

ACIS-HOOPS 新手上手学习方法

时间进度安排：

开始学习	-----1.5 天
环境准备	-----0.5 天
学会 ACIS 造型方法	-----1 天
在工程中实现对四面体的“实时拖动变形”功能	-----1 天
掌握 acispartview 工程功能实现机制	-----1 天
Acispartviewer 的修改：实现三维立方体的功能	-----2 天
参数化驱动 三维立方体的功能	-----2 天
三维立方体功能的产品化	-----2 天

如何学习 ACIS 和 HOOPS

学习前所具备的知识及能力：

具备 C/C++ 基础，MFC 基础，熟悉 pro_e UG 等 3D 软件的一种。

开始学习

首次接触 ACIS 和 HOOPS 不是马上进行学习，而是应该了解 HOOPS 和 ACIS 到底是是什么，有什么用。

ACIS 是什么？

（以下内容来自百度百科）

ACIS 是由美国 Spatial Technology 公司推出的，Spatial Technology 公司成立于 1986 年，并于 1990 年首次推出 ACIS。ACIS 最早的开发人员来自美国 Three Space 公司，而 Three Space 公司的创办人来自于 Shape Data 公司，因此 ACIS 必然继承了 Romulus 的核心技术。ACIS 的重要特点是支持线框、曲面、实体统一表示的非正则形体造型技术，能够处理非流形形体。

ACIS 是用 C++ 构造的图形系统开发平台，它包括一系列的 C++ 函数和类（包括数据成员和方法）。开发者可以利用这些功能开发面向终端用户的三维造型系统。ACIS 是一个实体造型器，但是线框和曲面模型也可以在 ACIS 中表示。ACIS 通过一个统一的数据结构来同时描述线框、曲面和实体模型，这个数据结构用分层的 C++ 类实现。ACIS 利用 C++ 的特点构造了标准的、可维护的接口。API 函数在不同 ACIS 版本之间保持一致性，而类及其接口函数则可能改变。ACIS 中应用到的主要 C++ 概念包括：数据封装、类构造重载、构造拷贝、类方法和操作符重载以及函数重载等。C++ 没有提供描述几何体的数学基本类，ACIS 提供了一些 C++ 基类实现这个功能，并且利用 C++ 的特性可以对它进行了扩充，这样 ACIS 就可以支持任意几何体的定义和构造功能。

ACIS 是美国 Spatial Technology 公司推出的三维几何造型引擎，它集线框、曲面和实体造型于一体，并允许这三种表示共存于统一的数据结构中，为各种 3D 造型应用的开发提供了几何造型平台。Spatial Technology 公司在 1986 年成立，目前 ACIS 3D Toolkit 在世界上已有 380 多个基于它的开发商，并有 180 多个基于它的商业应用，最终用户已近一百万。许多著名的大型系统都是以 ACIS 作为造型内核，如 AutoCAD，CADKEY，Mechanical Desktop，Bravo，TriSpectives，TurboCAD，Solid Modeler，Vellum Solid 等。

ACIS 主要功能是用来构建和保存读取实体数据，并对这些数据进行处理。注意：ACIS 无法在窗口中显示图形，你能看到的只是一些数据。

Hoops 是什么？

（以下内容来自百度百科）

HOOPS 3D Application Framework (HOOPS/3dAF)是由 Tech Soft America 公司开发并由 Spatial 再次销售的产品，该产品为当今世界上领先的 3D 应用程序提供了核心的图形架构和图形功能，这些 3D 应用程序涉及 CAD/CAM/CAE、工程、可视化和仿真等领域。有了 HOOPS/3dAF，用户就站在一个高起点上，能够快速和有效地开发和维护高性能的用户应用程序。用户通过将 HOOPS/3dAF 集成到相应的软件开发中，可以更好地管理开发成本、优化资源和缩短产品上市时间。

这里 HOOPS 可以将 ACIS 中的数据以图形的方式在窗口中显示出来。

学习过程中，可以花少量时间上网查询相关内容，了解 ACIS 和 HOOPS 的相关内容，方便更好的理解相关内容。（以上内容花半天的时间了解）

ACIS 的学习

了解完 ACIS 和 HOOPS 相关知识后，接下来我们要具体学习 ACIS 方面的相关知识。抓住一个核心：ACIS 的数据结构，即 ACIS 模型的拓扑结构，如下图，阅读教材：《基于 ACIS 的几何造型技术与系统开发》的第 1，2，3 章章节内容（4-12 章粗略阅读，有个大致印象即可，不必详读）。

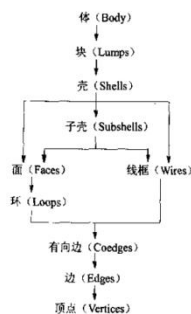


图 3.8 拓扑对象之间的关系

hoops 的学习

合理安排时间，在学习 ACIS 的同时，也要对 hoops 的相关内容有所掌握，结合 PPT 文档《HOOPS 基础培训课程》，理解 hoops 中相关组件的作用。

■ HOOPS/3dGS:

场景图 API

■ HOOPS/MVO:

