi

La técnica clásica (presentada en [3]) para la generación de diagramas de Voronoi de una nube de puntos de forma gráfica, permite generar una imagen en la que cada pixel contiene un color que representa al punto de la nube más cercano a ese pixel. Para ello se genera en tres dimensiones un cono por cada punto, y se dibuja en vista ortográfica superior, utilizando el algoritmo de Z-buffer para determinar qué cono resulta el más cercano a la cámara en cada pixel.

Esta técnica sin embargo, resulta problemática en el hardware actual, ya que es por lo general incapaz de dibujar superficies curvas como las de un cono. Es por esto que la técnica aproxima los conos utilizando triángulos, con la consiguiente pérdida de precisión en la superficie, y los problemas de rendimiento provocados por tener que aumentar la cantidad de polígonos por cada cono para reducir

*Parcialmente subvencionado por el proyecto Consolider Ingenio 2010 i-MATH C3-0159

**`juanval@gmail.com`

los errores en el Voronoi. La técnica aquí presentada, sin embargo, permite dibujar superficies que se comportan como conos perfectos en el Z-buffer, y que a pesar de todo presentan un rendimiento excelente.

Nuestra técnica consiste en dibujar un cuadrado en el lugar de cada cono y calcular, en cada pixel cubierto por el cuadrado, la profundidad exacta que tendría un cono perfecto. Para esto, utilizando un *fragment shader* en la tarjeta gráfica, se calcula en cada pixel la distancia desde ese pixel hasta el punto de la nube al que corresponde, y se asigna ese valor como profundidad Z del pixel. A continuación, al igual que en [3], el Z-buffer se encarga de seleccionar la superficie cuyo punto central sea el más cercano a ese pixel.

En la Figura 1, podemos ver arriba el método clásico y la imagen que genera, con imperfecciones en las aristas del diagrama de Voronoi. Debajo vemos las superficies que aparecen al calcular la profundidad de la superficie del cono en cada pixel, y cómo los diagramas de Voronoi generados no presentan más imperfecciones que las propias de la discretización.

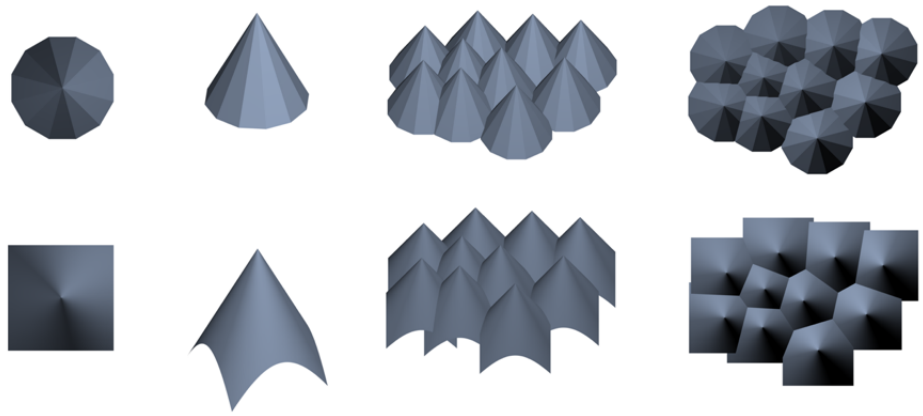

Figura 1: Arriba, el método clásico, con los conos triangulados, genera imperfecciones en el diagrama de Voronoi. Abajo, nuestro método genera superficies cónicas exactas en cada pixel, y el diagrama de Voronoi aparece libre de imperfecciones.

Por otro lado, las mayores texturas con las que puede trabajar una tarjeta gráfica actual son generalmente de 8192 pixels de lado, y esto puede ser insuficiente para representar el diagrama de Voronoi de una nube suficientemente grande. En este caso necesitaremos partir la imagen del diagrama en varias partes para conseguir suficiente resolución por cada celda.

Para esto, generamos varias imágenes contiguas en el espacio de los puntos, y a continuación las tratamos como secciones de una única imagen general.

3 Extracción de la triangulación

A continuación extraemos la imagen de la GPU, y realizamos el análisis del diagrama de Voronoi en la CPU.

