Modelamiento Geométrico

Nancy Hitschfeld Kahler
Departamento de Ciencias de La Computación
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile

Curso CC5501 Astroinformática
28-29 Noviembre 2011

Tabla de contenidos

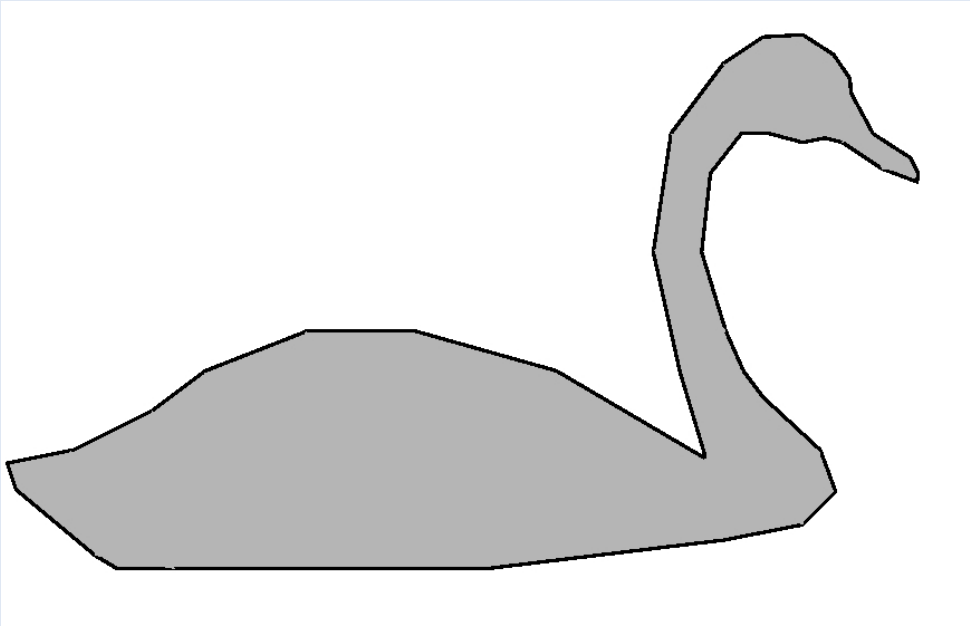
Introducción

- Geometría Computacional: rama de la computación que se dedica la estudio sistemático de algoritmos y estructuras de datos para modelar y resolver problemas geométricos
- Ejemplo de problemas geométricos:
  - Alguien camina en un campus y quiere saber cuál es la cabina telefónica más cercana. Cómo calcular computacionalmente la solución?
  - Se tiene un conjunto de puntos en el espacio. Cómo obtener una buena forma que represente geométricamente esta distribución de puntos?
  - Se desea simular el comportamiento de un objeto computacionalmente antes de fabricarlo. Qué se necesita?

Introducción

- Qué se entiende por modelamiento geométrico? Es una rama de la matemáticas aplicadas y de la geometría computacional

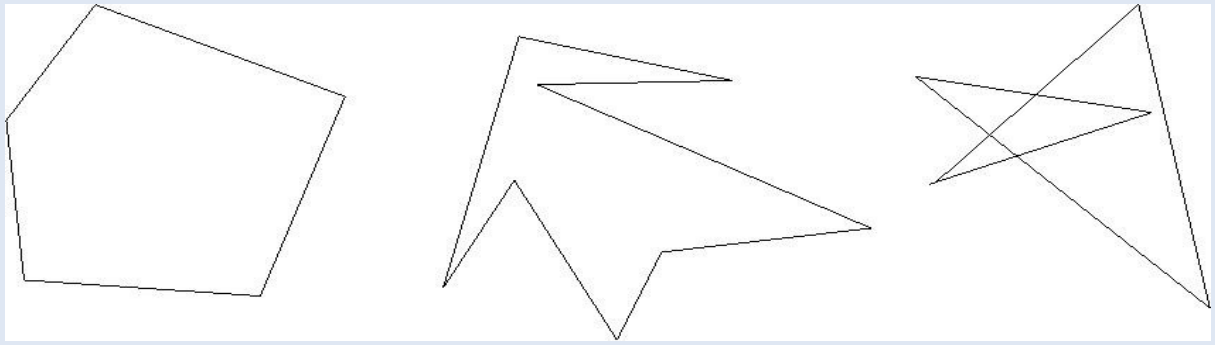
  Estudia métodos y algoritmos para la descripción matemática de formas (shapes) tanto en dos como tres dimensiones



Introducción

Polígono: región delimitada por segmentos que forman una curva simple cerrada.