实现了 3D 应用程序框架的功能

- 模型：文件的输入输出，模型的管理
- 显示：文字和相机的管理
- 操作：对象操作的管理

- HOOPS/MFC

封装了所需要的操作

- 与窗口的连接，获得窗口句柄和窗口的 ID 号
- 将鼠标和键盘事件映射到了 HOOPS/MVO
- 封装了剪贴板，打印机和打印机预览

- HOOPS/Stream

支持 HSF 的读写功能

- 数据是高度压缩的，大大缩短传输时间
- 数据的分类，流化处理

支持 2D 和 3D

- 支持 3dGS 中所有的几何体

- HOOPS/GM Bridge

连接 HOOPS 与建模内核（如 ACIS）

- 封装了连接模型与 HOOPS 几何的函数

将模型映射到 HOOPS 几何

- 读写 SAT 文件
- 选择与高亮显示的处理
- 当创建和更新模型的时候，图形数据也被创建和更新

- HOOPS ACIS Bridge 是 ACIS 组件的一部分

环境准备

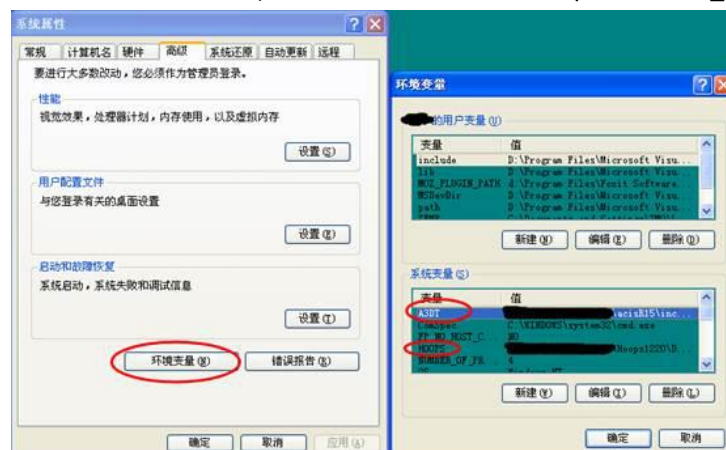
通过前面的内容学习，我们对 ACIS 和 HOOPS 作用及相关内容有了初步的认识，接下来将要搭建学习后续知识的平台。

acispartview 是一个简单的小型 ACIS/HOOPS 三维 CAD 平台, ACIS 是一个几何造型引擎, HOOPS 是图形显示平台, 通过学习该平台, 可以概况地了解基于 ACIS/HOOPS 的三维 CAD 系统的基本结构。(参考文档《ACIS-HOOPS 造型学习方法.doc》中第二部分, 工程设置入门)

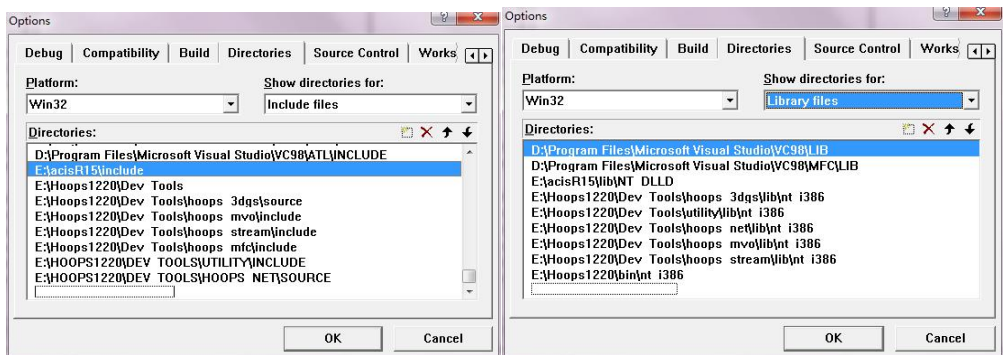
该平台的搭建工作可能遇到各种问题：（最好将 ACISR15 和 HOOPS1220 放在根目录下）

1 没有找到头文件和动态库文件

操作系统环境变量设置：添加如下环境变量：1)A3DT<自定义目录>\acisR15\include2)HOOPS<自定义目录>\Hoops1220\Dev Tool



在 VC6.0 中的菜单项 tools->options 的 Directories 选项卡中 include files 和 Library files 的 Directories 添加:



2 运行 activeproject.exe 时发现缺少 dll 文件

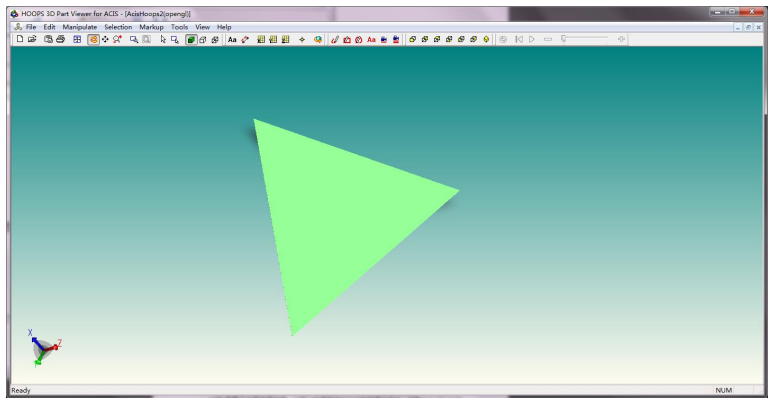
SpaBridged.dll	285 KB	应用程序扩展	2010-5-27 10:46
SpaBased.dll	2,493 KB	应用程序扩展	2005-10-13 16:36
SpaVisd.dll	961 KB	应用程序扩展	2005-10-17 13:06
SpaACISd.dll	21,041 KB	应用程序扩展	2005-10-17 13:08
hoops_stream1220d.dll	1,617 KB	应用程序扩展	2005-11-30 20:54
hoops_net1220d.dll	209 KB	应用程序扩展	2005-11-30 20:54
hoops_mv1220d.dll	1,221 KB	应用程序扩展	2005-11-30 20:54
hoops_mv1220d.dll	73 KB	应用程序扩展	2005-7-20 18:25
hoops1220d.dll	7,133 KB	应用程序扩展	2008-11-6 9:47
acishoops3dpartviewerd.exe	1,357 KB	应用程序	2010-5-27 10:46
acishoops3dpartviewerd.ilk	3,463 KB	Intermediate file	2010-5-27 10:46

找到上图中的文件并将其放在 activeproject.exe 的文件夹内即可。

如遇到其它问题可参考文档《ACIS-HOOPS 造型学习方法.doc》中工程设置的内容。

工程运行及效果

将工程运行起来，加载部分 sat 文件，观察并实验工具栏上部分按钮的功能及效果。



掌握工程中的基础技术

学会 ACIS 造型方法

活学活用 ACIS 造型方法，可以从构造一个几何体开始。仔细阅读并理解教材《基于 ACIS 的几何造型技术与系统开发》中第十三章，第十四章的内容。掌握构造一个实体模型的基本方法和各个 API 的使用方法。教材中涉及的 Scheme 语言暂不学习。

以下是构建一个实体的基本方法：

```

api_start_modeler(0); //生成内部数据结构
api_initialize_ "组件";
//api 函数和直接函数调用
api_terminata_ "组件";
api_stop_modeler(); //删除内部数据结构

```

例子:

```

void main(){
    api_start_modellor(0);           //启动造型器
    api_initialize_constructors();    //初始化应用程序组件
    BODY *block;                     //声明一个指针
    api_make_cuboid(100,50,200,block); //调用 ACIS 的 API 函数构造几何体
    FILE *output = fopen("cube.dbg","w"); //打开一个文件
    debug.size(block, output);        //调用调式功能的函数将数据结构存入文件
    fclose(output);                   //关闭文件
    api_terminate_constructors();      //中断组件，释放组件所占用的组件
    api_stop_modeller();              //停止造型器
}

```

利用 ACIS 和 HOOPS 来开发三维图形软件，编程时的基本思路：

启动：

```
api_start_modeller(0);
```

.....