A este efecto, en [2] se presenta una técnica para la extracción de la triangulación de Delaunay en 3D a partir de un diagrama de Voronoi 3D discretizado. Nuestra técnica es esencialmente la misma, trasladada a 2D.

Inicialmente localizamos en la imagen todos aquellos pixeles que tengan más de 1 color distinto al suyo propio entre sus 8-vecinos. A continuación, agrupamos estos pixeles seleccionados buscando las componentes 8-conexas entre ellos. Cada una de estas componentes conexas será un vértice del diagrama de Voronoi.

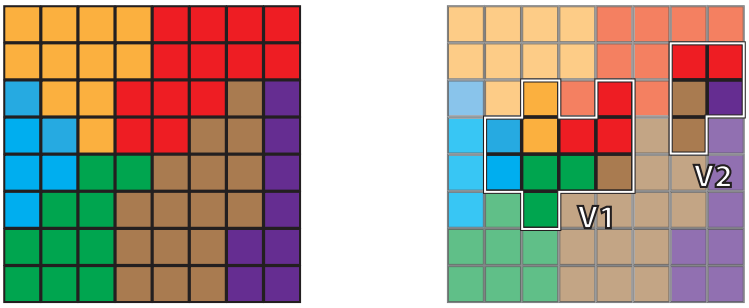

Figura 2: A la izquierda, imagen ampliada de un vértice del diagrama de Voronoi gráfico. A la derecha, extracción de las componentes conexas de los puntos vértice.

Finalmente, sabiendo que el diagrama de Voronoi es el grafo dual de la triangulación de Delaunay, cada uno de estos vértices generará una cara en la triangulación.

En la Figura 2 podemos ver un ejemplo de un diagrama de Voronoi gráfico en el cual se buscan las componentes 8-conexas de pixeles vértice, y se encuentran 2 vértices del diagrama de Voronoi: el vértice V1 aparece donde confluyen las caras roja, amarilla, marrón, verde y azul; y V2 aparece en la unión de las caras roja, marrón y morada. Esto supone que los puntos correspondientes a los colores de V1 formarán una cara en la triangulación de Delaunay, y los de V2 formarán otra.

Aquí aparece un problema evidente con los vértices del diagrama de Voronoi con más de 3 zonas incidentes, que suponen una cara del Delaunay de más de 3 lados. Esto sucederá siempre que existan más de 3 puntos situados en posición concéntrica. En ese caso, deberemos buscar por otro método una triangulación para estos puntos, pero su disposición obligatoriamente convexa permite simplificar considerablemente este proceso.

4 Resultados

Las triangulaciones obtenidas con este método, siempre que se cumplan las condiciones de densidad y resolución mínimas, son Delaunay-correctas en todos los casos probados.

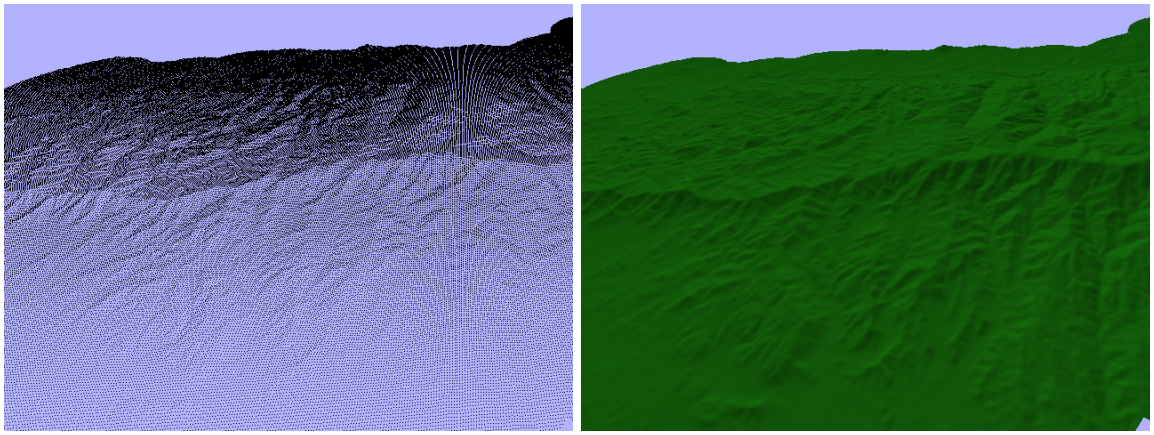

Figura 3: Nube de puntos de un terreno de la sierra de Filabres, y su correspondiente triangulación con normales e iluminación aplicadas.