- Sean $v_0, v_1, v_2, v_3, \ldots, v_{n-1}$ n puntos (vértices) en el plano. Sean $e_0 = v_0v_1$, $e_1 = v_1v_2, \ldots, e_{n-1} = v_{n-1}v_0$ los segmentos(arcos) que conectan los puntos. Se dice que estos segmentos rodean un polígono ssi:
- La intersección de cada par de segmentos es solo un punto: $e_i \cap e_{i+1} = v_{i+1}$



•Representación natural de un póligono: secuencia de puntos ordenados clock-wise(cw) o counter clock wise (ccw)

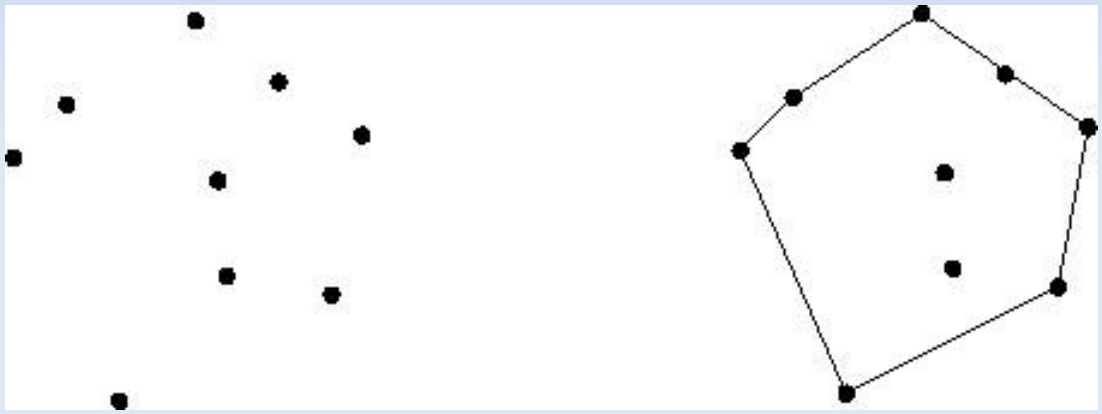
•Cómo verificar si un polígono está orientado clock-wise?

Introducción

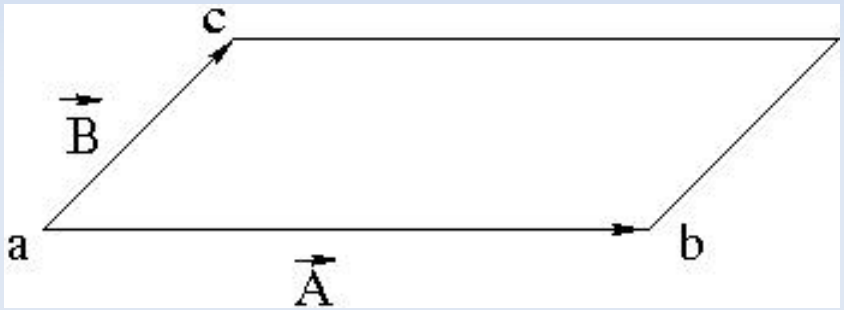
- Cómo resolver correctamente un problema geométrico?
  - comprender las propiedades del problema geométrico: si uno no entiende la geometría del problema, ningún algoritmo ayudará
  - aplicación apropiada de técnicas algorítmicas y estructuras de datos: si uno conoce bien el problema geométrico, pero no conoce las técnicas algorítmicas correctas, tampoco sirve
- Problema concreto: Cálculo de la cerradura convexa de un conjunto de puntos P
  - Input: conjunto de puntos P
  - Output: polígono convexo más pequeño que rodea el conjunto de puntos P