造型：

```

HC_Open_Segment_By_Key(m_pHView->GetViewKey());
BODY* sphere_body;
api_make_sphere(radius, sphere_body);

```

.....

转换：

```

HA_Set_Rendering_Option();
HA_Render_Entity ((ENTITY*) sphere_body);
HC_Close_Segment();

```

.....

终止：

```

HA_Close();
api_stop_modeller();

```

在工程中实现对四面体的“实时拖动变形”功能

- 1 建立 ACIS/HOOPS 造型平台 acispartview。
- 2 在 acispartview 中构造 ACIS 数据结构的四面体模型。
参考教材 14.5。
- 3 实现在 Hoops 中显示该四面体模型。

参考[程序 1](#)。添加了两个成员变量: m_iIndexVertex、m_positon[4]。

4 响应顶点 (Vertex) 被选中的消息。

参考[程序 2](#)。

5 实现“实时拖动变形”功能: 将某一顶点拖动到屏幕任意位置后, 重新构造新四面体。

参考[程序 3](#)。

6 完善该程序。

6.1 将构件四面体的功能添加到菜单功能中;

6.2 拖拽鼠标, 构造四面体动态线框, 以表达拖拽的实时性。

参考程序 1:

“CSolidHoopsView.h”下添加如下两个成员变量:

public:

```
int m_iIndexVertex;           //标示当前被选中顶点的序号@zxc
SPAposition* m_positon[4];    //四面体的四个顶点@zxc
```

void CSolidHoopsView::OnTetra()

```
{
    HSolidModel* pSolidModel = (HSolidModel*)(m_pHView->GetModel());
    if (!pSolidModel)
        return ;
    pSolidModel->m_bSolidData = true;    // 将 m_bSolidData 设为 TURE, 则可以选中顶点
    pSolidModel->m_bSolidModel = true;
    pSolidModel->SetBRepGeometry(true);
    pSolidModel->SetModelHandedness(HandednessSetLeft);
    //构造一个四面体
    m_positon[0] = new SPAposition(0.0,0.0,0.0);
    m_positon[1] = new SPAposition(10.0,0.0,0.0);
    m_positon[2] = new SPAposition(0.0,10.0,0.0);
    m_positon[3] = new SPAposition(0.0,0.0,10.0);
    BODY *block;
    Tetrahedron(block);           //四面体的构造函数, 参考教材 14.5
    pSolidModel->AddAcisEntity(block);
    HC_Open_Segment_By_Key(m_pHView->GetModelKey());
    HA_Render_Entity(block); // ha_bridge 将 ACIS 几何体转换为 Hoops 几何体
    HC_Close_Segment();
    m_pHView->Update();
}
```

参考程序 2:

```
void CSolidHoopsView::OnLButtonDown(UINT nFlags, CPoint point)
```

```

{
    CHoopsView::OnLButtonDown(nFlags,point);
    //判断顶点是否选中，如果选中，记录被选择点的序号
    HSelectionSet* sel_set = m_pHView->GetSelection();
    const HSelectionItem *selItem=sel_set->GetSelectionItemAt(0);//@zxc
    if (selItem != NULL)
    {
        HC_KEY hKey=selItem->GetKey();
        if (hKey!=NULL)
        {
            char type[MVO_BUFFER_SIZE];
            HC_Show_Key_Type(hKey, type);
            char* pdest;
            pdest=strstr(type,"marker");
            if (pdest != NULL)
            {
                // AfxMessageBox(_T("选中的对象是顶点！"));
                ENTITY* enty=HA_Compute_Entity_Pointer(hKey);
                VERTEX* vertex=(VERTEX*)enty;
                SPAPosition pos=vertex->geometry()->coords();
                for (int i=0;i<=3;i++)
                {if (*m_positon[i] == pos)
                    {m_iIndexVertex=i+1;
                     break;}}}}}}
            }
        }
    }
}

```

参考程序 3:

```

void CSolidHoopsView::OnLButtonUp(UINT nFlags, CPoint point)
{
    HSolidModel* pSolidModel = (HSolidModel*)(m_pHView->GetModel());
    if (!pSolidModel)
        return ;
    if (m_iIndexVertex!=0)
    {
        pSolidModel->DeleteAllEntities();//删除原来的四面体，再重构一个新的四面体
        HC_Open_Segment_By_Key(m_pHView->GetSceneKey());
        HPoint localPixel,world;
        localPixel.Set(point.x,point.y,0.0);
        HC_Compute_Coordinates(".", "local pixels",&localPixel,"world",&world);
        m_positon[m_iIndexVertex-1]->set_x(world.x);
        m_positon[m_iIndexVertex-1]->set_y(world.y);
        m_positon[m_iIndexVertex-1]->set_z(world.z);
        m_iIndexVertex=0;//标示符 m_iIndexVertex 清零，重新构造四面体后，
        HC_Close_Segment();           可对四面体反复拖拽。
        BODY *block;
        Tetrahedron(block);
        pSolidModel->AddAcisEntity(block);
        HC_Open_Segment_By_Key(m_pHView->GetModelKey());
        HC_Insert_Line(0.0,0.0,0.0,20.0,20.0,20.0);
    }
}