En cuanto a rendimiento, en la máquina de pruebas (un Core 2 Duo a 2.4 GHz con 4GB RAM y una tarjeta gráfica GeForce 8600M GT), los tiempos encontrados para la nube de la Figura 3, de unos 260.000 puntos, son los siguientes:

1. Generación y extracción del diagrama de Voronoi y sus vértices: 3.93 segundos
2. Extracción de la triangulación a partir de los vértices del diagrama de Voronoi: 0.72 segundos

En total, 4.65 segundos para generar una triangulación con unos 530.000 triángulos.

Es importante tener en cuenta que éste es el caso óptimo para este algoritmo, con los puntos en disposición regular y uniforme. En otro caso sería necesario aumentar la resolución de las imágenes y el tamaño de los conos, lo que degradaría el rendimiento.

Existen así mismo varias oportunidades de optimización que no se han aprovechado, como la utilización de GPGPU para todo el proceso. Esto eliminaría la transmisión del diagrama de Voronoi entre la GPU y la CPU, y aceleraría también el proceso de las imágenes. En la implementación actual, se consumen aproximadamente 1.6 segundos por cada imagen en estos dos pasos.

4.1 Limitaciones del algoritmo

Este método presenta ciertas limitaciones para casos generales, tal y como veremos a continuación.

En primer lugar, la discretización del diagrama de Voronoi da lugar a problemas siempre que la resolución de la imagen no sea suficiente para mostrar con detalle la estructura de éste.

En nuestro caso, simplemente detectamos si dos vértices resultan demasiado cercanos en la imagen usando una técnica de bucketing, y solucionamos esto aumentando la resolución del diagrama de Voronoi mientras no desaparezca el problema. En caso de tener una nube de densidad no uniforme, este método presenta el problema de renderizar zonas a resolución innecesariamente alta. Métodos más sofisticados podrían reducir las zonas calculadas a alta resolución, mejorando el rendimiento notablemente.

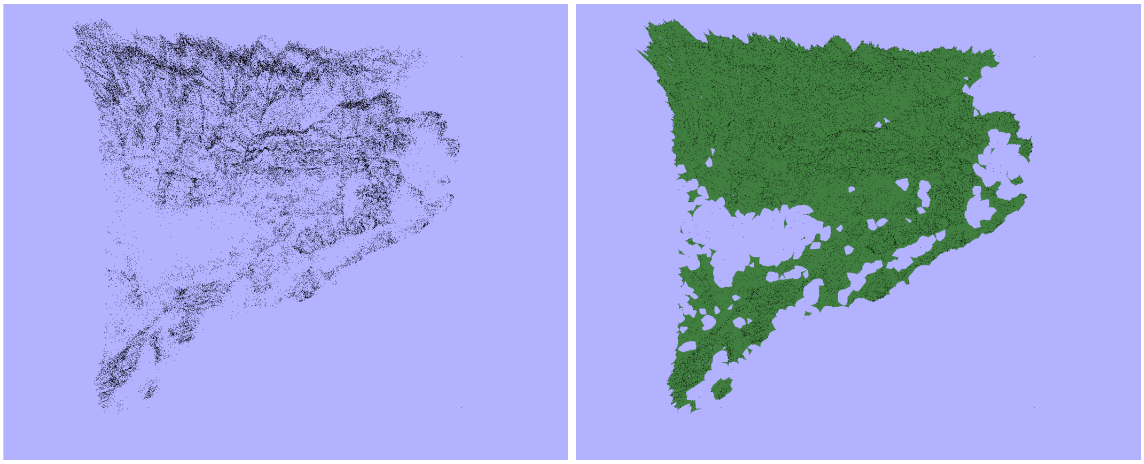

Figura 4: Aquí vemos una nube con zonas poco densas, que provocan el fallo en la triangulación.