Introducción

- Ejemplo de la cerradura convexa de un conjunto de puntos



Introducción

- Operaciones básicas:
  - int area(Point a, Point b, Point c) (a, b, c definen un triangulo ccw)
    - $2*area(a,b,c) = a_0b_1-a_1b_0+a_1c_0-a_0c_1+b_0c_1-b_1c_0$
    - 

  - boolean left(Point a, Point b, Point c){ return area2(a,b,c)>0; }
  - Boolean collineal(Point a, Point b, Point c){ return area2(a,b,c) == 0; }
- Respuesta exacta si coordenadas son enteras
- Intersección de segmentos? También respuesta exacta con enteros

Algoritmos para calcular la Cerradura convexa en 2D (I)

- Primeras Ideas:
  - Todos los pares de puntos (p,q) que forman parte de la cerradura no tienen ningún punto a la izquierda (o derecha) dependendiendo de donde se mire. Este algoritmo es $O(n^3)$
  - Calcular la secuencia de vértices en orden: partir de cualquier arco eligiendo uno de sus puntos extremos, buscar el siguiente arco que lo contenga, y continuar así hasta llegar al otro extremo del arco inicial.
  - Qué pasa cuando un punto esta sobre la línea que pasa por (pq)? A esto se le llama caso degenerado. El algoritmo mejorado para que incluya el caso anterior será entonces:
    - pq pertenece a CH(P) ssi todos los demás puntos r están a la derecha o sobre pq.
  - Qué otra cosa puede influir en la correctitud del algoritmo? Podemos testear de manera exacta si un punto esta a la derecha o izquierda de una línea?
    esto no es necesariamente verdadero: si los puntos estan dados en coordenadas de punto flotante y los cálculos son hechos usando aritmética de punto flotante, existirán errores de redondeo que puede distorsionar el resultado calculado.
    - Referencia: http://docs.sun.com/source/806-3568/ncg_goldberg.html

Algoritmos para calcular la Cerradura convexa en 2D (I)

- Imaginen que hay tres puntos p,q,r que son casi colineales y que todos los otros puntos estan lejos a la derecha de ellos. El algoritmo chequea pares (p,q), (r,q) y (p,r). Es posible que debido a los errores de redondeo tengamos la siguientes respuestas:

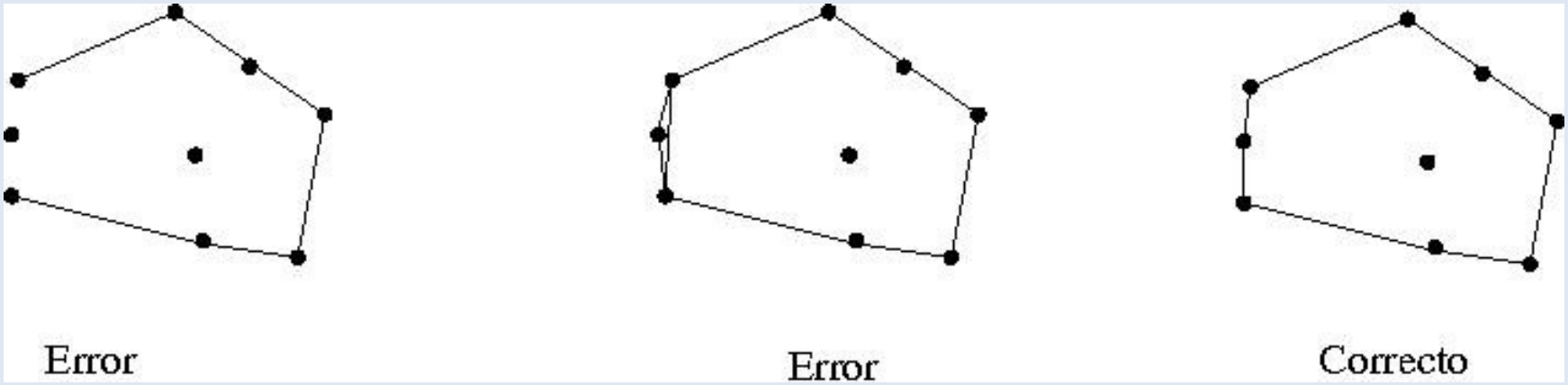
  • r está a la derecha de (p,q), p está a la derecha de (r,q) y q está a la derecha de (p,r). Esto es geométricamente imposible, pero el algoritmo puede responder esto y aceptar los tres arcos como parte de la cerradura.
  • Peor aún. Los tres casos podrian dar la respuesta contraria y rechazar los tres arcos y el polígono resultante no sera cerrado.