```



```

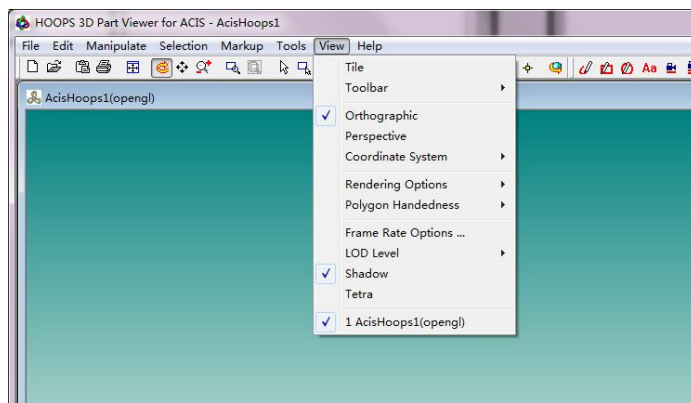
        HA_Render_Entity(block);
        HC_Close_Segment();
        m_pHView->Update();
    }
    CHoopsView::OnLButtonUp(nFlags,point);}

```

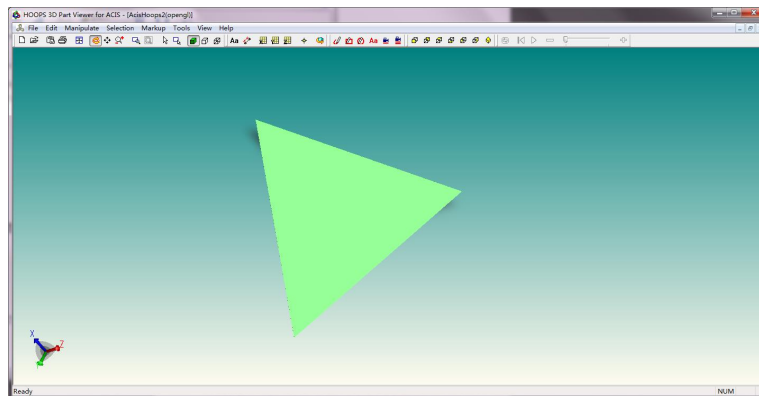
掌握 acispartview 工程功能实现机制

通过上面的例子，理解工程中部分功能模块的实现机制

例 1:




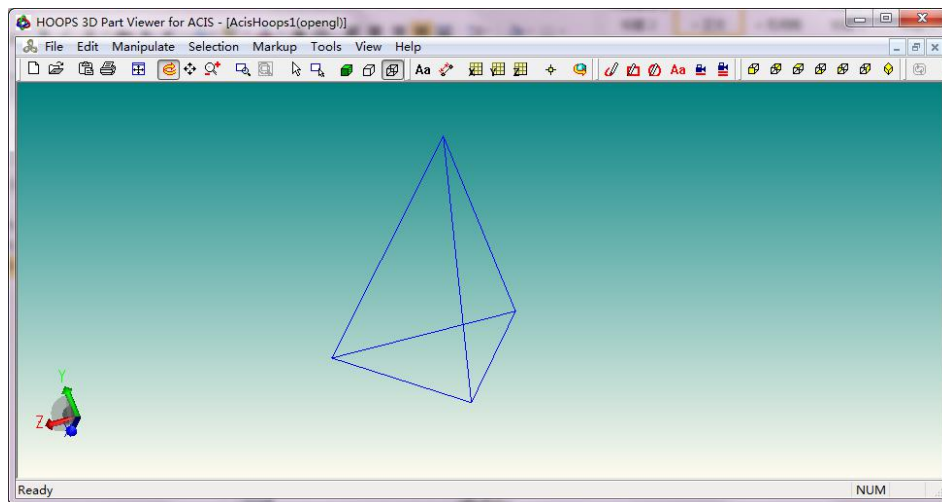
点击菜单项 View->Tetra，在窗口中显示了一个四面体



它的实现机制是：点击 tetra 后，产生了一个消息 COMMAND 消息，该消息 ID 是 IDM_TETRA，根据消息 ID，系统调用 OnTetra()函数，OnTetra()函数里面封装了四面体的实现过程。

例 2:

点击  按钮，窗口中四面体显示为实线

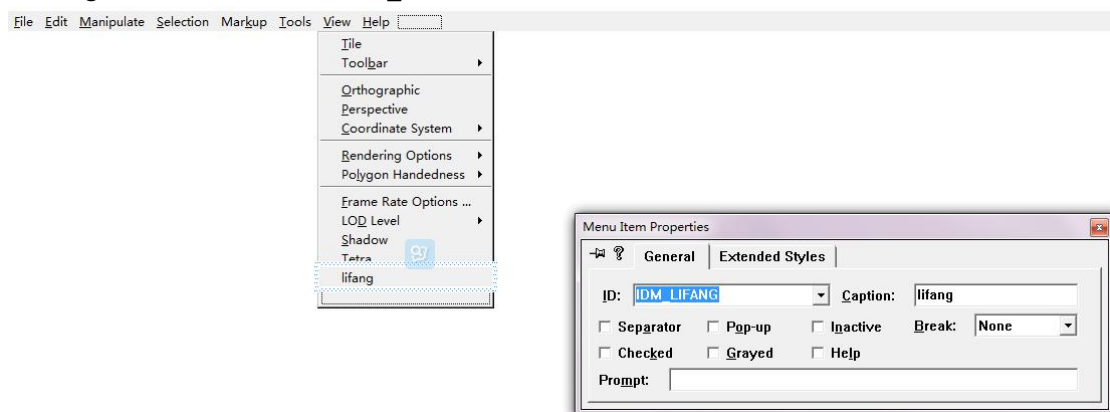


点击 Wireframe 按钮后，产生了一个 COMMAND 消息，消息 ID 为 ID_TOOLS_RENDERMODE_WIREFRAME，系统根据消息 ID 调用函数，CSolidHoopsView::OnToolsRendermodeWireframe()，该函数封装了将实体用实线表示的方式，同时也产生了一个 ON_UPDATE_COMMAND_UI 消息，保证了三个 toolbars 只激活其中一个。

通过以上例子的学习，利用 VC6.0，查看源代码，使我们对 acispartviewer 工程中的功能实现有了初步的了解，工具栏和菜单栏中功能的实现基本与上面的例子类似。我们在学习过程中，还应该不断的积累文档内容，了解相关 API，这样保证我们在今后的工作中能更加得心应手。

Acispartviewer 的修改：实现三维立方体的功能

在 acispartviewer 窗口中构建一个立方体，首先在资源文件的菜单选项里添加菜单项 (lifang)，设置其消息 ID (IDM_LIFANG)



