TRIANGULACIONES

PARTE I: TRIANGULACIÓN DE UNA NUBE DE PUNTOS

Definición: *Dado un conjunto $S=\{s_1, ..., s_n\}$ de puntos de R^2, llamamos triangulación de S a una partición de su cierre convexo en triángulos, tal que cada punto de S sea un vértice de un triángulo y viceversa.*

Consideremos el siguiente algoritmo:

Entrada: **$S=\{s_1, ..., s_n\}$**

1. **Para cada par de vértices, añadir el segmento que los une si no corta a ningún segmento anterior.**

Este algoritmo voraz calcula una triangulación de S, con coste en tiempo $O(n^2)$. Sin embargo, la triangulación que proporciona dicho algoritmo no es útil en la mayoría de las aplicaciones, como por ejemplo en la interpolación de una función continua sobre una superficie.

Esta función puede representar un mapa de alturas, que se miden sólo en ciertos puntos (los de S), y que debe ser interpolada en el resto (esto es frecuente en los GIS). Podría darse una situación como la siguiente:

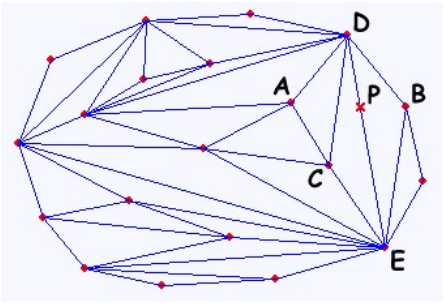


FIGURA 1

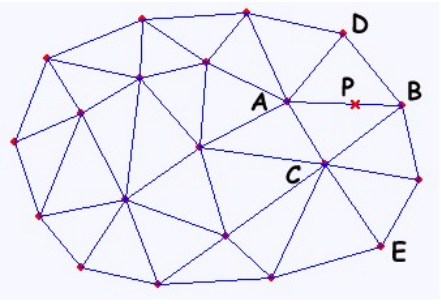


FIGURA 2

En la **figura 1** se calcula la altura de P interpolando las de C, D, E. Si las alturas medidas en estos puntos son 100m, 70m y 20m respectivamente, la altura en P resultaría 53m. Si las alturas en A y B son 100m y 110m respectivamente, este valor para P resulta inadecuado. Esto no ocurre en la triangulación de la **figura 2**, en la que el valor en P depende de A, B y C, obteniéndose una altura de 106m.

Definición: *Se dice que una triangulación T_1 es mejor que otra T_2, y se representa $T_2<=T_1$, si $A_{T2}<=A_{T1}$ (en orden lexicográfico), donde A_{T1} y A_{T2} son las listas de los ángulos ordenados de menor a mayor de T_1 y T_2 respectivamente.*

Definición: *Una triangulación T se dice equilátera si no existe otra triangulación T' tal que T<=T'.*

La **figura 2** es un ejemplo de triangulación equilátera, mientras que la de la **figura 1** no lo es.

Una triangulación equilátera reduce el número de ángulos muy agudos, por lo que es mejor en cálculos que requieran precisión. Por esto es frecuente que en muchas aplicaciones sea preferible una triangulación de este tipo a otra cualquiera.

La triangulación de una nube de puntos guarda una estrecha relación con el diagrama de Voronoi de dicha nube de puntos. Así, se tiene la siguiente

Definición: *Dado un conjunto $S=\{s_1, ..., s_n\}$ de puntos de R^2, el dual geométrico de su diagrama de Voronoi es una triangulación de los puntos de S. Dicha triangulación se llama triangulación de Delaunay.*

Es decir, la triangulación de Delaunay se obtiene uniendo vecinos de Voronoi. Posee varias propiedades importantes:

- *Minimiza el máximo radio de una circunferencia circunscrita.*
- *Maximiza el mínimo ángulo (de hecho, la triangulación de Delaunay es equilátera).*
- *Maximiza la suma de los radios de las circunferencias inscritas.*
- *La distancia entre cualquier par de vértices a través de aristas de la triangulación es, a lo sumo, una constante (2'42) por su distancia euclídea.*

Sin embargo, calcular la triangulación de Delaunay a través del diagrama de Voronoi es difícil (a pesar de que el método sea óptimo, con coste O(n log n)), por lo que conviene disponer de otras técnicas para conseguirla.

