维诺图的算法实现

# 1.Voronoi图的定义

任意两点p 和q 之间的欧氏距离，记作 dist(p, q) 。就平面情况而言，我们有dist(p, q) =  (px-qx)2+ (py-qy)2

设P := {p1, …, pn}为平面上任意 n 个互异的点；这些点也就是基点。按照我们的定义，所谓P对应的Voronoi图，就是平面的一个子区域划分——整个平面因此被划分为n 个单元（cell ），它们具有这样的性质：任一点q位于点pi 所对应的单元中，当且仅当对于任何的pj∈Pj, j≠i, 都有dist(q, pi)<dist(q, pj)。我们将与P对应的Voronoi图 记作Vor(P)。

“Vor(P) ”或者“Voronoi图”所指示的仅仅只是组成该子区域划分的边和顶点。在Vor(P)中，与基点pi 相对应的单元记作V (pi)——称作与pi 相对应的Voronoi单元（Voronoi cell）。



任给平面上两点p 和q ，所谓 p 和q 的平分线（bisector），就是线段 pq的垂直平分线。该平分线将平面划分为两张半平面（half-plane）。点 p 所在的那张开半平面记作 h(p, q) ，点 q 所在的那张开半平面记作 h(q, p) 。请注意，r  ∈  h(p, q) 当且仅当 dist(r, p) < dist(r, q) 。据此，可以得出如下观察结论：

     V (pi)  =   ∩ h(pi, pj) , 1≤j≤n, j≠ i

也就是说，V (pi)是(n-1)张半平面的公共交集；它也是一个（不见得有界的）开的凸多边形（convex polygon）子区域.

很显然，Voronoi顶点到相邻的三个site距离相等；Voronoi边上任意一点到相邻的两个site距离相等；

对于任何点q，我们将以q为中心、内部不含P中任何基点的最大圆，称作q关于P的最大空圆（largest empty circle ），记作Cp(q)。以下定理指出了Voronoi图的顶点及边所具有的特征：

对于任一点集P 所对应的Voronoi图Vor(P) ，下列命题成立：    
1) 点q 是Vor(P) 的一个顶点，当且仅当在其最大空圆Cp(q)的边界上，至少有 三个基点； (Voronoi顶点是三个site的外接圆的圆心)   
2) pi 和pj 之间的平分线确定了Vor (P) 的一条边，当且仅当在这条线上存在一 个点 q，Cp(q)的边界经过pi 和pj，但不经过其它站点。

# **2.构造Voronoi图**

我使用的是**Delauney三角网剖分算法。该算法分为三步：**

**一、凸包生成：1）**求出如下四点：min(x-y)、min(x+y)、max(x-y)、max(x+y)并顺次放入一个数组，组成初始凸包；2）对于凸包上的点I，设它的后续点为J，计算矢量线段IJ右侧的所有点到IJ的距离，求出距离最大的点K；3）将K插入I,J之间，并将K赋给J；4）重复2，3步，直到点集中没有在IJ右侧的点为止；5）将J赋给I，J取其后续点，重复2，3，4步，当遍历了一次凸包后，凸包生成完成。

**二、环切边界法凸包三角剖分：**在凸包数组中，每次寻找一个由相邻两条凸包边组成的三角形，在该三角形的内部和边界上都不包含凸包上的任何其他点，然后去掉该点得到新的凸包链表，重复这个过程，最终对凸包数组中的点进行三角剖分成功。

**三、离散的内插：1）**建立三角形的外接圆，找出外接圆包含待插入点的所有三角形，构成插入区域；2）删除插入区域内的三角形公共边，形成由影响三角形顶点构成的多边形；3）将插入点与多边形所有顶点相连，构成新的Delaunay三角形；4）重复1，2，3，直到所有非凸包上的离散点都插入完为止**。**

# 3.具体的数据结构和部分源代码

## 3.1基本的数据结构

点：struct Site

{

int m\_nid;

Vector1 m\_pos;

};

边：struct Edge

{

int m\_nid;

float a, b, c;

Vector1 m\_initPos;

Site\* m\_pEnd[2];

Site\* m\_pReg[2];

CCircleEvent\* m\_pLEvent; // 在边的左边发生的圆事件

CCircleEvent\* m\_pREvent; //在边的右边发生的圆事件

};

垂直平分线：struct HalfEdge

{

HalfEdge(Edge\* pEdge, Site\* pSite)

{

m\_pEdge = pEdge;

m\_pLeft = NULL;

m\_pRight = NULL;

m\_pNext = NULL;

m\_pSite = pSite;

m\_pEvent = NULL;

};

Edge\* m\_pEdge; // this pointer points edge on the right side of halfedge defined here

HalfEdge\* m\_pLeft;

HalfEdge\* m\_pRight;

HalfEdge\* m\_pNext;

Site\* m\_pSite;

CCircleEvent\* m\_pEvent; // pointer to circle for which this halfedge is center edge.