在 CSoidHoopsView 类的头文件中添加消息处理函数 OnLifang ()，

```
afx_msg void OnToolsLightsLightGeometryOn();
afx_msg void OnToolsVisibilityMarkersOnly();
afx_msg void OnUpdateToolsVisibilityMarkersOnly(CCmdUI* pCmdUI);
afx_msg void OnTetra();
afx_msg void OnLifang(); // 实现立方体
//}}AFX_MSG
#ifdef ACIS
afx_msg void OnUpdateQuestTools(CCmdUI* pCmdUI);
```

在 CSolidHoopsView 类的实现文件中添加消息映射 ON_COMMAND,

```
BEGIN_MESSAGE_MAP(CSolidHoopsView, CHoopsView)
//{{AFX_MSG_MAP(CSolidHoopsView)
ON_COMMAND(ID_FILE_SAVE_AS, OnFileSaveAs)
ON_COMMAND(ID_QUERY, OnQuery)
ON_COMMAND(ID_SUBTRACT, OnSubtract)
ON_COMMAND(ID_UNION, OnUnion)
ON_COMMAND(ID_ZOOM, OnZoom)
ON_COMMAND(ID_ZOOM_TO_EXTENTS, OnZoomToExtents)
ON_COMMAND(ID_ZOOM_TO_WINDOW, OnZoomToWindow)
ON_COMMAND(ID_ORBIT, OnOrbit)
ON_COMMAND(ID_PAN, OnPan)
ON_COMMAND(ID_APERTURE_SELECT, OnApertureSelect)
ON_COMMAND(ID_WINDOW_SELECT, OnWindowSelect)
ON_COMMAND(ID_INTERSECT, OnIntersect)
ON_COMMAND(IDM_TETRA, OnTetra) // @zxc
ON_COMMAND(IDM_LIFANG, OnLifang) // 立方体实现
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

以上内容可直接用 VC6.0 中 MFC ClassWizard 自动生成, 然后编写 OnLifang() 函数实现内容:

```
*/
void CSolidHoopsView::OnLifang()
{
    HSolidModel* pSolidModel = (HSolidModel*)(m_pHView->GetModel());
    if(!pSolidModel) return;

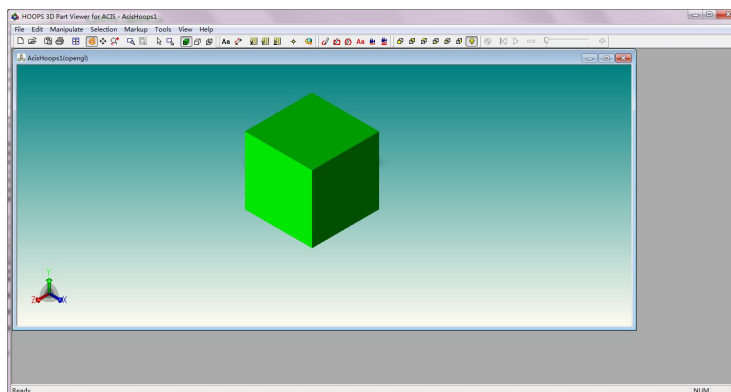
    pSolidModel->m_bSolidData = true;
    pSolidModel->m_bSolidModel = true;
    pSolidModel->SetBRepGeometry(true);
    pSolidModel->SetModelHandedness(HandednessSetLeft);

    BODY *block;

    API_BEGIN
        api_make_cuboid(10,10,10,block);
    API_END

    pSolidModel->AddAcisEntity(block);
    HC_Open_Segment_By_Key(pSolidModel->GetModelKey());
    HA_Render_Entity(block);
    HC_Close_Segment();

    m_pHView->Update();
}
```



实现效果:

Acispartviewer 参数化驱动 三维长方体的功能

在实现立方体的功能上，我们将实现长方体的功能，长方体的长、宽、高数值由我们自己设置。

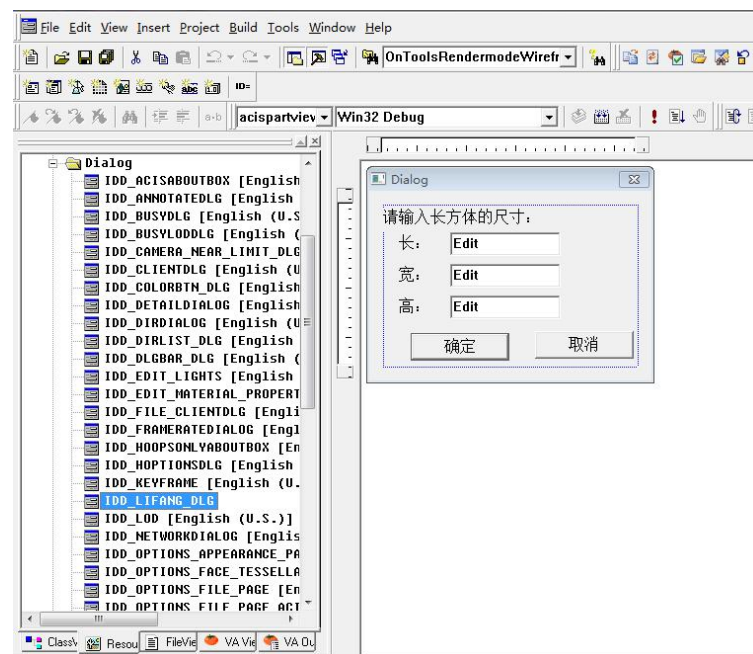
思路：在菜单栏或者工具栏增加一个菜单项，消息映射函数里创建一个模式对话框，该对话框里设置数值，点击确定创建长方体，点击取消，不做任何操作。

例：

首先在资源文件的菜单选项里添加菜单项（lifang），设置其消息 ID（IDM_LIFANG）添加对应的消息函数 OnLifang()，方法与前面例子相同。

设置全局变量，用来存放立方体的长宽高等数据。

在资源菜单里添加一个对话框窗口，具体设置如图：



创建该对话框类 CLifangDlg。重写 OnOK（）函数，实现当点击确定按钮时，将编辑框内的数据传给全局变量。

