**1、创建NXhello界面**

**代码：**

#include <uf\_ui.h>//包含UF\_initialize（）和UF\_terminate（）函数原型的头文件

#include <uf.h>//包含uc1601（）函数原型的头文件

extern void ufusr(char \*param, int \*retcode, int param\_len)

{

if (UF\_initialize()!=0)//获取NX openapi的执行权限

{

return;

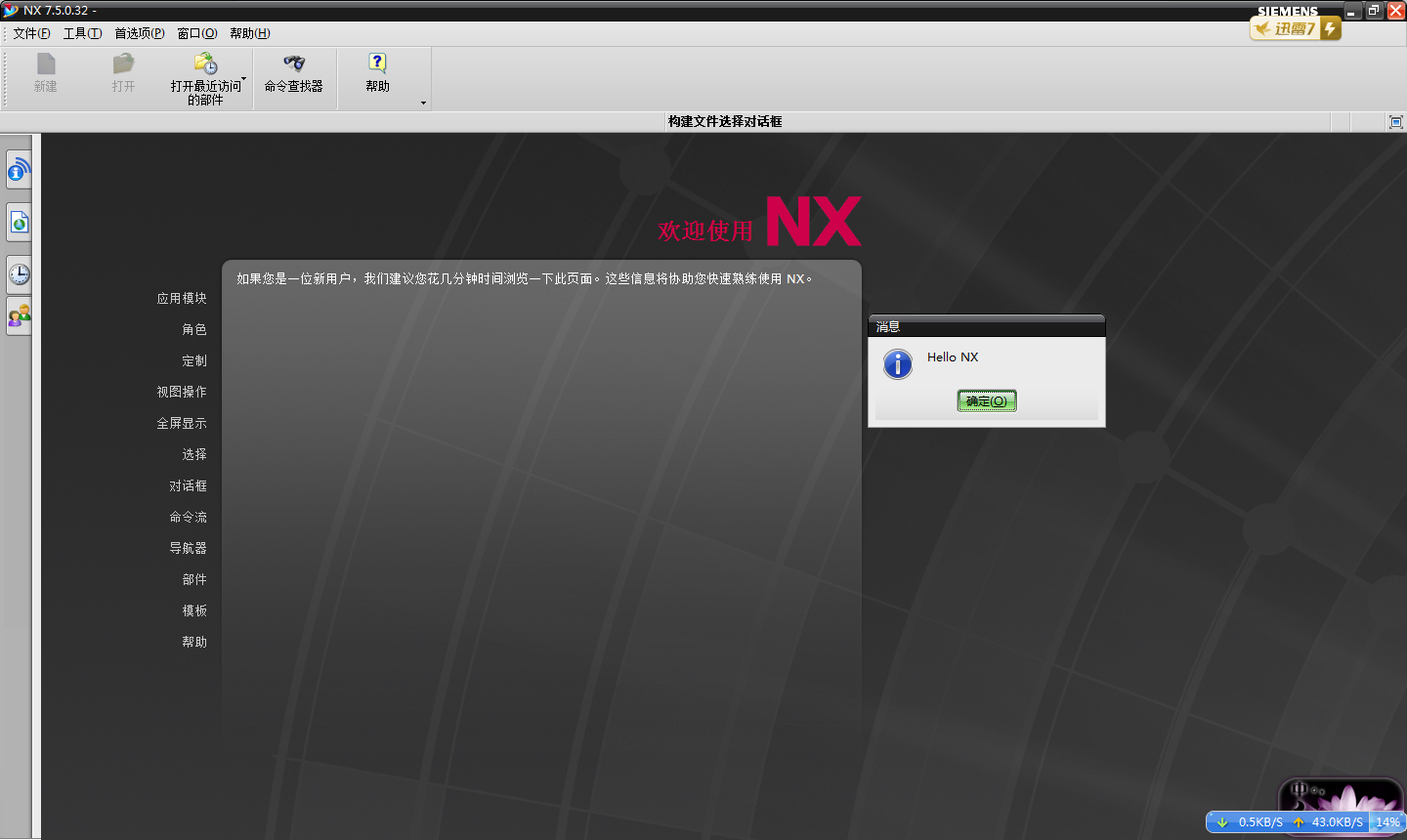
uc1601("Hello NX",1);//弹出消息窗口，显示“Hello NX”

