计算几何大实验报告

---障碍 voronoi 图的近似生成

华北所　　p0816024 王凡 p0816020 杨玉来

1. 问题背景

Voronoi 图是以两点间线段的长度作为距离,对平面的一种分割。然而,在地理环境中,存在着许多自然的或人为的障碍,使得从一点到另一点不能直线到达。为了扩大 Voronoi 图的应用领域,本文对传统的 Voronoi 图进行扩充,给出了障碍 Voronoi 图的定义、性质,以及离散生成线段障碍 Voronoi 图的方法。

马路本来是让人们通行的,但在现实中,马路中央往往被加上护栏,使人们只能沿马路方向通过,不能跨越护栏,人们若要穿越马路,需到十字路口处。这样,加上护栏的马路段可被看作障碍。现考察一个城市的某个地区,将其所含的各所学校看作基点,加上护栏的马路段看作障碍,对此地区进行规划,使得学生入学选择的学校遵循就近原则。下图近似的障碍 voronoi 相关图：

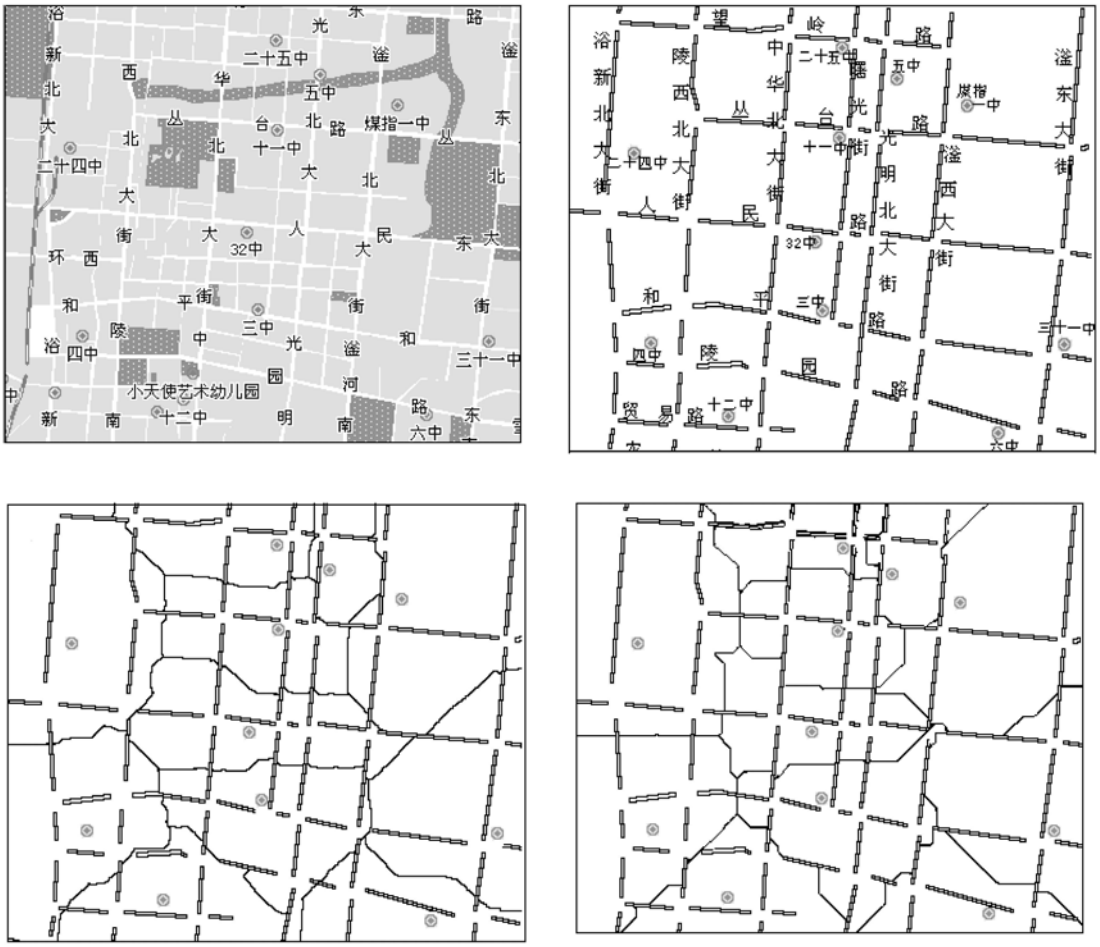


图 1

1. 相关知识

先定义障碍距离,然后给出障碍Voronoi 图的定义。

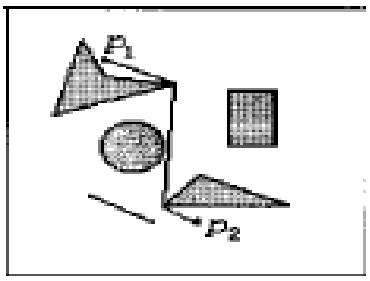


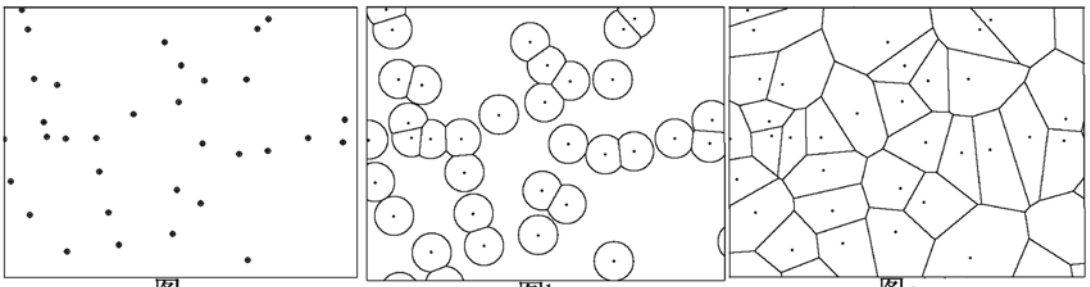
图 2