```
void CLifangDlg::OnOK()
{
    // TODO: Add extra validation here
    CString strlong, strweight, strhight;
    GetDlgItemText(IDC_EDIT_LONG, strlong);
    GetDlgItemText(IDC_EDIT_WEIGHT, strweight);
    GetDlgItemText(IDC_EDIT_HIGHT, strhight);
    g_dblong = _ttoi(strlong);
    g_dbweight = _ttoi(strweight);
    g_dbhight = _ttoi(strhight);

    CDialog::OnOK();
}
```

在 OnLifang（）函数实现该长方体创建的功能。

```
#include "LifangDlg.h"
extern int g_dblong,g_dbweight,g_dbhight;
void CSolidHoopsView::OnLifang()
{
    CLifangDlg lifangdlg;
    int iRet = lifangdlg.DoModal();
    if (iRet == IDCANCEL)
    {
        return;
    }

    HSolidModel* pSolidModel = (HSolidModel*)(m_pHView->GetModel());
    if(!pSolidModel) return;

    pSolidModel->m_bSolidData = true;
    pSolidModel->m_bSolidModel = true;
    pSolidModel->SetBRepGeometry(true);
    pSolidModel->SetModelHandedness(HandednessSetLeft);

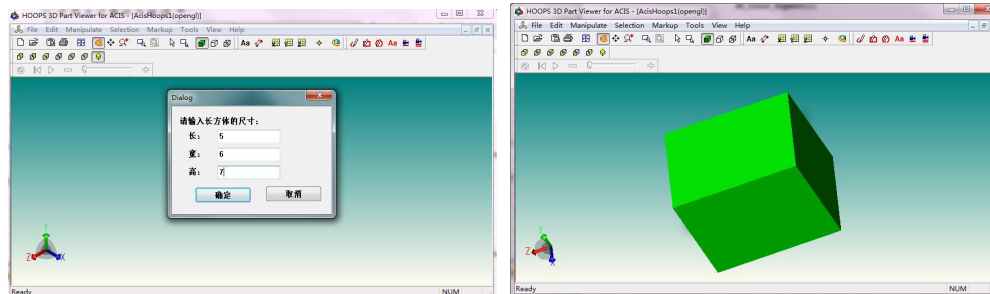
    BDDY *block;

    API_BEGIN
        api_make_cuboid(g_dblong,g_dbweight,g_dbhight,block);
    API_END

    pSolidModel->AddAcisEntity(block);
    HC_Open_Segment_By_Key(pSolidModel->GetModelKey());
    HA_Render_Entity(block);
    HC_Close_Segment();

    m_pHView->Update();
}
```

实现效果：



思考：

该功能是否还有需要完善的地方？

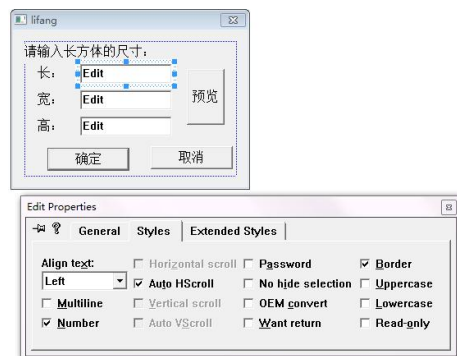
如编辑框内数值为 0 时，程序崩溃。

三维立方体功能的产品化

针对上面例子中部分功能不完善处，下面我们来实现改进工作。

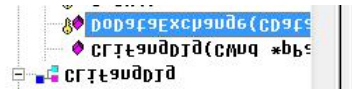
1 编辑框内可以填写除数字以外的符号。

打开该对话框的资源文件，设置其属性 Properties 的 styles 选项卡中勾选 Number 选项。



2 编辑框内设置初始值。

打开编辑框的类，在 DoDataExchange（）函数中对编辑框内的内容进行初始化。

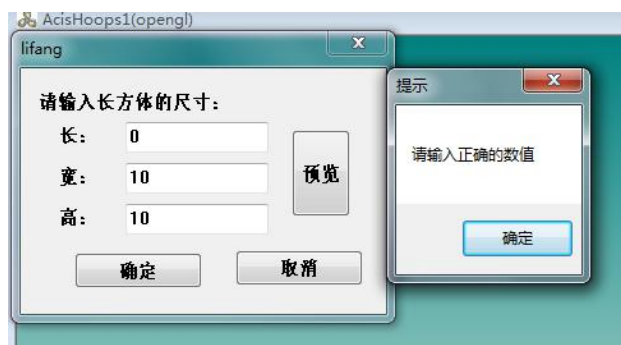


初始化代码如下

```
void CLifangDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //初始化编辑框
    CString strlong = "10",strweight = "10",strheight = "10";
    GetDlgItem(IDC_EDIT_LONG)->SetWindowText(strlong);
    GetDlgItem(IDC_EDIT_WEIGHT)->SetWindowText(strweight);
    GetDlgItem(IDC_EDIT_HEIGHT)->SetWindowText(strheight);
    //{{AFX_DATA_MAP(CLifangDlg)
    // NOTE: the ClassWizard will add DDX and DDU calls here
    //}}AFX_DATA_MAP
}
```

3 编辑框内设置数值为 0 时程序会崩溃

解决数值为 0 的问题首先要对编辑框内得到的数据进行判断，判断为非正数的时候不能通过，达到如下效果



为了达到该效果，就需要对点击确定时进行处理，当数据正确时，能正常的进行后续操作，当数据不对时，弹出对话框提示输入正确的数据，并继续进行输入数据操作。这里判断位置仍然在 OnOk（）函数中进行判断。代码如下：


```

void CLifangDlg::OnOK()
{
    // TODO: Add extra validation here
    CString strlong, strweight, strheight;
    GetDlgItemText(IDC_EDIT_LONG, strlong);
    GetDlgItemText(IDC_EDIT_WEIGHT, strweight);
    GetDlgItemText(IDC_EDIT_HEIGHT, strheight);
    g_dblong = _ttoi(strlong);
    g_dbweight = _ttoi(strweight);
    g_dbheight = _ttoi(strheight);
    g_clocle = 0;