Uno de estos métodos consiste en usar la siguiente relación:

Proposición: *Sea $S=\{s_1, ..., s_n\}$ un conjunto de puntos de R^2. Si proyectamos S sobre un paraboloide con ecuación $z=x^2+y^2$ (esto es, asociamos a cada punto $s_i=(xi, y_i)$ del plano el punto $(x_i, y_i, x_i^2+y_i^2)$ de R^3), entonces la proyección sobre R^2 de la parte inferior del casco convexo de los puntos en el paraboloide es la triangulación de Delaunay de S.*

Así, el algoritmo sería el siguiente

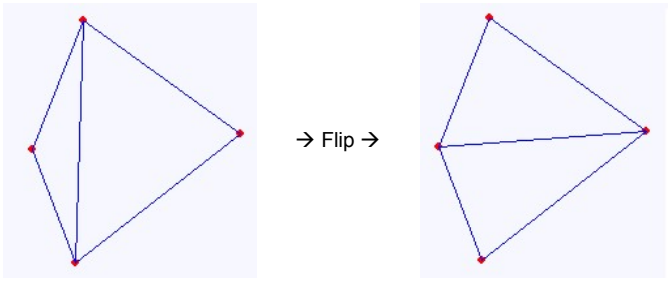
Entrada: **$S=\{s_1, ..., s_n\}$**

1. **Proyectar los puntos de S sobre el paraboloide $z=x^2+y^2$.**
2. **Calcular el casco convexo de los nuevos puntos.**
3. **Proyectar la parte inferior sobre el plano.**

Un método más fácil de calcular la triangulación de Delaunay, aunque con un coste $O(n^2)$, es mediante *giros* de aristas, modificando una triangulación inicial hasta llegar a la de Delaunay.

Definición: *Dados dos triángulos que comparten un lado, formando un cuadrilátero convexo, se dice que el lado común es legal si maximiza el ángulo mínimo. El lado común es ilegal en caso contrario.*

Definición: *Dados dos triángulos que comparten un lado, formando un cuadrilátero convexo, se denomina flip (giro) en la diagonal común al cambio de dicha diagonal por la otra en el cuadrilátero. Se dice que un flip es positivo si convierte una diagonal ilegal en legal.*



Ilegal Legal

Teorema: *Una triangulación es de Delaunay si y sólo si no contiene aristas ilegales.*

Además, para saber si una diagonal es legal basta comprobar si la circunferencia que pasa por los puntos de la misma y un punto cualquiera del cuadrilátero al que pertenece no contiene al otro punto. En caso contrario, la diagonal es ilegal.

Por tanto, un algoritmo para obtener la triangulación de Delaunay es el siguiente:

Entrada: **T, triangulación inicial cualquiera**

1. **Poner todas las aristas internas en una cola**
2. **Mientras la cola no esté vacía**
    1. **Sacar una arista, a, de la cola**
    2. **Si C_a tiene diagonal ilegal, hacer un flip positivo y añadir las aristas exteriores de C_a a la cola**

(C_a es el cuadrilátero con diagonal a)

APPLET 1

TRIANGULACIÓN DE DELAUNAY

Se puede disponer de una variante de este método, conocida como algoritmo incremental, en el que los puntos de la nube se introducen poco a poco, y la triangulación se optimiza tras cada inserción:

Entrada: **T, triangulación de Delaunay, P, nuevo punto, p_{-1}, p_{-2}, p_{-3}, triángulo que contiene a T**

1. **Encontrar el triángulo $p_ip_jp_k$ en T en el que se encuentra P**
2. **Si $p_ip_jp_k$ contiene a P**
    1. **Añadir las aristas de P a p_i, p_j, p_k**
    2. **Legalizar las aristas p_ip_j, p_ip_k, p_jp_k**
3. **Si no (está sobre una arista, p_ip_j por ejemplo)**
    1. **(Sea p_l el otro punto del triángulo que contiene a p_ip_j) Añadir aristas de P a p_i, p_j, p_k, p_l**
    2. **Legalizar las aristas p_ip_l, p_lp_j, p_jp_k p_kp_i**