UF\_terminate();//释放NX OPEN API的执行权限

}

}

**运行结果截图：**



**2、创建一个模型，并在信息窗口显示模型的tag值。**

代码：

#include <stdio.h>

#include <uf.h>

#include <uf\_modl.h>

#include < uf\_ui\_ugopen.h>

#include <uf\_part.h>

#include <uf\_ui.h>

static void do\_ugopen\_api(void)

{ typedef unsigned int tag\_t;

UF\_FEATURE\_SIGN sign = UF\_NULLSIGN;//无布尔运算

double cyl\_orig[3] = {0,0,0};//圆柱的圆心坐标

char \*cyl\_height="100";

char \*cyl\_diam="40";

double direction[3]={0,0,1};//延Z轴正方向

tag\_t obj=NULL\_TAG;

UF\_MODL\_create\_cyl1(sign, cyl\_orig,cyl\_height,cyl\_diam,direction, &obj);

UF\_UI\_open\_listing\_window(); //显示信息框:tag

/\*Returns the tag of the current display part. In a non-assembly part, this is the

same as the work part. If there currently isn't a displayed part, a NULL\_TAG is returned. \*/

obj=UF\_PART\_ask\_display\_part();

if(obj == NULL\_TAG) //没有部件的时候显示的信息提示框

{

uc1601 ( "当前没有任何文件可供操作！",1);

}

else

{

char s[10];

sprintf(s, "%d",obj); //sprinf()功能：把格式化的数据写入某个字符串

UF\_UI\_write\_listing\_window(s);

return;

}

}

extern void ufusr( char \*param, int \*retcode, int rlen )//提供入口点

{

if((UF\_initialize())!=0)

return;

do\_ugopen\_api ();

UF\_terminate();

return;

}

extern int ufusr\_ask\_unload(void)