Por otro lado, en la práctica resulta inviable asegurar que la triangulación encontrada es la triangulación de Delaunay completa de los puntos, ya que para esto los cuadrados utilizados deberían tener tamaño infinito. Esto supone que la triangulación encontrada siempre será Delaunay-correcta en el ambiente local definido por el tamaño de los cuadrados. Así mismo, la adyacencia (y la triangulación) entre puntos a mayor distancia que esta, no será encontrada por el algoritmo. Podemos ver un ejemplo de esto en la Figura 4

En la Figura 5 se aprecia un ejemplo de un diagrama de Voronoi que presenta ambos problemas, causados por una nube de puntos con zonas de densidades muy dispares. Mientras en las zonas densas es imposible generar una triangulación correcta, en las poco densas no es posible siquiera generar triangulación alguna.

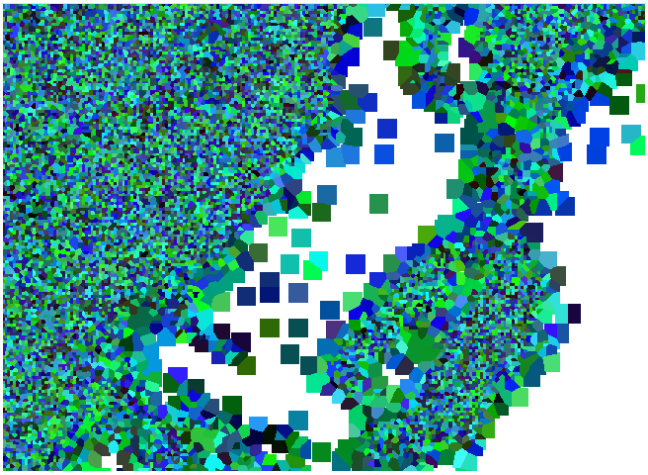

Figura 5: Las zonas de muy alta o baja densidad resultan problemáticas en el diagrama de Voronoi gráfico.

5 Conclusión y trabajo futuro

Este método presenta excelentes cualidades para mallas de densidad uniforme, consiguiendo un rendimiento comparable a otros métodos de triangulación. Esto es así a pesar de no haber utilizado el propio hardware gráfico para el análisis de las imágenes, lo que podría haber aumentado el rendimiento considerablemente.

Por el contrario, trabajando con mallas de densidad variable resulta necesario aumentar la resolución para que la celda más pequeña del diagrama de Voronoi tenga un número suficiente de pixeles, así como aumentar el tamaño de los cuadrados para que sean capaces de contener las mayores regiones del Voronoi. Esto provocará irremediablemente la pérdida de rendimiento en el proceso.

6 Agradecimientos

Este trabajo forma parte del proyecto *GeoVys* (Consolider Ingenio 2010 i-MATH C3-0159) desarrollado en colaboración entre las universidades Politécnica de Madrid y de Valladolid y dirigido por Manuel Abellanas.

Referencias

[1] Begoña Abellanas, Manuel Abellanas, Carlos Vilas, *VOREST: Modelización de bosques mediante diagramas de Voronoi* , Actas de los XII Encuentros de Geometría Computacional, Universidad de Valladolid, junio 2007. 249-256.

[2] Boltcheva, D.; Bechmann, D.; Thery, S.; 2007 *Discrete Delaunay: Boundary extraction from voxel objects*. 3-D Digital Imaging and Modeling, 2007. 3DIM '07. Sixth International Conference on 21-23 Page(s):209 - 216

[3] Hoff,III, Kenneth E. and Keyser,, John and Lin,, Ming and Manocha,, Dinesh and Culver,, Tim. 1999 *Fast computation of generalized Voronoi diagrams using graphics hardware*. In SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, New York, 277–286. Computer Graphics Proceedings, Annual Conference Series, ACM.

[4] Rong,, Guodong and Tan,, Tiow-Seng and Cao,, Thanh-Tung and Stephanus, 2008 *Computing two-dimensional Delaunay triangulation using graphics hardware* I3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games, Pages: 89–97, ACM.