};

存储边的链表：CEdgeList::CEdgeList(CVoronoi\* pVoronoi)

{

m\_nEdgeCount = 0;

m\_pLeftEnd = NULL;

m\_pRightEnd = NULL;

m\_pVoronoi = pVoronoi;

}

## 3.2三类事件：CEvent()、CSiteEvent()、CCircleEvent()

1. CEvent()

{

m\_pNext = NULL;

}

1. CSiteEvent()

{

m\_pSite = NULL;

}

1. CCircleEvent()

{

m\_pHEdge[0] = NULL;

m\_pHEdge[1] = NULL;

m\_pHEdge[2] = NULL;

}

CCircleEvent(const Vector& center, float yvalue)

{

m\_ptCenter = center;

m\_yvalue = yvalue;

m\_pHEdge[0] = NULL;

m\_pHEdge[1] = NULL;

m\_pHEdge[2] = NULL;

}

1. void CCircleEvent::HandleEvent(CEdgeList\* pEdgeList, CVoronoi\* pVoronoi)

{

for (int i = 0; i < 2; i++)

{

Edge\* pEdge = m\_pHEdge[i]->m\_pEdge;

if (pEdge->m\_pREvent == this) pEdge->m\_pREvent = NULL;

else if (pEdge->m\_pLEvent == this) pEdge->m\_pLEvent = NULL;

if (pEdge->m\_pEnd[0] == NULL) pEdge->m\_pEnd[0] = pSite;

else pEdge->m\_pEnd[1] = pSite;

}

m\_pHEdge[1]->m\_pEvent = NULL;

Edge\* newEdge = pVoronoi->CreateNewEdge(m\_ptCenter);

newEdge->m\_pEnd[0] = pSite;

newEdge->m\_pReg[0] = m\_pHEdge[0]->m\_pSite;

newEdge->m\_pReg[1] = m\_pHEdge[2]->m\_pSite;

Vector vec, vec1, vec2;

vec1 = m\_pHEdge[0]->m\_pSite->m\_pos;

vec2 = m\_pHEdge[2]->m\_pSite->m\_pos;

newEdge->a = vec2.x - vec1.x;

newEdge->b = vec2.y - vec1.y;

newEdge->c = 0.5f\*(vec2.x\*vec2.x - vec1.x\*vec1.x + vec2.y\*vec2.y - vec1.y\*vec1.y);

vec1 = m\_pHEdge[0]->m\_pEdge->m\_initPos - m\_ptCenter;

vec2 = m\_pHEdge[1]->m\_pEdge->m\_initPos - m\_ptCenter;

vec1.Normalize();

vec2.Normalize();

vec = vec1 + vec2;

vec.Normalize();

Site\* site = m\_pHEdge[1]->m\_pSite;

m\_pHEdge[0]->m\_pEdge = newEdge;

pEdgeList->Delete(m\_pHEdge[1]);

CCircleEvent\* pEvent;

if ((pEvent = pEdgeList->CircleEventOnLeftOfCircleEvent(newEdge, m\_pHEdge[2], m\_yvalue, vec, site)) != NULL)

{

pEvent->m\_ptCenter.x,pEvent->m\_ptCenter.y, pEvent->m\_pHEdge[0], pEvent->m\_pHEdge[1], pEvent->m\_pHEdge[2]);

pVoronoi->AddNewEvent(pEvent, TRUE);

}

if ((pEvent = pEdgeList->CircleEventOnRightOfCircleEvent(newEdge, m\_pHEdge[0], m\_yvalue, vec, site)) != NULL)

{

pEvent->m\_ptCenter.x,pEvent->m\_ptCenter.y, pEvent->m\_pHEdge[0], pEvent->m\_pHEdge[1], pEvent->m\_pHEdge[2]);

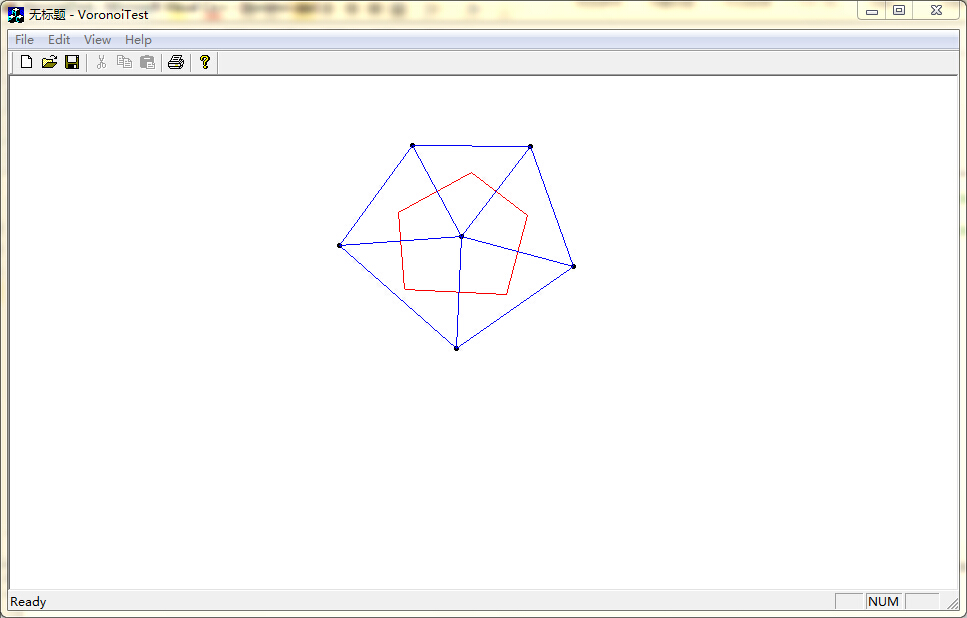
pVoronoi->AddNewEvent(pEvent, TRUE);

}

}

## 3.3实验结果

插入6个点后的Voronoi图



接着再插入两个点，图像为

![C:\Users\Jian\AppData\Roaming\Tencent\Users\603075536\QQ\WinTemp\RichOle\JRMTP${4BPMM7P4FF](7B2T.jpg](data:image/jpeg;base64,)