计算几何实验报告

三维凸包的实现

班级：计算机系计硕 99
实验人：乐秀宇、孙昱峰、邬浩
日期：2000 年 1 月 19 日
实验环境：Microsoft Visual C++ 6.0 和 Open Inventer

实验目的及要求：

在三维空间中任意给定 N 个点，N 大于等于 4，其中可能存在三点共线和四点共面的情况。

要求构造给定的这些点的三维凸包。

算法及其实现：

本程序是基于 Preparata-Hong 算法的。

基本的思想是使用“三维包扎”技术，对于给定的点集，找出其凸包的顶点以及相应的边，从而可以得到三维凸包的各个面。具体实现步骤如下：

首先将给定的点按某一坐标轴的方向进行排序。

```
void sort_points( int count )
{
    for( int i=0; i<count; i++ )
        for( int j=i+1; j<count; j++ )
        {
            if( points[i].y > points[j].y )
            {
                POINT3 pp = points[i];
                points[i] = points[j];
                points[j] = pp;
            }
        }
}
```

然后按此顺序以四个点为一组将给定的点分组。

```
int chull( int pos, int n, PPOLY &h )
{
    int n1,n2;
    PPOLY h1;
```

```
    PPOLY h2;
    if( n < 4 )
    {
        MessageBox( NULL, "输入的点数小于 4!!!", "错误", MB_OK );
        return -1;
    }
    if( n == 4 )
    {
        h = new( POLY );
        all_polys[polys_count++] = h;
        tetrahull( pos, h );
    }
    else
    {
        n1 = n / 2;
        n1 = n1 - n1 % 4;
        n2 = n - n1;
        if( (chull( pos, n1, h1 ) < 0) || (chull( pos + n1, n2, h2 ) < 0) )
            return -1;
        h = new( POLY );
        all_polys[polys_count++] = h;
        merge_3d( h1, h2, h );
    }
    return 0;
}
```

对每组中的四个点简单连线，得到一组原始的凸包。

```
void tetrahull( int pos, PPOLY h )
{
    PEDGE edges = new EDGE[6];
    all_edges[edges_count].e = edges;
```

```
    all_edges[edges_count++].count = 6;
    PPOINT3 p = new POINT3[4];
    all_points[points_count++] = p;
    make_tetra_edges( pos, p, edges );
    tetra_hull_2d( edges, p, h );
}
```

对每两个原始的凸包进行合并。先将这两个凸包按排序坐标轴方向进行投影，考虑各种情况找到支撑边。

```
    void merge_2d( PPOLY pa, PPOLY pb, PPOLY *pnew,
                   PEDGE *redge, PEDGE *ledge,
                   PEDGE *par, PEDGE *pbr )
    {
        PEDGE pal;
        PEDGE pbl;
        BOOL backmoved;

        *redge = new EDGE;
        (*redge)->v1 = new VERTEX;
        (*redge)->v2 = new VERTEX;
        *ledge = new EDGE;
        (*ledge)->v1 = new VERTEX;
        (*ledge)->v2 = new VERTEX;
        all_edges[edges_count].e = *redge;
        all_edges[edges_count++].count = 1;
        all_edges[edges_count].e = *ledge;
        all_edges[edges_count++].count = 1;

        *par = pa->rit2d;
        *pbr = pb->rit2d;
        if( abs_eq( pa->rit2d->v1->v->x, pb->rit2d->v1->v->x ) )
```

```
        (*pnew)->rit2d = *par;
    else if( pa->rit2d->v1->v->x < pb->rit2d->v1->v->x )
    {
        sup_line_fwd( par, pbr, &backmoved );
        if( backmoved )
            (*pnew)->rit2d = pb->rit2d;
        else
            (*pnew)->rit2d = *redge;
    }
    else
    {
        sup_line_bwd( pbr, par );
        (*pnew)->rit2d = pa->rit2d;
    }
    link_edge_2d( *redge, (*pbr)->bwd2d, *par );


    pal = pa->lef2d;
    pbl = pb->lef2d;
    if(abs_eq( pa->lef2d->v1->v->x, pb->lef2d->v1->v->x ))
        (*pnew)->lef2d = pbl;
    else if( pa->lef2d->v1->v->x < pb->lef2d->v1->v->x )
    {
        sup_line_fwd( &pbl, &pal, &backmoved );
        if( backmoved )
            (*pnew)->lef2d = pa->lef2d;
        else
            (*pnew)->lef2d = *ledge;
    }
    else
    {
```

```
            sup_line_bwd( &pal, &pbl );
            (*pnew)->lef2d = pb->lef2d;
        }
        link_edge_2d( *ledge, pal->bwd2d, pbl );

        if( (*redge)->v1->v == (*ledge)->v2->v )
            re_link_2d( *ledge, *redge );
        if( (*redge)->v2->v == (*ledge)->v1->v )
            re_link_2d( *redge, *ledge );
        if( (*pnew)->rit2d->v1->v == (*ledge)->v1->v )
            (*pnew)->rit2d = *ledge;
        if( (*pnew)->lef2d->v1->v == (*redge)->v1->v )
            (*pnew)->lef2d = *ledge;

}
```

