# Fortune 算法的一种实现

**jayven**

**介绍**

Fortune 扫描线算法能在 $\Omega(n\log n)$ 的时间复杂度下为平面点集计算出 *voronoi* 图/*delaunay* 三角剖分，这里介绍一种实现方法。

**概述**

*Voronoi* 图可计算出一个有限平面点集中每一个点的领域，所谓一个点 *P* 的领域 *E(P)*，即任意点 $p \in E(P)$，都满足 *distance(P, p) > distance(Q, p)*，其中 *Q* 为此平面点集的任意其他点。而有限平面点集的 *delaunay* 三角剖分为此点集的 *voronoi* 图的对偶图。

**基础算法参看**

*Computational Geometry Algorithms and Applications 计算几何-算法于应用（第二版）*

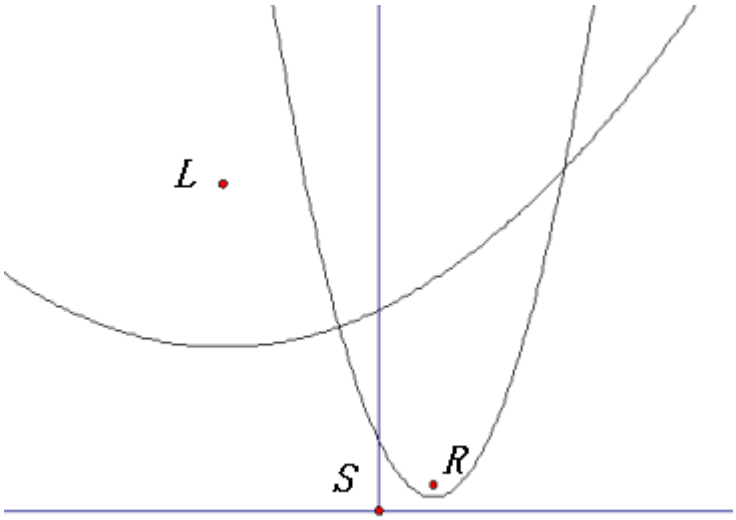
**实现要点**

1、算法使用平衡二叉树（BST）作为海浪线的储存结构，这里首先使用一个链表储存所有的海浪线，并使用 treap（见 http://en.wikipedia.org/wiki/Treap）用以加速查找。

2、程序使用一组排序过了的基点数组（见 *dt_run_vertexes*）以及一个可能会变长的堆（见 *ce_pq_**），前者是将要按顺序发生的基点事件（弧出现），后者储存的是在计算过程中出现的圆事件（弧消失）。

3、按基点事件（见 *handle_site_event*）跟圆事件（见 *handle_circle_event*）的发生顺序进行处理。

4、若是基点事件，则查找其发生所在的海浪线（判断的函数为 *after_break_point*），如图：



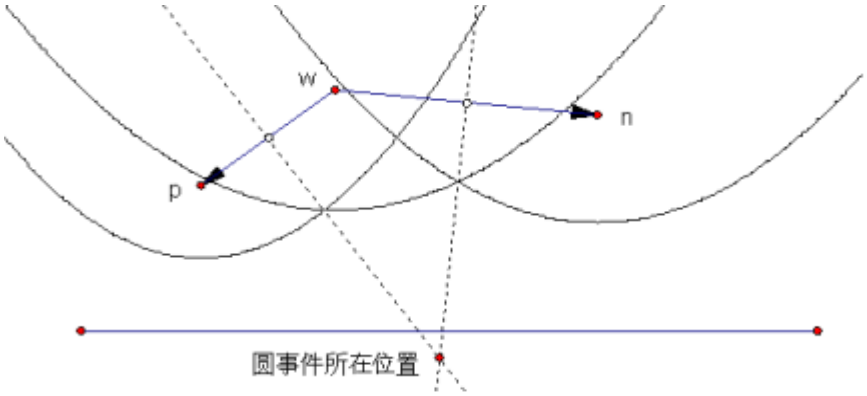

令扫描线为 $y = Y$，则抛物线 $P(L)$ 的方程是 $(x - L.x)^2 + (y - L.y)^2 = (y - Y)^2$

同理，抛物线 $P(R)$ 的方程是 $(x - R.x)^2 + (y - R.y)^2 = (y - Y)^2$，直线方程为 $x = S.x$，代入两抛物线的方程可得各自交点的 y 值，两者 y 坐标之差为：

$$Delta = (L.y - R.y) - \frac{(S.x - L.x)^2}{S.y - L.y} + \frac{(S.x - R.x)^2}{S.y - R.y}$$

从图中可看出从 $Delta$ 正负值和 $L.y - R.y$ 的正负值即可判断出 S 处于断点的左边还是右边，上述公式需要 4 此乘/除运算。

找到海浪线正确的位置后即插入到海浪线上，新增了海浪线后，有可能新增两个未来的圆事件。

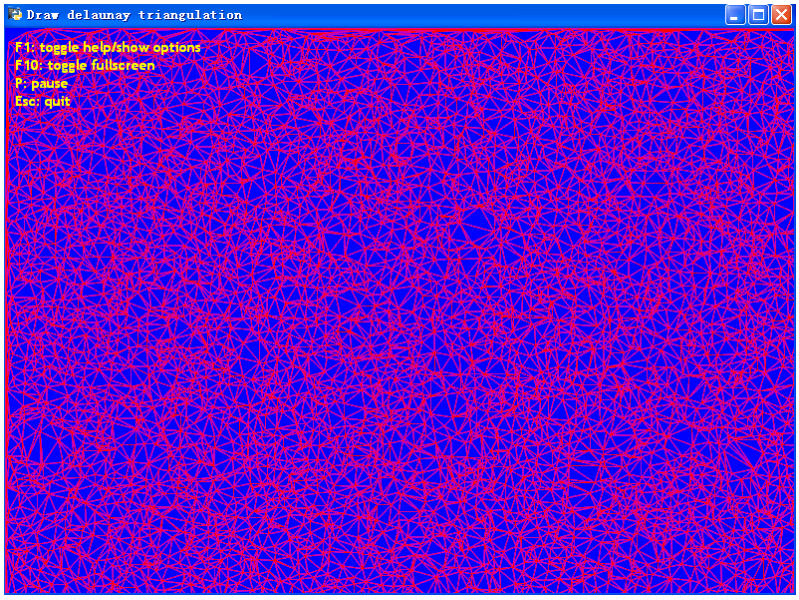



海浪线上的一段弧，加上它前面（如果有的话）和后面（如果有的话）的连续三段弧，令这三段弧的基点分别是 *w/p/n*，则若从向量 *w->p* 到 *w->n* 的角度小于 180 度的话，那么，将来是有可能有圆事件的，如图（因为其中垂线，即三个基点的领域边界会相交）
这个可以用行列式判断得到（见程序），若有候选圆事件的话，需要算出这三点的外接圆的最低端坐标，即候选圆事件坐标。

**参考**

http://www.cs.cmu.edu/~quake/triangle.html

**SceenShot**


Draw delaunay triangulation
F1: toggle help/show options
F10: toggle fullscreen
P: pause
Esc: quit