定义 设Gi (i = 1 ,2 , …, m) 为平面上m 个互不相交的几何图形, p1 、p2 为平面上的两个点。将从其中任一点出发,至多仅与Gi 的边界相交,而到达另一点的最短Euclid 距离,称为p1 、p2 间的穿过障碍Gi 的Euclid 距离, 简称为p1 、p2 间的障碍距离(图2) ,记做D(p1 , p2) 。求平面上的障碍距离已有成熟的算法,在许多有关计算几何或算法的专著中都有述及

2. 算法原理

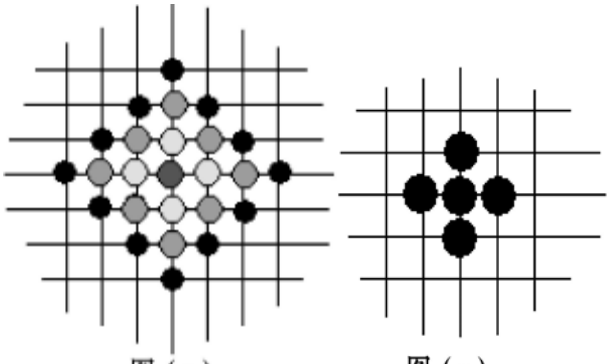
利用用 voronoi 边到相邻母点距离相等的特性，将 voronoi 边绘制出来，从而实现整个算法。

先将屏幕置为白色,并对每个基点、指定不同的颜色（非黑色），使不同的基点颜色互不相同，将障碍颜色置为黑色。然后对每个基点,从其周围开始,以基点为圆心,以指定的颜色(即基点的颜色) 和相同的速度向外扩展逐点画圆。如下图：



图a 图b 图c

为了方便实现，采用近似的菱形代替圆



图（a）　　图（a）

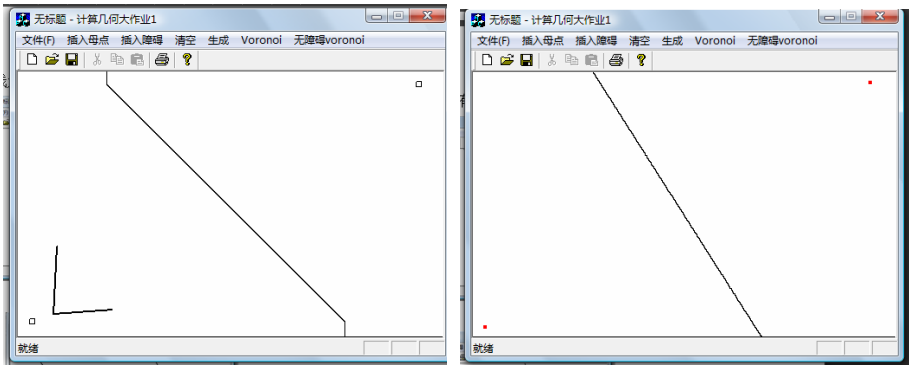
每个基点的菱形都以相同速度向外扩张，逐点画菱形时,对于当前点,先检测周围 4 点的颜色，是白色则将其涂成基点颜色并将其作为下次画菱形的基点，否则就跳过检测下一点。这样遇到障碍不能通过，只能从端点绕过。等颜色涂满后，不同颜色的区域代表不同基点的 voronoi 区域。

