Razonamiento Geométrico
Tema 6: Diagrama de Voronoi

Sesión 13: Triangulación de Delaunay

Copyright © 2002-2003 Universidad de Alicante

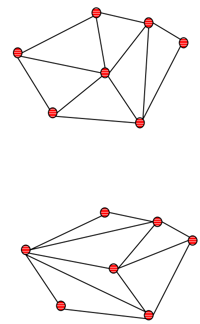
Índice

- **Triangulación de Delaunay**
- **Relación entre TD y DV**
- **Relación entre TD y recubrimiento convexo 3D**
- **Algoritmo para el cálculo de la TD**

Copyright © 2002-2003 Universidad de Alicante

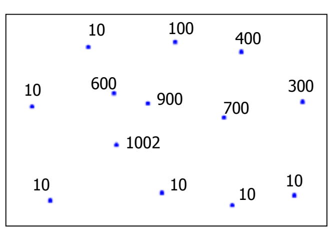
Triangulación

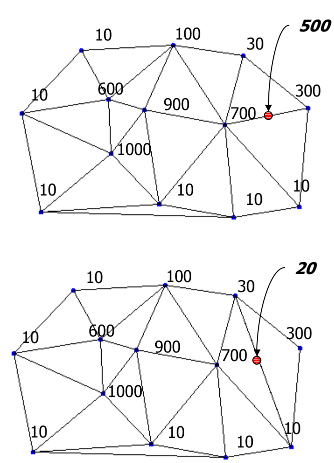
Dado un conjunto de puntos
en el plano S, una triangulación
de S se define como una región
planar cuyos vértices son
los puntos de S y cuyas caras
son triángulos



 Copyright © 2002-2003 Universidad de Alicante

Interpolación de mapas de altura



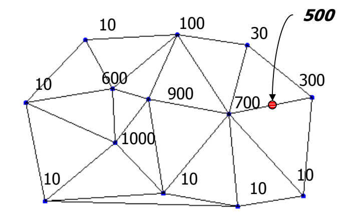


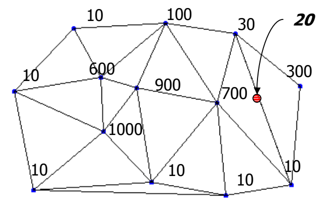
 Copyright © 2002-2003 Universidad de Alicante

Triangulación “óptima”

Una triangulación T_1 es mejor que T_2 cuando el menor ángulo de los triángulos de T_1 es mayor que el menor ángulo de los triángulos de T_2.

La mejor triangulación de un conjunto de puntos es la que maximiza el ángulo mínimo de los triángulos

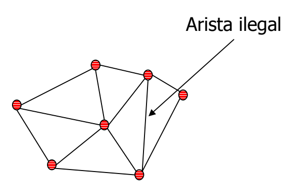


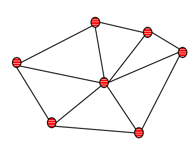


Copyright © 2002-2003 Universidad de Alicante

Aristas ilegales (1)

Una arista de una triangulación es ilegal si al “girarla” aumenta el ángulo mínimo de los triángulos adyacentes

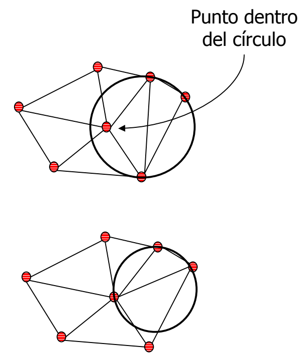




Copyright © 2002-2003 Universidad de Alicante

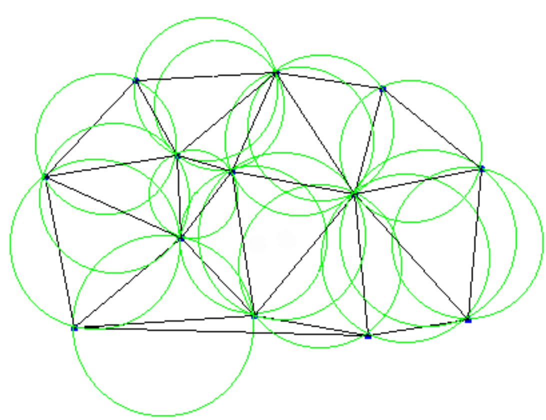
Aristas ilegales

La circunferencia definida por los vértices de un triángulo contienen otro punto de la triangulación si y sólo el triángulo tiene una arista es ilegal.