  De los casos anteriores podemos tener:

  · puntos extremos que aparecen mas de dos veces
  · puntos extremos que aparecen una sola vez.

Algoritmos para calcular la Cerradura convexa en 2D (I)

- Que conclusión podemos sacar?
  - El algoritmo anterior funciona, es lento, incluye casos degenerados(maneja casos especiales) pero no es robusto:
  - Pequeños errores de cálculo pueden hacerlo fallar en situaciones inesperadas.
  - Cuál es el problema? Nosotros hemos asumido que podemos calcular de manera exacta con números reales.



Por qué se producen los errores de precisión

- Representación de números reales
  - $\pm$ d.dd... d $\times \beta^{e}$
  - $\pm d_0 . d_1 d_2 \ldots d_{p-1} \times \beta^{e}$ (p número dígitos mantisa, β base)
  - Representa al número: $+- (d_0+d_1\beta^{-1}+d_2\beta^{-2}+d_{p-1}\beta^{(p-1)})\beta^{e}$ $(0<=d_i<\beta)$
  - Podemos representar β^{p} mantisas, con $e_{max}-e_{min}+1$ exponentes
- Ejemplo: $\beta=2$, $p=3$, $e_{max} = 2$, $e_{min} = -1$ => 100 con cada exponente
  101
  110
  111
- Total: 16 números distintos
- Los números representados en base 10 son:
- Exponente 2 => 4, 5, 6, 7 Exponente 1 => 2, 2.5, 3, 3.5
- Exponente 0=> 1, 1.25, 1.5, 1.75 Exponente -1 => 0.5, 0.625, 0.75, 0.825
  - Qué pasa con la distribución de los puntos en la recta numérica?

Algoritmos para calcular la Cerradura convexa en 2D (I)

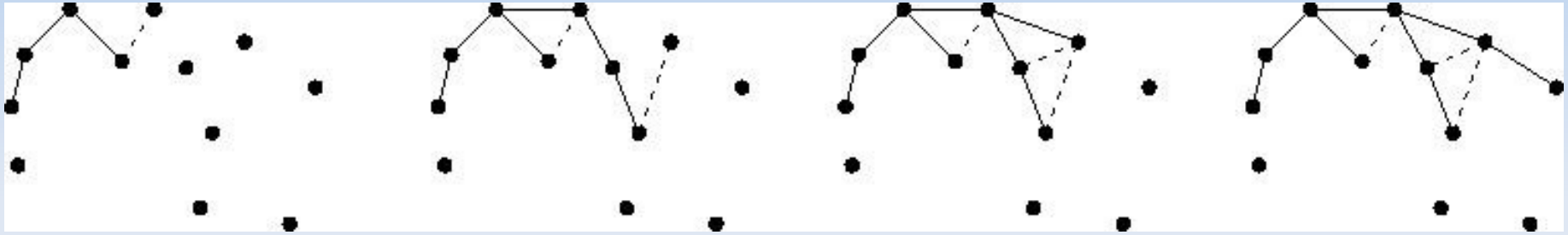
- Podemos encontrar algún algoritmo mejor? Qué buenas técnicas algorítmicas existen?
  - dividir para reinar
  - incremental
  - line-sweeping
  - ....
  - http://www.cse.unsw.edu.au/~lambert/java/3d/hull.html