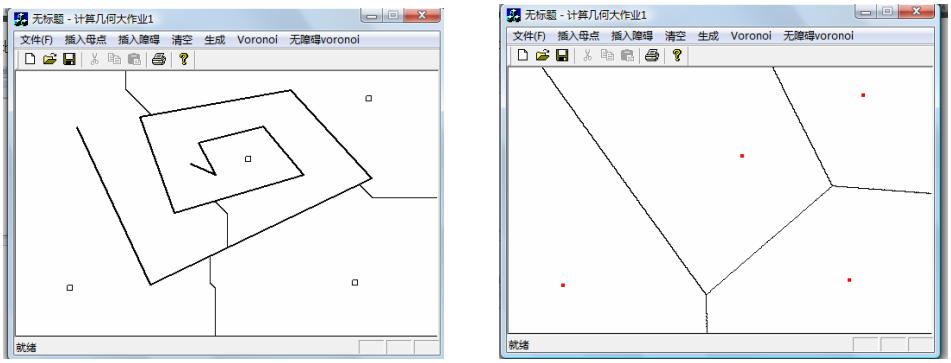
纵向及横向扫描全屏幕,若某一像素点与其后继像素点颜色不同,将其置为黑色，横向(或纵向)扫描全屏幕,将黑色像素点保留原色,其它像素点置为白色，最后得到图为障碍voronoi图。