(Legalizar la arista p_ip_j cuando se introduce p:

1. **Si p_ip_j es ilegal**
    1. **(Sea $p_ip_jp_k$ el triángulo adyacente a pp_ip_j) Reemplazar p_ip_j por pp_k**
    2. **Legalizar p_ip_k, p_jp_k)**

Existen muchos criterios de optimización. Uno de los más famosos se conoce como **triangulacón de peso mínimo**, y consiste en buscar una triangulación de la nube de puntos que minimice la longitud total de las aristas. No se conoce ningún algoritmo de coste polinomial que resuelva este problema, ni se sabe si es **NP-completo**. El algoritmo que proporciona la mejor aproximación, de Levcopoulos y Krznaric, da una solución con un factor multiplicativo constante de la longitud óptima. Otro criterio busca la triangulación que minimice la máxima longitud de las aristas. Edelsbrunner y Tan demostraron que esta triangulación contiene las aristas del árbol recubridor mínimo (Minimmum Spannig Tree, MST). Esto proporciona un algoritmo de coste polinomial: calcular el MST y después triangular los polígonos simples resultantes, usando programación dinámica.

PARTE II: TRIANGULACIÓN DE POLÍGONOS

Definición: *Dado un polígono P, una triangulación de P es una subdivisión de su interior en triángulos, tal que los vértices del polígono sean vértices de los triángulos y viceversa. Se denomina diagonales a los lados de los triángulos (de cualquier triangulación de P) que no son lados de P.*

La triangulación de un polígono es un problema fundamental en geometría computacional, y es ampliamente usada en la teselación de geometrías curvadas, como las descritas por splines (Kumar y Manocha, 1994). Tiene importancia como preprocesamiento en diversos algoritmos, como por ejemplo en el problema de la galería de arte:

Definición: *Decimos que un punto y es visible desde otro punto x en un polígono simple P si el segmento que los une está completamente contenido en el interior de P.*

Problema de la galería de arte: *Dado un polígono simple P, que representa la planta de un edificio, podemos considerar el siguiente problema: dónde poner 'guardias' para que cada punto del edificio esté vigilado por alguno de ellos.*

La solución a este problema, que se describe a continuación, pasa por calcular una triangulación de la planta del edificio.

Teorema (de la galería de arte): *Dado un polígono simple con n vértices, existe un conjunto de guardias que lo vigilan por completo con, a lo sumo, $\lfloor n/3 \rfloor$ guardias.*

Lema: *Sea T el grafo resultante de una triangulación de un polígono simple. Entonces T es 3-coloreable.*

Entonces, es posible triangular la planta del edificio y colorearla usando sólo 3 colores, que deben aparecer en cada uno de los triángulos. Por tanto, si se toman todos los vértices que tengan el color menos usado se vigila el edificio completo, ya que se vigila cada uno de los triángulos que lo componen.

Existen muchas aplicaciones en las que, al igual que el caso de nubes de puntos, es importante descomponer polígonos en triángulos con formas especiales (por ejemplo, evitando que tengan vértices muy agudos).

La triangulación restringida de Delaunay, proporciona un método para forzar la aparición de las aristas de un PSLG (Planar Straight-Line Graph), G, en la triangulación de Delaunay. Un triángulo *abc* aparece en la triangulación restringida de Delaunay si su circunferencia circunscrita no contiene ni pasa por ningún vértice de G visible desde cualquiera de los puntos de *abc*. Esta definición generaliza la de la triangulación de Delaunay en el caso de que G no contenga aristas. Si G es un polígono, entonces la triangulación restringida de Delaunay contiene sólo triángulos interiores a G.

Sin embargo también existen otras muchas aplicaciones en las que la forma de los triángulos es indiferente. Sólo consideraremos el problema de, dado un polígono simple, calcular una triángulación suya.