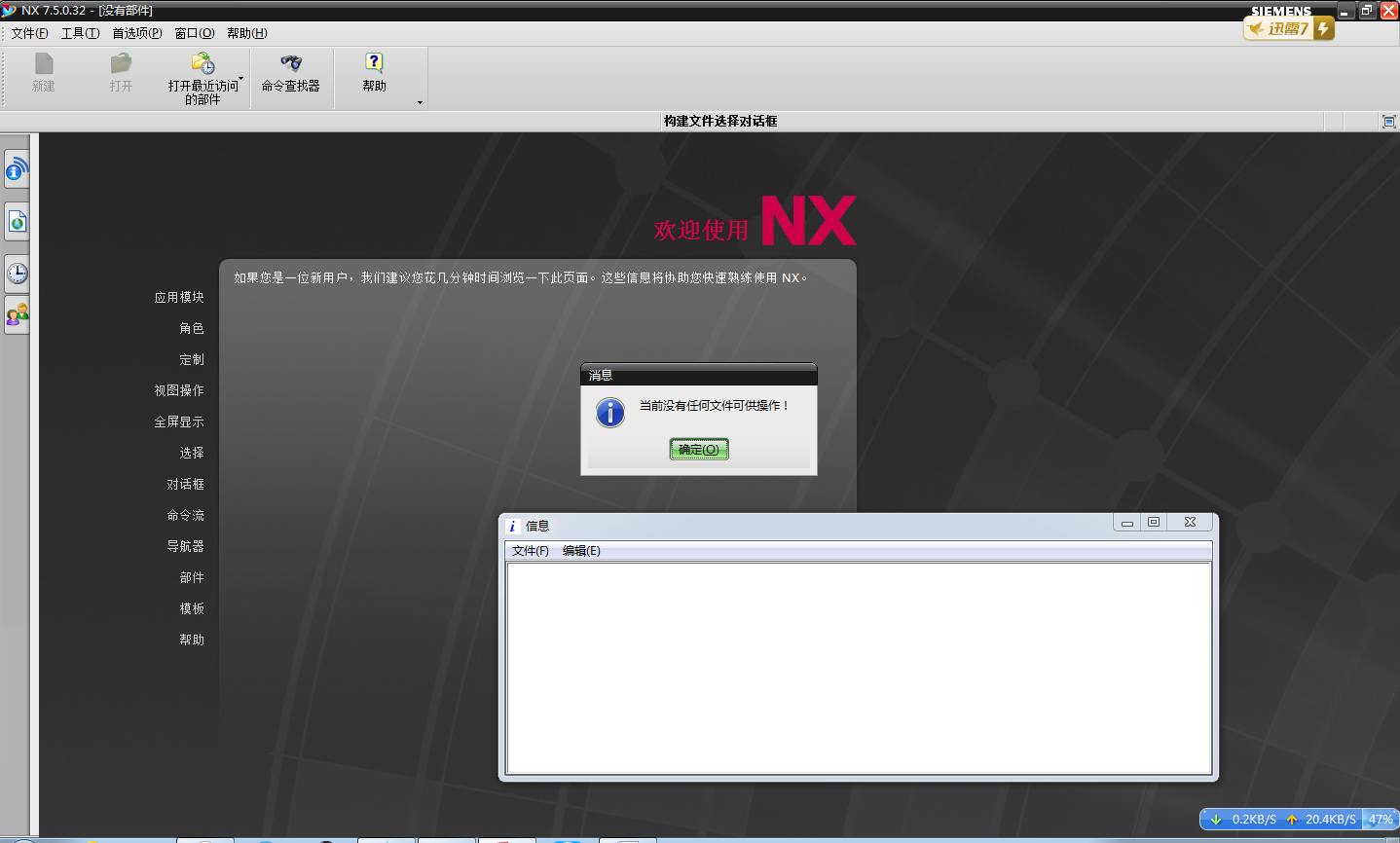
{

return(UF\_UNLOAD\_IMMEDIATELY);