具体见参考文献

3. 运行结果比较

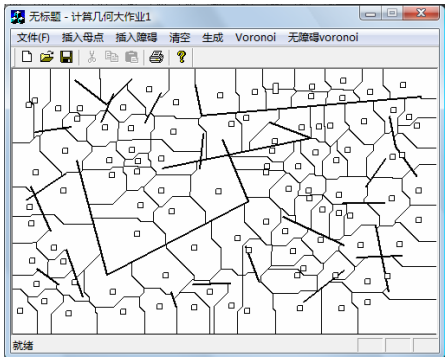
有障碍和无障碍 voronoi 图对比：黑粗线为障碍左为有障碍图，右为无障碍图



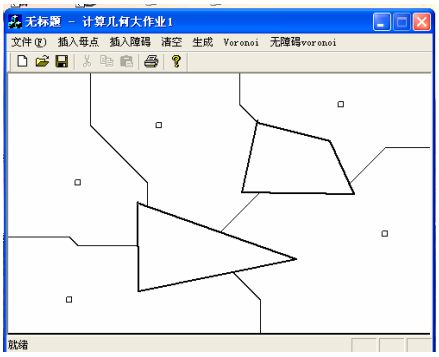


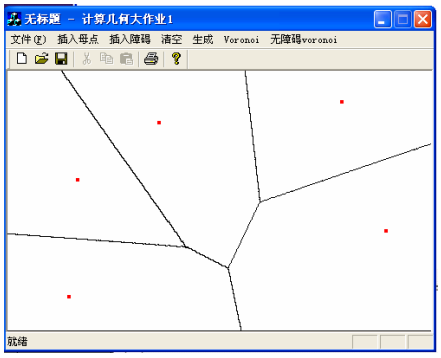
极端情况输入：

大量点，障碍：

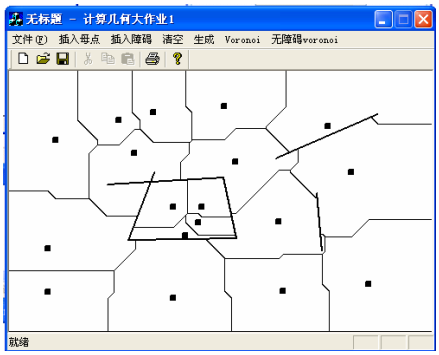


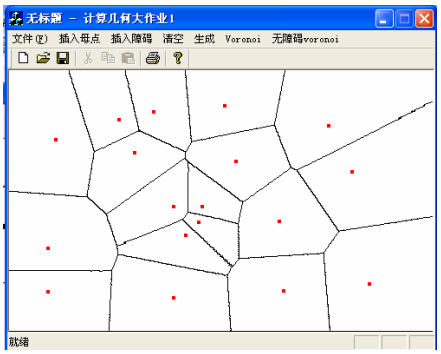
障碍为封闭形





母点在障碍内：





4.改进措施

因为实验采用的是菱形，最后的结果只是近似的边界，要想获得精确的结果，还是得由圆形进行扩散。

改进思路：输入母点使每个母点颜色不一样，输入障碍，使障碍和母点颜色不一样，障碍端点和障碍其他地方颜色不一样，由母点进行，向外一层层扩展画圆：

1. 遇到空白处，将该点置为母点的颜色
2. 遇到障碍颜色停止,则停止本圆的下一个点的检测，只画出到当前为止的部分圆弧
3. 遇到障碍端点，将端点加入链队，使之和母点一起同样的速度向外扩展画圆，颜色和首先遇到它的母点颜色一致。
4. 遇到其他母点颜色，不作处理