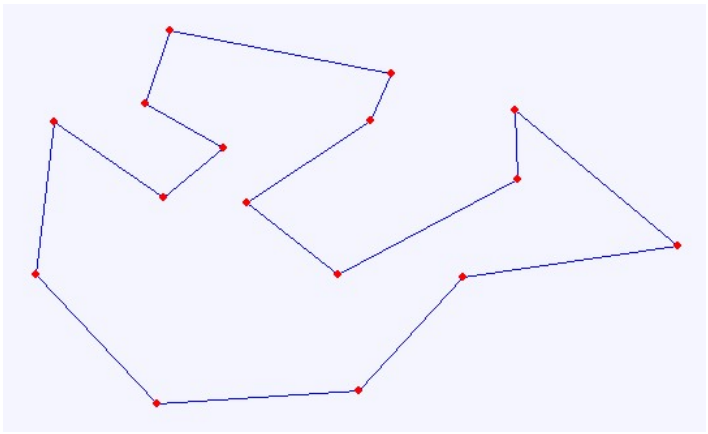
Es posible encontrar un algoritmo que resuelve este problema en tiempo polinomial, basado en añadir diagonales :

Lema: *Toda diagonal divide un polígono en dos, con menor número de vértices.*

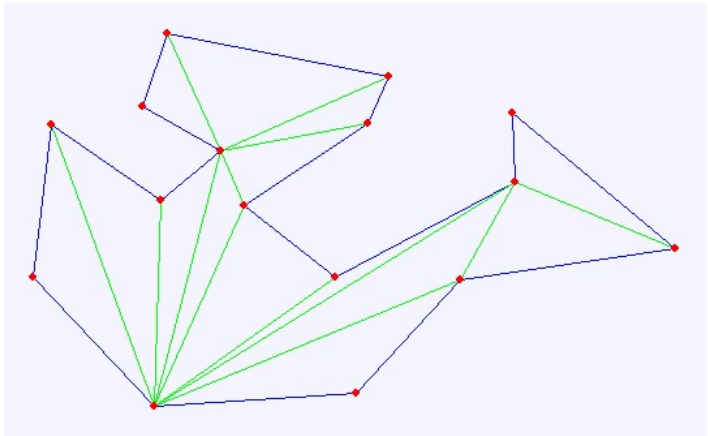
Lema: *Todo polígono con más de 3 vértices admite una diagonal.*

Teorema: *Todo polígono admite una triangulación.*

Esto proporciona un método para triangular polígonos con coste en tiempo $O(n^3)$.



ANTES



DESPUÉS

APPLET 2

TRIANGULACIÓN DE POLÍGONOS

Existen muchos algoritmos que resuelven el problema con coste en tiempo $O(n \cdot log\ n)$, pero un problema abierto durante mucho tiempo fue determinar si existe un algoritmo de coste $O(n)$. Este problema fue resuelto en 1991 por Chazelle, aunque el algoritmo resultó ser tan complicado que no puede competir con los algoritmos prácticos, aunque asintóticamente peores, de coste $O(n \cdot log\ n)$. El algoritmo de Chazelle reduce el problema de la triangulación de un polígono P al de calcular el *mapa de visibilidad horizontal* de P, esto es, la partición del polígono obtenida trazando líneas horizontales a izquierda y derecha de los vértices.

Si se busca optimizar propiedades se puede recurrir a los siguientes algoritmos:

PROPIEDAD	ALGORITMO	ORDEN
Delaunay	*Varios*	$O(n \cdot log\ n)$
Minimiza máximo ángulo	*Inserción rápida de aristas*	$O(n^2 \cdot log\ n)$
Minimiza máxima pendiente	*Inserción de aristas*	$O(n^3)$
Minimiza longitud total	*Algoritmos de aproximación*	$O(n \cdot log\ n)$

Uno de los algoritmos de coste *O(n·log n)* se basa en la definición de polígono monótono. Primero, se considera el caso de polígonos monótonos, para luego generalizarlo a polígonos cualesquiera, descomponiéndolos en una colección de polígonos monótonos disjuntos. La triangulación de polígonos monótonos tiene lugar en tiempo *O(n)* (Fournier y Montuno, 1984).

Definición: *Una cadena poligonal, C, se dice estrictamente monótona con respecto a una línea L dada, si cualquier línea ortogonal a L intersecta a C en, a lo sumo, un punto. La cadena poligonal C es monótona con respecto a L si toda línea ortogonal a L intersecta a C en un segmento único.*

Definición: *Un polígono, P, se dice monótono con respecto a una línea L, si su frontera puede descomponerse en dos cadenas monótonas con respecto a L.*

El coste para comprobar si un polígono es monótono con respecto al eje horizontal es *O(n)*.