将此支撑边还原至三维，过此边做一垂直于非排序坐标轴的坐标轴的有向平面，此时两原始凸包都在此平面的一侧。考察所有于支撑边顶点像邻的线段，对其中与所作平面夹角最大的一条线段作为新的支撑边，并继续上述动作，直到会到初始的支撑边，从而完成两个原始凸包的合并。

```
    void merge_3d( PPOLY p1, PPOLY p2, PPOLY px )
    {
        PPOINT3 c;
        PEDGE ecurr, erit, elef, ear, ebr, e1best,
                    e2best,oldebest, enext;
        double cot1, cot2;

        merge_2d( p1, p2, &px, &erit, &elef, &ear, &ebr );
        ecurr = erit;
        e1best = ear;
        e2best = ebr;
```

```
c = new POINT3;
all_points[points_count++] = c;

make_ref_pt( c, ecurr );
rev_edge( ecurr );
first_edge( RIGHT, c, ecurr, &e2best, &cot2 );
first_edge( LEFT, c, ecurr, &e1best, &cot1 );
do
{
	best_edge( LEFT, c, ecurr, &e1best, &cot1 );
	best_edge( RIGHT, c, ecurr, &e2best, &cot2 );
	if( cot1 < cot2 )
	{
		link_edge( &enext, e1best->v2->v, ecurr->v2->v,
				NULL, e1best, ecurr, NULL, elef, erit );
		ecurr->v1->nedgelef = e1best;
		ecurr->v2->nedgerit = enext;
		oldebest = e1best;
		e1best = e1best->v2->nedgelef;
		oldebest->v2->nedgelef = enext;
		oldebest->v1->nedgerit = ecurr;
		c = oldebest->v1->v;
	}
	else
	{
		link_edge( &enext, ecurr->v1->v, e2best->v2->v,
				NULL, ecurr, e2best, NULL, elef, erit );
		ecurr->v1->nedgelef = enext;
		ecurr->v2->nedgerit = e2best;
		oldebest = e2best;
```

```
                    e2best = e2best->v2->nedgerit;
                    oldebest->v1->nedgelef = ecurr;
                    oldebest->v2->nedgerit = enext;
                    c = oldebest->v1->v;
                }
                ecurr = enext;
            }while( ecurr != erit );
            rev_edge( erit );
            re_align_edges( erit );
}
```

对上述算法进行递归调用，最终完成给定点的三维凸包的实现。

```
int Hull3D( int count, PFACE faces , PPOINT3 *p )
{
    edges_count = 0;
    points_count = 0;
    polys_count = 0;
    add_points( count );
    sort_points( count );
    PPOLY h = NULL;
    if( chull( 0, count, h ) < 0 )
        return -1;
    int face_count = 0;
    list_face( h, faces, &face_count );
    free_mem();
    *p = points;
    return face_count;
}
```

显示实现：

应用 Open Inventer 将所有的给定点和每次形成的三角形平面，并且实现任意视角的旋

转和缩放。

由于显示依赖与环境，所以这里不便于重现。