当屏幕被填满颜色，由同一颜色形成的区域,便是具有相同颜色母点的 Voronoi 区域。

再用本文的扫描方法生成 voronoi 图形。

经过老师的指点，报告在实验结果数据方面增加了部分样例，并对程序健壮性进行了佐证。本程序的时间复杂度跟点的数目无关，跟屏幕的大小相关，是采用离散化的思想。

本程序主要由杨玉来编写，王凡辅助完成。采用 **MFC** 编程，由于接触不多，对软件不太熟悉，还有许多不足之处，希望老师多多指教。

5.参考文献

障碍 Vorono i 图的结晶生成及其应用　　计算机应用与软件　　2007.8 第 24 卷

6.附程序主要代码：

```
#include <list>
#include <vector>
class cPOINT{
public:
 cPOINT(){}
 cPOINT(int xx,int yy,COLORREF cc):x(xx),y(yy),color(cc){}
 ~cPOINT(){}
 void operator =(cPOINT cp){
        x=cp.x;
        y=cp.y;
        color=cp.color;
 }
 int x,y;
 COLORREF color;
 };
list<cPOINT> q;
list<cPOINT> coreq;
list<cPOINT> voi_redraw;
int dcorep=0,dbarrier=0;
vector < vector<CPoint > > barriers;
vector<CPoint > temp;
// 中点画圆法
void cirpixel(HDC hDc,int x0,int y0,int x,int y,COLORREF color)
{
      if(GetPixel(hDc,x0+x,y0+y)==0x00ffffff)SetPixel(hDc,x0+x,y0+y,color);
if(GetPixel(hDc,x0+y,y0+x)==0x00ffffff)SetPixel(hDc,x0+y,y0+x,color);
if(GetPixel(hDc,x0-x,y0+y)==0x00ffffff)SetPixel(hDc,x0-x,y0+y,color);
if(GetPixel(hDc,x0+y,y0-x)==0x00ffffff)SetPixel(hDc,x0+y,y0-x,color);
```

```
if(GetPixel(hDc,x0+x,y0-y)==0x00ffffff)SetPixel(hDc,x0+x,y0-y,color);
if(GetPixel(hDc,x0-y,y0+x)==0x00ffffff)SetPixel(hDc,x0-y,y0+x,color);
if(GetPixel(hDc,x0-x,y0-y)==0x00ffffff)SetPixel(hDc,x0-x,y0-y,color);
if(GetPixel(hDc,x0-y,y0-x)==0x00ffffff)SetPixel(hDc,x0-y,y0-x,color);
}
void MidpointCircle(HDC hDc,int x,int y,int r,COLORREF color)
{
     int xx,yy;
     float d;
     xx=0;yy=r;d=1.25-r;
    cirpixel(hDc,x,y,xx,yy,color);
     while(xx<yy)
     {
          if(d<0)
          {
               d+=2*xx+3;xx++;
          }else
          {
               d+=2*(xx-yy)+5;xx++;yy--;
          }
          cirpixel(hDc,x,y,xx,yy,color);
     }

}
void CMy1View::OnLButtonDown(UINT nFlags, CPoint point)
{
cPOINT pp;
CClientDC dc(this);
if(dcorep==1){   // dcorep 为是否按下"插入母点"的标志
pp.x=point.x;
pp.y=point.y;
iii+=10;
pp.color=(0x00000001+iii*0x00030102)%0x00ffffff;
COLORREF color=pp.color;
CRect rect(point.x-3,point.y-3,point.x+3,point.y+3);
dc.FillRect(rect,&CBrush(pp.color));
q.push_back(pp);
coreq.push_back(pp);
}
if(dbarrier==1){      //// dcorep 为是否按下"插入障碍"的标志
 temp.push_back(point);
 if(temp.size()>1)
 {
       CPen pen;
```

```
        pen.CreatePen(PS_SOLID,1,RGB(0,0,0));
        dc.SelectObject(&pen);
        dc.MoveTo(temp[temp.size()-2]);
        dc.LineTo(point);
 }
        barries_flag=1;
}
}


void CMy1View::OnRButtonDown(UINT nFlags, CPoint point)
{
      core_flag=1;
 barriers.push_back(temp);
 if(temp.empty())barriers.pop_back();
 temp.clear();
 CView::OnRButtonDown(nFlags, point);
}

void CMy1View::OnInsert()   //插入障碍
{
 // TODO: Add your command handler code here
 barriers.push_back(temp);
 if(temp.empty())barriers.pop_back();
 temp.clear();
 barries_flag=1;
 dcorep=0;
 dbarrier=1;



}



void CMy1View::OnCoreV()    //插入母结点
{
 // TODO: Add your command handler code here
 dcorep=1;
 dbarrier=0;
}
void CMy1View::OnClear()   //清空所有数据
{
 dcorep=0;
```

```
 dbarrier=0;
 iii=1;
 q.clear();
 coreq.clear();
 temp.clear();
 barriers.clear();
 voi_flag=0;
 pre_voi=0;
 core_flag=0;
  barries_flag=0;
 Invalidate();
}
void CMy1View::OnCreat()   //进行预处理_染色
{
 if(coreq.empty())return;
 int color;
 cPOINT pt,p,temp;
 cPOINT cp;
 CClientDC dc(this);
 CRect rcClient;
GetClientRect (&rcClient);
 CRect rect(0,0,rcClient.Width()-1,rcClient.Height()-1);
);
 dc.FillRect(rect,&CBrush(0X00FFFFFF));
 list<cPOINT>::iterator c_iter=coreq.begin();
      while(c_iter!=coreq.end()){
            ::SetPixel(dc,(*c_iter).x,(*c_iter).y,(*c_iter).color);
            c_iter++;
      }
      for(int i=0;i<barriers.size();++i){
            vector<CPoint> tmp=barriers[i];
            dc.MoveTo(tmp[0]);
            for(int j=1;j<tmp.size();++j){
                  CPen pen;
                  pen.CreatePen(PS_SOLID,1,RGB(0,0,0));
                  dc.SelectObject(&pen);
                  dc.LineTo(tmp[j]);
            }
      }
      list<cPOINT>::iterator iter=q.begin();
      while(iter!=q.end())
      {
            pt=(*iter);
            color=::GetPixel(dc,pt.x,pt.y-1);          //上