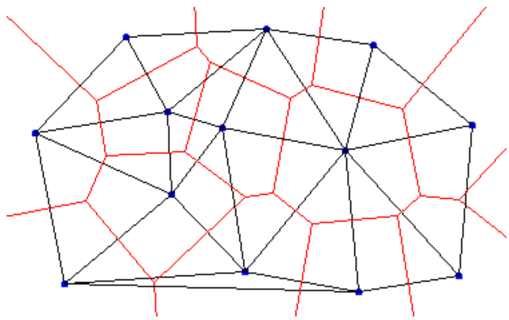
Copyright © 2002-2003 Universidad de Alicante

Triangulación óptima



Copyright © 2002-2003 Universidad de Alicante

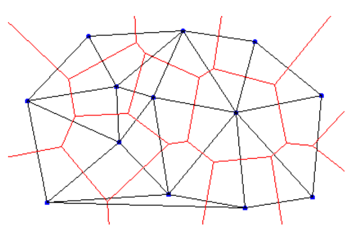
Triang. de Delaunay dual de DV



 Copyright © 2002-2003 Universidad de Alicante

DV a partir de la TD

1. Para cada triángulo de la triangulación
2. Vértice del DV = centro del círculo circunscrito por el triangulo
3. Aristas del DV = perpendiculares a los lados del triangulo



 Copyright © 2002-2003 Universidad de Alicante

Centro del círculo circunscrito

- **Dados tres puntos a, b, c, el centro p del círculo circunscrito por ellos se calcula como**

$$p_x = (b_y a_x^2 - c_y a_x^2 - b_y^2 a_y + c_y^2 a_y + b_x^2 c_y + a_y^2 b_y + \\ + c_x^2 a_y - c_y^2 b_y - c_x^2 b_y - b_x^2 a_y + b_y^2 c_y - a_y^2 c_y) / D$$

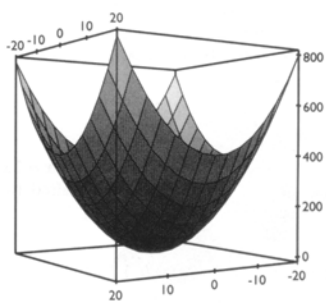
$$p_y = (a_x^2 c_x + a_y^2 c_x + b_x^2 a_x - b_x^2 c_x + b_y^2 a_x - b_y^2 c_x - \\ - a_x^2 b_x - a_y^2 b_x - c_x^2 a_x + c_x^2 b_x - c_y^2 a_x + c_y^2 b_x) / D$$

siendo

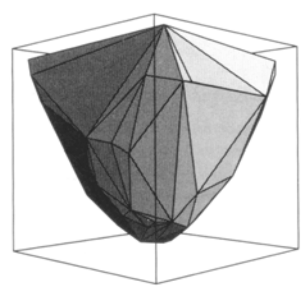
$$D = 2 (a_y c_x + b_y a_x - b_y c_x - a_y b_x - c_y a_x + c_y b_x)$$
$$radio^2 = (a_x - p_x)^2 + (a_y - p_y)^2$$

Copyright © 2002-2003 Universidad de Alicante

Relación TD y RC 3D (1)



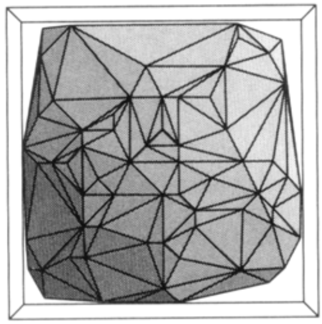
$z = x_i^2+y_i^2$



1. Para todo punto de S
2. $(x_i,y_i) \rightarrow (x_i, y_i, x_i^2+y_i^2)$
3. Cálculo del RC de los puntos 3d resultantes

Copyright © 2002-2003 Universidad de Alicante

Relación TD y RC 3D (2)



La vista infrerior del RC 3D
es la TD del conjunto de puntos

Copyright © 2002-2003 Universidad de Alicante

Código fuente (O'Rourke)