Algoritmos para calcular la Cerradura convexa en 2D (II)

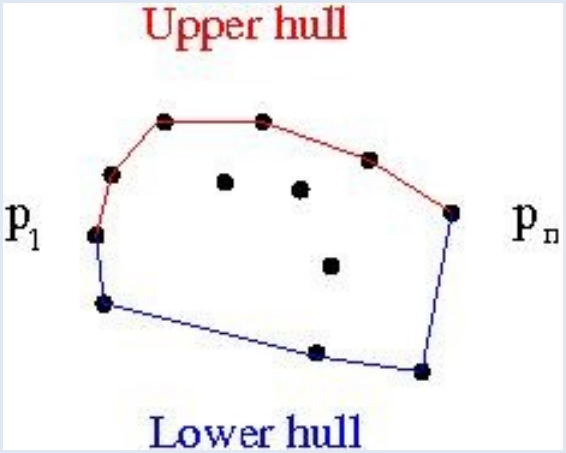
- Usando una estrategia incremental
  - Puntos son insertados uno a uno, actualizando la solución despues de cada adición
  - Inserción de izquierda a derecha
- Pasos del algoritmo:
  - Ordenar los puntos por la coordenada x O(n log n)
  - Calcular la parte superior del hull (upper hull): parte de la cerradura convexa que va desde p_1 (el punto de más a la izquierda) hasta p_n (punto de más a la derecha) en orden clockwise
  - Calcular la parte inferior de la cerradura convexa (lower hull) partiendo de p_n hasta p_1.

Algoritmos para calcular la Cerradura convexa en 2D (II)

Pasos del algoritmo para construir el upper hull:



Resultado:



Algoritmos para calcular la Cerradura convexa en 2D (II)

- Algoritmo: p_1 a p_{i-1} pertenecen a la cerradura convexa superior. Al agregar p_i puede pasar lo siguiente:
  - p_i puede hacer doblar a la izquierda => sacar p_{i-1} e intentar con p_{i-3}, p_{i-2} y p_i
  - p_i puede hacer doblar a la derecha => agregar p_i. Esto es lo que se debe cumplir para ir generando un poligono convexo.
- Qué problemas hemos ignorado en la solución anterior que hacen que este algoritmo no sea correcto?
  - Asumimos que no hay dos puntos con la misma coordenada x: esto se puede solucionar ordenando primero por la coordenada x, y si dos puntos tienen la misma coordenada x, ordenamos por la coordenada y.
  - Ignoramos que los tres puntos por los que decidimos que se dobla a la izquierda o a la derecha pueden ser colineales. El punto del medio no debería estar en la cerradura asi es que lo debemos tratar como si doblaramos a la izquierda: el test debe retornar verdadero si dobla a la derecha, y falso de lo contrario.

Algoritmos para calcular la Cerradura convexa en 2D (II)

- Qué hace el algoritmo en presencia de errores en la aritmética de punto flotante?
  - un punto dentro de la cerradura real no es eliminado
  - un punto que debería estar en la cerradura es eliminado
- Pero la integridad de la estructura del algoritmo no es dañada. El algoritmo calcula una cadena poligonal cerrada.
  - La lista de puntos es una lista Clockwise (CW) en que cada tres puntos doblan a la derecha o casi doblan a la derecha
  - El único problema que aun puede ocurrir es que cuando tres puntos están muy cercanos, una vuelta que es realmente una virada a la izquierda puede ser interpretada como un virada a la derecha. Es puede generar un dent (hendidura) en el polígono.
  - Solución? Si dos puntos están muy cercanos, juntarlos y considerar uno solo

Algoritmos para calcular la Cerradura convexa en 2D (II)

- Es ésta una buena solución? Si
  - No podemos esperar tener un resultado exacto con aritmética inexacta
  - En muchas aplicaciones esto puede ser suficiente
- Importante: Hay que ser cuidadoso en la implementación de los tests básicos para evitar la mayor cantidad de errores posible
- De qué orden es el algoritmo? O(n log n)
- Software a mirar: qhull (http://www.qhull.org/)