```

```
            if(color==0X00ffffff&&pt.y>0)
            {
                ::SetPixel(dc,pt.x,pt.y-1,pt.color);
                temp.x=pt.x;temp.y=pt.y-1;temp.color=pt.color;
                q.push_back(temp);
            }

            color=::GetPixel(dc,pt.x,pt.y+1);                    //下
            if(color==0X00ffffff&&pt.y<rcClient.Height()-1)
            {
                ::SetPixel(dc,pt.x,pt.y+1,pt.color);
                temp.x=pt.x;temp.y=pt.y+1;temp.color=pt.color;
                q.push_back(temp);
            }
            color=::GetPixel(dc,pt.x-1,pt.y);                    //左
            if(color==0X00ffffff&&pt.x>0)
            {
                ::SetPixel(dc,pt.x-1,pt.y,pt.color);
                temp.x=pt.x-1;temp.y=pt.y;temp.color=pt.color;
                q.push_back(temp);
            }
            color=::GetPixel(dc,pt.x+1,pt.y);                    //右
            if(color==0X00ffffff&&pt.x<rcClient.Width()-1)
            {
                ::SetPixel(dc,pt.x+1,pt.y,pt.color);
                temp.x=pt.x+1;temp.y=pt.y;temp.color=pt.color;
                q.push_back(temp);
            }
            iter++;
        }
        list<cPOINT>::iterator iter1=coreq.begin();
        while(iter1!=coreq.end()){
            CRect rect((*iter1).x-3,(*iter1).y-3,(*iter1).x+3,(*iter1).y+3);
            dc.FillRect(rect,&CBrush(0x00000001));
            iter1++;
        }
 pre_voi=1;//重绘标志
 core_flag=0;
 barries_flag=0;
}
void CMy1View::OnVoronoi()
{
if(pre_voi==0)return;
 if(coreq.empty())return;
```

```
 CClientDC dc(this);
 int i,j;
 CRect rcClient;
  GetClientRect (&rcClient);
 for(j=0;j<rcClient.Height();++j)
        for (i=0;i<rcClient.Width()-1;++i)
        {
 if((::GetPixel(dc,i,j)!=::GetPixel(dc,i+1,j))||(::GetPixel(dc,i,j)!=::GetPixel(dc,i,j+1))){
                    ::SetPixel(dc,i,j,RGB(0,0,0));
                    cPOINT temp;
                    temp.x=i;
                    temp.y=j;
                    temp.color=RGB(0,0,0);
                    voi_redraw.push_back(temp);
               }
        }
 for(j=0;j<rcClient.Height();++j)
        for (i=0;i<rcClient.Width();++i)
             if(::GetPixel(dc,i,j)!=RGB(0,0,0))::SetPixel(dc,i,j,RGB(255,255,255));
 voi_flag=1;
 pre_voi=0;
 core_flag=0;
  barries_flag=0;
}
void CMy1View::Oncirvor()
{
 CClientDC dc(this);
 CRect rcClient;
     GetClientRect (&rcClient);
 CRect rect(0,0,rcClient.Width()-1,rcClient.Height()-1);
     dc.FillRect(rect,&CBrush(0X00FFFFFF));
 list<cPOINT>::iterator c_iter=coreq.begin();
       while(c_iter!=coreq.end()){
             ::SetPixel(dc,(*c_iter).x,(*c_iter).y,(*c_iter).color);
             c_iter++;
             }
     int flag=0;
 for(int r=1;;++r){
       list<cPOINT>::iterator iter1=coreq.begin();
       while(iter1!=coreq.end()){
 if((*iter1).x+r>rcClient.Width()+85&&(*iter1).x-r<-85&&(*iter1).y+r>rcClient.Height()+85&
&(*iter1).y-r<-85){
                    flag=0;
                    iter1++;
```

```
                continue;
            }
MidpointCircle(dc,(*iter1).x,(*iter1).y,r,(*iter1).color);
MidpointCircle(dc,(*iter1).x-1,(*iter1).y,r,(*iter1).color);
flag=1;
iter1++;
      }
      if(flag==0)break;
}
if(coreq.empty())return;
int i,j;
for(j=0;j<rcClient.Height();++j)
      for (i=0;i<rcClient.Width()-1;++i)
      {
if((::GetPixel(dc,i,j)!=::GetPixel(dc,i+1,j))||(::GetPixel(dc,i,j)!=::GetPixel(dc,i,j+1))){
                ::SetPixel(dc,i,j,RGB(0,0,0));
                cPOINT temp;
                temp.x=i;
                temp.y=j;
                temp.color=RGB(0,0,0);
                voi_redraw.push_back(temp);
            }
      }
      voi_flag=0;
      pre_voi=0;
      core_flag=0;
    barries_flag=0;
      for(j=0;j<rcClient.Height();++j)
            for (i=0;i<rcClient.Width();++i)
            if(::GetPixel(dc,i,j)!=RGB(0,0,0))::SetPixel(dc,i,j,RGB(255,255,255));
list<cPOINT>::iterator iter1=coreq.begin();
while(iter1!=coreq.end()){
CRect rect((*iter1).x-2,(*iter1).y-2,(*iter1).x+2,(*iter1).y+2);
dc.FillRect(rect,&CBrush(0x000000ff));
iter1++;
      }}
```