    //解决编辑框输入0时程序崩溃问题

    if (g_dblong > 0 && g_dbweight > 0 && g_dbheight > 0)
    {
        CDialog::OnOK();
    }
    else
    {
        MessageBox(L"请输入正确的数值", L"提示", MB_OK);
        return;
    }
}

```

添加条件判断即可解决该问题。

4 添加预览功能

首先，要添加预览功能，则需要对对话框不关闭的前提下进行，否则无法在窗口中显示我们需要的图形。前面的对话框都是在建立的模式对话框，达到预览功能则需要建立非模式对话框。

创建具有预览功能的对话框。



① 建立非模式对话框

创建模式对话框的方法如下：

```

CBookinfo Bookinfo;
Bookinfo.DoModal();

```

首先自定义 CBookinfo 类是派生于对话框类，定义一个 CBookinfo 类的对象 Bookinfo，使用 Bookinfo 对象调用类中的成员函数，显示模式对话框。

创建非模式对话框的方法如下：

```

m_BookinfoDlg = new CBookinfo();
m_BookinfoDlg->Create(IDD_DIALOG1, this);
m_BookinfoDlg->ShowWindow(SW_SHOW);

```

为 CBookinfo 类对象分配内存空间，指针 pBookinfo 指向这个内存的地址。通过指针调用类中成员函数 Create 创建非模式对话框，再调用 ShowWindow 函数将其显示出来。

我们例子中代码只需在菜单项 flifang 的响应函数中添加即可。

```
void CSolidHoopsView::OnFLifang()
{
    CLifangDlg *plifangdlg = new CLifangDlg();
    plifangdlg->Create(IDD_LIFANG_DLG, this);
    plifangdlg->ShowWindow(SW_SHOW);
}
```

- ② 点击确定、预览等按钮都可达到显示立方体的功能，那么就需要用消息来传递，当视图窗口接收到了对话框发来的消息，即可在窗口显示图形。

下面，首先要在 CSolodHoopsView 中添加自定义的消息处理函数
在头文件中添加自定义的消息 ID

```
#define WM_MYMSG (WM_USER+831)
```

在 CSolodHoopsView 添加消息处理函数

```
ON_COMMAND(IDM_FLIFANG, OnFLifang) // 立方体头现
ON_MESSAGE ( WM_MYMSG, OnFLifang1 ) // 消息处理
ON_COMMAND(IDM_MYCLIPBOARD, OnMyClipboard)
```

自定义消息处理函数 OnFLifang1 的代码，实际上就是调用 Onlifang () 函数进行长方体的构造。

```
RESULT CSolidHoopsView::OnFLifang1(WPARAM wParam, LPARAM lParam)
{
    OnLifang();
    return 0;
}
```

实现对话框在点击确定和预览时都发送 WM_MYMSG 消息，能够让视图窗口接收到。
在它们的代码中添加如下代码即可发送消息

```
{
    CMDIFrameWnd *pFrame;
    CMDIChildWnd *pChild;
    CView *pView;
    pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd;
    pChild = (CMDIChildWnd *) pFrame->GetActiveFrame();
    pView = pChild->GetActiveView();
    if(pView != NULL)
        pView->PostMessage(WM_MYMSG, 0, 0);
}
```

点击预览和确定按钮代码有所区别，点击预览对话框不关闭，具体代码如下：

<pre>void CLifangDlg::OnYlan() { // TODO: Add your control notification handler code here CString strlong, strweight, strheight; GetDlgItemText(IDC_EDIT_LONG, strlong); GetDlgItemText(IDC_EDIT_WEIGHT, strweight); GetDlgItemText(IDC_EDIT_HEIGHT, strheight); g_dblong = _ttoi(strlong); g_dbweight = _ttoi(strweight); g_dbheight = _ttoi(strheight); g_clocle = 255; if (g_dblong > 0 && g_dbweight > 0 && g_dbheight > 0) { CMDIFrameWnd *pFrame; CMDIChildWnd *pChild; CView *pView; pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd; pChild = (CMDIChildWnd *) pFrame->GetActiveFrame(); pView = pChild->GetActiveView(); if(pView != NULL) pView->PostMessage(WM_MYMSG, 0, 0); } else { MessageBox(L"请输入正确的数值", L"提示", MB_OK); return; } }</pre>	<pre>void CLifangDlg::OnOK() { // TODO: Add extra validation here CString strlong, strweight, strheight; GetDlgItemText(IDC_EDIT_LONG, strlong); GetDlgItemText(IDC_EDIT_WEIGHT, strweight); GetDlgItemText(IDC_EDIT_HEIGHT, strheight); g_dblong = _ttoi(strlong); g_dbweight = _ttoi(strweight); g_dbheight = _ttoi(strheight); g_clocle = 0; //解决编辑框输入时程序崩溃问题 if (g_dblong > 0 && g_dbweight > 0 && g_dbheight > 0) { CMDIFrameWnd *pFrame; CMDIChildWnd *pChild; CView *pView; pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd; pChild = (CMDIChildWnd *) pFrame->GetActiveFrame(); pView = pChild->GetActiveView(); if(pView != NULL) pView->PostMessage(WM_MYMSG, 0, 0); } CDialog::OnOK(); } else { MessageBox(L"请输入正确的数值", L"提示", MB_OK); return; }</pre>
---	---

备注：熟悉 MFC 中自定义消息传递的用法（具体内容查资料）

三、同一进程里发送消息

1.发送消息

```
void CSendMessageDlg::OnBnClickedButtonSend()
{
    CString* msg = new CString("发送的字符串");
    ::SendMessage(m_hWnd,WM_USER+1,0,(LPARAM)msg);
    delete msg;
}
```

2.添加消息响应函数

(1) SendMessageDlg.h 添加