Triangulación de polígonos monótonos:

El método que se describe a continuación es un método voraz.

Dado un polígono monótono P, el primer paso es buscar los vértices visibles para cada vértice *v* de P. Cada vez que se encuentra un vértice *u* visible desde *v*, se añade la arista que los une. Al finalizar este proceso se debe haber triangulado P (como se vio anteriormente). Este algoritmo se puede aplicar a cualquier PSLG, sin embargo, su coste es muy elevado. Por otra parte, si el polígono es monótono, el proceso anterior puede realizarse en tiempo *O(n)*:

Algoritmo para la triangulación de un polígono monótono:

***Entrada:* P, polígono monótono con n vértices**

1. **Ordenar los vértices de P según coordenada y decreciente, resultando la lista L={v(1), ..., v(n)}**
2. **Guardar v(1) y v(2) en una pila, PILA. Sea i=3**
3. **(Sea {u(1), ... u(s)} el contenido de la PILA, donde la cima es u(s)) Si v(i) es adyacente a u(1), pero no a u(s), entonces**
    1. **añadir las aristas {v(i), u(2)}, ..., {v(i), u(s)},**
    2. **sacar todos los vértices de la PILA,**
    3. **poner u(s) y v(i) en la PILA,**
    4. **i← i+1,**
    5. **ir al paso 6**
4. **Si v(i) es adyacente a u(s), pero no a u(1), entonces**
    1. **mientras el vértice bajo la cima de la PILA (sea u') sea visible desde v(i)**
        1. **añadir una arista {v(i), u'}**
        2. **sacar la cima de la PILA**
    2. **poner v(i) en la PILA**
    3. **i← i+1**
    4. **ir al paso 6**
5. **Si v(i) es adyacente a u(s) y a u(1), entonces**
    1. **añadir las aristas {v(i), u(2)}, ..., {v(i), u(s-1)}**
    2. **sacar todos los vértices de la PILA y parar**
6. **Si i<=n, volver al paso 3.**

Triangulación de un polígono simple cualquiera:

El problema ahora es encontrar una descomposición de un polígono simple en polígonos monótonos. Para esto se emplea un algoritmo que realiza un barrido del plano. Se añadirán diagonales que dividan al polígono en piezas monótonas.

La ausencia de monotonía (horizontal) tiene lugar sólo en los vértices cuyo ángulos interior es mayor que 180º y ambas aristas están a su izquierda o a su derecha. La notación común para el primer tipo de vértice es *vértice de unión* (*merge vertex*), y para el segundo *vértice de separación* (*split vertex*), debido al sentido del barrido.

Al encontrar un vértice de separación (aristas a la derecha), existe una arista e_j superior y una arista e_k inferior. Se une entonces el vértice al más cercano a la izquierda de la línea de barrido que esté entre e_j y e_k. A éste vértice se le denomina *auxiliar(e_j)*. Si no hay ningún vértice entre estas aristas, el *auxiliar(e_j)* está definido como el extremo izquierdo de una de las dos aristas que esté más cerca de la línea de barrido.

Los elementos básicos del algoritmo son:

- **Puntos de parada:** *Los extremos de los segmentos del polígono. Se ordenan por orden creciente de coordenada x.*
- **Estado de barrido:** *El estado de la línea de barrido viene dado por la lista de aristas que intersectan a la propia línea de barrido, ordenada de arriba abajo.*
- **Procesamiento de los puntos de parada:** *Hay seis tipos de puntos de parada. Sea v el punto actual en el que está detenida la línea:*
    - **Vértice de separación:** Se busca en el estado de la línea de barrido la arista e inmediatamente superior a v. Se añade una diagonal uniendo v a auxiliar(e). Se añaden las dos aristas incidentes en v al estado del barrido, y se pone v como el auxiliar de la más baja de las dos aristas, y como nuevo auxiliar de e.
    - **Vértice de unión:** Se encuentran las dos aristas incidentes a este vértice en el estado de barrido (deben ser adyacentes). Se eliminan ambas (de la lista). Sea e la aristas inmediatamente superior a ambas. Poner v como nuevo auxiliar de e.
    - **Vértice de comienzo:** (Ambas aristas están a la derecha de v, con ángulo interior menor de 180º) Se inserta este vértice y sus aristas en el estado de barrido.
    - **Vértice final:** (Ambas aristas están a la izquierda de v, con ángulo interior menor de 180º) Se eliminan ambas aristas del estado de barrido.
    - **Vértice de la cadena superior:** (Una arista está a la izquierda, y la otra a la derecha. El interior del polígono está debajo) Se cambia la arista izquierda por la arista derecha en el estado de barrido. Se pone v como el auxiliar de la nueva arista.
    - **Vértice de la cadena inferior:** (Una arista está a la izquierda, y la otra a la derecha. El interior del polígono queda encima del vértice) Se cambia la arista izquierda por la arista derecha en el estado de barrido. Sea e la arista que pasa por encima de v. Se pone v como auxiliar de e.

Esto sólo inserta arista para los vértices de separación. Para procesar los vértices de unión se aplica un algoritmo esencialmente igual que realice el barrido de derecha a izquierda. Esto podría introducir diagonales repetidas, aunque este problema puede solucionarse con cierta atención especial en los cambios de vértice auxiliar. Existen muchos casos especiales, que pueden ser tratados fácilmente, por lo que el algoritmo es muy eficiente.

Los métodos vistos hasta ahora describen algoritmos deterministas para resolver el problema de la triangulación. Sin embargo, es posible usar un método no determinista. Esto es lo que hace el algoritmo incremental no determinista de Seidel, cuya complejidad esperada es *O(n·log* n)*. En la práctica, el tiempo empleado por este algoritmo en la triangulación de un polígono simple es casi lineal. Este algoritmo consta de tres partes:

1. *Descomponer el polígono en trapezoides.*

    Sea S un conjunto de segmentos (no horizontales) del polígono que no intersectan entre sí. El algoritmo no determinista se usa para crear la descomposición trapezoidal del plano con los segmentos de S. Esto se hace tomando una ordenación cualquiera s1, ..., sN de los segmentos de S y añadiendo un segmento de cada vez para construir los trapezoides. La restricción de que no haya segmentos horizontales se impone para limitar el

número de vecinos de cada trapezoide (sin embargo, no hay pérdida de generalidad). El número de trapezoides es lineal con repecto al número de segmentos. Seidel demuestra que, si cada permutación de s1, ..., sN es igualmente probable, la construcción de los trapezoides lleva un tiempo esperado O(n·log n).*

2. *Descomponer los trapezoides en polígonos monótonos.*

   Esta operación tiene un coste lineal.
3. *Triangular los polígonos monótonos.*
   Recuérdese que esto puede hacerse en tiempo lineal.

BIBLIOGRAFÍA:

- *"Fast Polygon Triangulation based on Seidel's Algorithm"*, Atul Narkhede, Dinesh Manocha; Department of Computer Science, UNC Chapel Hill.
- *"Computational Geometry in C (Second Edition)"*, Joseph O'Rourke (capítulo 22, *Triangulations*).
- *"Computational Geometry, Methods and Applications"*, Jianer Chen; Computer Science Department, Texas A&M University.
- *"Computer Graphics (Lecture Notes)"*, (Spring 1997) Dave Mount.
- *"TEMA 6: Diagramas y triangulaciones"*, Domingo Gallardo; DCCIA, Universidad de Alicante, 1999.
- *"Sesión 6: Triangulación de polígonos"*, Domingo Gallardo; DCCIA, Universidad de Alicante, 1999.

OTRAS REFERENCIAS:

- *"A polynomial time algorithm for the minmax angle triangulation"*, H. Edelsbrunner, T.S. Tan, y R. Waupotitsch; SIAM J. Sci. Statist. Comput.
- *"Triangulating simple polygons and equivalent problems"*, A. Fournier y D.Y. Montuno; ACM Trans. on Graphics.
- *"Incremental Delaunay triangulation"*, Dani Lischinski; Academic Press, Boston.

Profesor
Alberto Márquez

Autores de esta página
Manuel Fernández Rueda
Jesús Muñoz Galiano