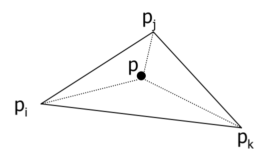
```
main()
{
        int x[NMAX],y[NMAX],z[NMAX]; /* input points xy,z = x^2 + y^2 */
        int n;                       /* number of input points */
        int i, j, k, m;              /* indices of four points */
        int xn, yn, zn;              /* outward normal to (i,j,k) */
        int flag;                    /* t if m above of (i,j,k) */

        /* Input points and compute z = x^2 + y^2. */
        scanf("%d", &n);
        for ( i=0; i<n; i++ ) {
                scanf("%d %d", &x[i], &y[i]);
                z[i]=x[i] * x[i]+y[i] * y[i];
        }

        /* For each triple (i,j,k) */
        for ( i=0; i < n-2; i++ )
        for ( j=i+1; j < n; j++ )
        for ( k=i+1; k < n; k++ )
        if ( j != k ) {
           /* Compute normal to triangle (i,j,k). */
           xn=(y[j]-y[i])*(z[k]-z[i])-(y[k]-y[i])*(z[j]-z[i]);
           yn=(x[k]-x[i])*(z[j]-z[i])-(x[j]-x[i])*(z[k]-z[i]);
           zn=(x[j]-x[i])*(y[k]-y[i])-(x[k]-x[i])*(y[j]-y[i]);

           /* Only examine faces on bottom of paraboloid: zn < 0. */
           if ( flag=(zn < 0) )
                   /* For each other point m */
                   for (m=0; m<n; m++)
                           /* Check if m above (i,j,k). */
                           flag=flag &&
                                   ((x[m]-x[i])*xn +
                                    (y[m]-y[i])*yn +
                                    (z[m]-z[i])*zn <= 0);
           if (flag)
                   printf("%d\t%d\t%d\n", i, j, k);
        }
}
```

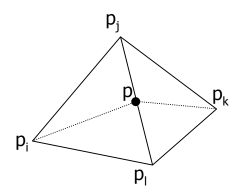

Copyright © 2002-2003 Universidad de Alicante

Algoritmo incremental

TriangulaciónIncremental(TD, p, p_{-1},p_{-2},p_{-3})
// p_{-1}, p_{-2}, p_{-3} forman un triángulo que contiene a TD

1. Encontrar el triángulo $p_ip_jp_k$ de TD en el que se encuentra p
2. Si p se encuentrar en el interior de $p_ip_jp_k$
3. Añadir aristas de p a p_i, p_j, p_k
4. Legalizar aristas pipj, pj,pk, pkpi
5. Sino (p se encuentra en una arista de $p_ip_jp_k$, digamos p_ip_j)
6. Añadir aristas de p a p_i, p_j, p_k, p_l (p_l es el punto del otro triángulo incidente con $p_ip_jp_k$)
7. Legalizar aristas p_ip_l, p_lp_j, p_jp_k, p_kp_i



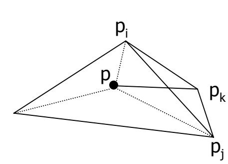


 Copyright © 2002-2003 Universidad de Alicante

Legalizar arista

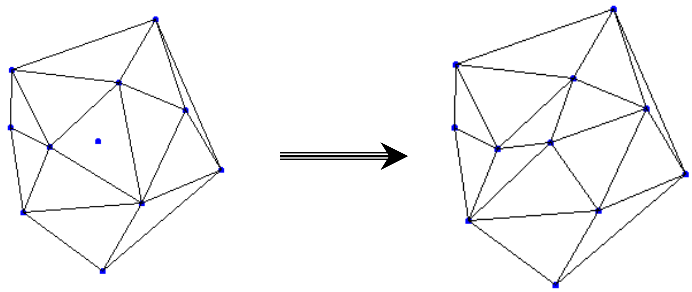
Legalizar la arista p_ip_j al introducir p

1. Si p_ip_j es ilegal
2. Sea $p_ip_jp_k$ el triangulo adyacente a pp_ip_j
3. Reemplazar p_ip_j con p_rp_k
3. Legalizar aristas p_ip_k y p_kp_j



 Copyright © 2002-2003 Universidad de Alicante

Ejemplo



 Copyright © 2002-2003 Universidad de Alicante