```
afx_msg HRESULT OnClickBtn1(WPARAM,LPARAM);
```

(2) SendMessageDlg.cpp

BEGIN_MESSAGE_MAP

END_MESSAGE_MAP()中间

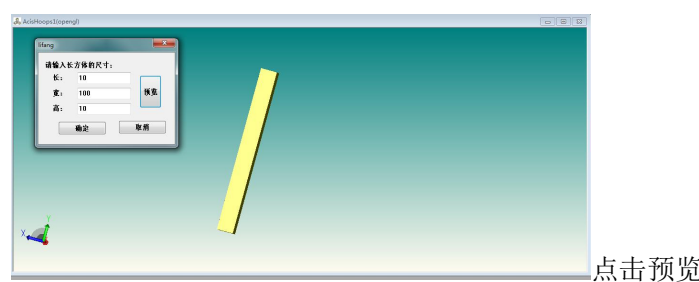
```
ON_MESSAGE(WM_USER+1,OnClickBtnSend)
```

(3) 实现函数

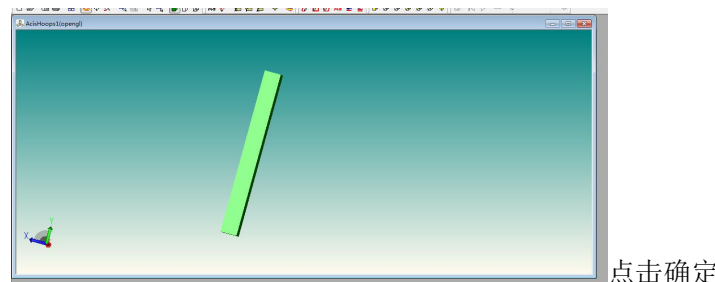
```
HRESULT CSendMessageDlg::OnClickBtn1(WPARAM wParam,LPARAM lParam)
{
    CString* rmsg = (CString*)lParam;
    MessageBox(*rmsg);
    return TRUE;
}
```

3.点击发送，响应消息

- ③ 实现预览时颜色和点击确定颜色不相同。
设置预览时，实现的颜色为黄色，实现时为绿色，调用 `api_rh_set_entity_rgb()` 对实体进行着色，该函数的用法如下：
`api_rh_set_entity_rgb((ENTITY*)&body,rgb_color(r,g,b));`
//其中，`api_gi_set_entity_rgb` 是着色函数，`(ENTITY*)&body` 指定实体，`body,rgb_color(r,g,b)` 进行着色。
达到的效果如下

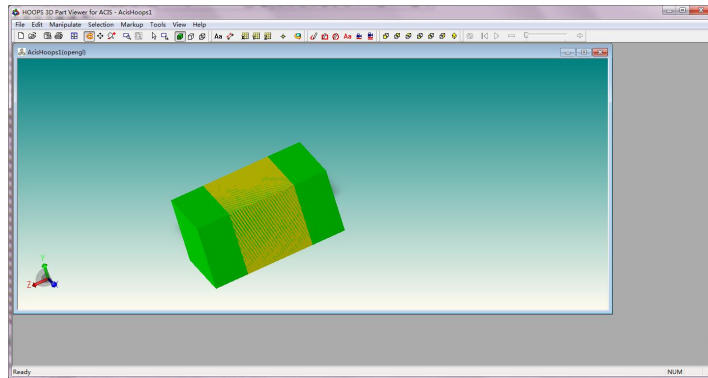


点击预览



点击确定

5 视图中显示多个立方体，预览颜色和非预览颜色同时出现，如下图



解决该问题时，将实体类定义为全局变量，前面我们都是用的局部变量，定义后，每次显示时将原实体取消显示即可达到效果，添加代码

```
-----  
void CSolidHoopsView::OnLiFang()  
{  
    HSolidModel* pSolidModel = (HSolidModel*)(n_pHView->GetModel());  
    if(!pSolidModel) return;  
  
    // 实体显示前清楚前次实体  
    HC_Open_Segment_By_Key(pSolidModel->GetModelKey());  
    HA_Delete_Entity_Geometry(block);  
    HC_Close_Segment();  
  
    API_BEGIN  
        api_make_cuboid(g_dblong,g_dbweight,g_dbhight,block);  
        api_rh_set_entity_rgb(block,rgb_color(g_clocle,255,0));  
    API_END  
  
    HC_Open_Segment_By_Key(pSolidModel->GetModelKey());  
    HA_Render_Entity(block);  
    HC_Close_Segment();  
  
    n_pHView->Update();  
}
```

附录：（以下内容仅供参考）

ACIS 的 API 相关函数：

```
api_make_... 坐标原点构建
api_solid_... 给定位置构建
//立方体
api_make_cuboid(length(x),width(y),height(z),BODY)
api_solid_block(SPAposition(左上角顶点坐标), SPAposition(右下角顶点坐标),BODY)
//球体
api_make_sphere(半径, BODY)
api_solid_sphere(SPAposition(圆心坐标),半径,BODY)
//圆环体
api_make_torus(外环半径,环宽,BODY)
api_solid_torus(SPAposition(环心坐标), 外环半径,环宽,BODY)
//圆锥体
api_make_frustum(height(z), length(x),width(y),顶部半径,BODY)
api_solid_cylinder_cone(SPAposition(顶部圆心坐标), SPAposition(底部圆心坐标),m*M_PI(底部
长轴),n*M_PI(底部短轴),O(顶部半径),NULL,BODY)
//圆柱体
api_make_frustum(height(z), length(x),width(y),顶部半径,BODY)
api_solid_cylinder_cone(SPAposition(顶部圆心坐标), SPAposition(底部圆心坐标),m*M_PI(底部
长轴),n*M_PI(底部短轴),O(顶部半径),NULL,BODY)
//棱柱
api_make_prism(height(z), length(x),width(y),棱数,BODY)
//棱锥
api_make_pyramid(height(z),length(x),width(y),顶部半径,棱数,BODY)
```

```
outcome api_ent_area ( ENTITY * ent,
double req_rel_accy,
double & area,
double & est_rel_accy_achieved,
AcisOptions * ao = NULL
) //计算面积
```

api_set_file_info 设置文件头部信息，生成模型文件时这些信息会被自动加入到文件的头部
api_save_entity_list 将实体的 ASCII 信息写入文件
api_restore_entity_list 将磁盘上的 SAT 文件读入到内存中

ENTITY_LIST 类的成员函数

int add(ENTITY*) 将实体加入列表

int count() 返回列表中实体的个数，包括被删除的实体

int iteration_count() 返回列表中实体的个数，不包括被删除的实体

int lookup(ENTITY*) 在列表中查找实体

int remove(ENTITY*) 在列表中删除实体

ENTITY *operator[](int) 返回给定位置的实体指针

void init() 准备进行列表遍历

ENTITY* next() 依次返回列表中的实体

api_apply_transf(BODY,transf) // 旋转实体

api_unite(body1,body2) //实体求并 结果保存在第二个变量中，第一个变量所表示的实体将被系统删除