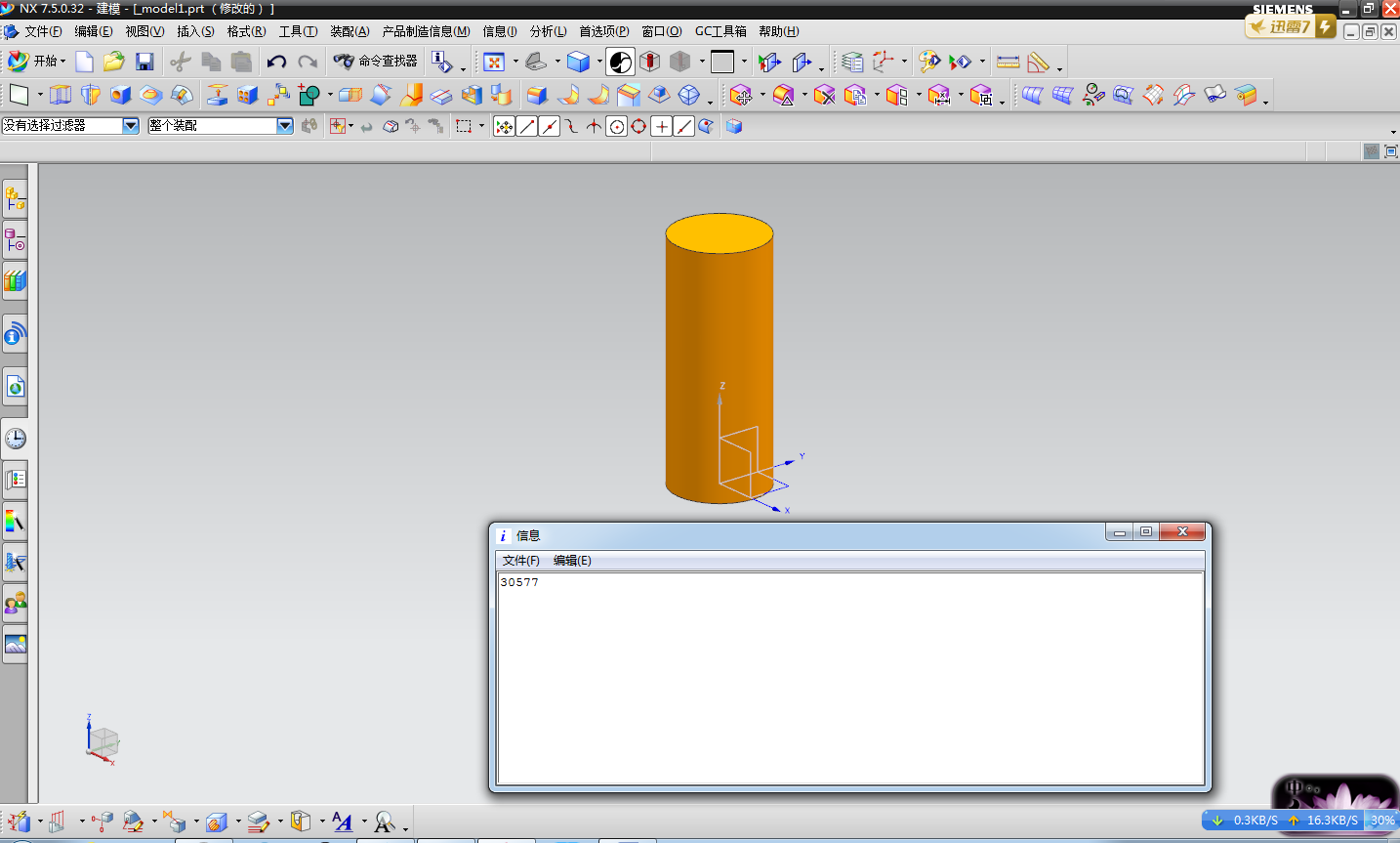
}

**运行结果截图：**

**（1）无部件时的输出**



**（2）有部件时的输出结果**



**3、open part 用户入口实例**

**创建open part user exit**

**在用户点击“open”时创建新部件，在其中创建sphere**

**详细说明user exit设置文件过程**

**代码如下：**

#include < uf\_assem.h>

#include <uf\_part.h>

#include <uf\_ui\_ugopen.h>

#include <stdio.h>

#include <uf.h>

#include <uf\_modl.h>

#include <uf\_ui.h>

#define UF\_CALL(X) (report( \_\_FILE\_\_, \_\_LINE\_\_, #X, (X)))

static int report( char \*file, int line, char \*call, int irc)

{

if (irc)

{

char messg[133];

printf("%s, line %d: %s\n", file, line, call);

(UF\_get\_fail\_message(irc, messg)) ?

printf(" returned a %d\n", irc) :

printf(" returned error %d: %s\n", irc, messg);

}

return(irc);

}

static void do\_ugopen\_api(void)

{

UF\_FEATURE\_SIGN sign = UF\_NULLSIGN;

tag\_t part=null\_tag;//如果没有赋值的话，直接就是"tag\_t part"，当你执行.dll文件时，就直接跑到建模环境了，没有信息提示框。当然你可以多申明几个tag\_t

char part\_name[13] = "F:\\long.prt";//设置part名字以及存储路径

double center [ 3 ]={0,0,0};//球心坐标

char \* diam="100";

int units =1;

uc1601("创建直径为100的球",1);

UF\_PART\_new (part\_name, units, &part);//在当前的会话框中创建新的part，并把其作为工作部件

//试比较下面两个函数

UF\_MODL\_create\_sphere(sign,part,center,diam, &part );

//UF\_MODL\_create\_sphere1(sign,center,diam, &part);

}

extern void ufusr(char \*param, int \*retcode, int paramLen)

{ if (!UF\_CALL(UF\_initialize()))

{

do\_ugopen\_api();

UF\_CALL(UF\_terminate());

}

}

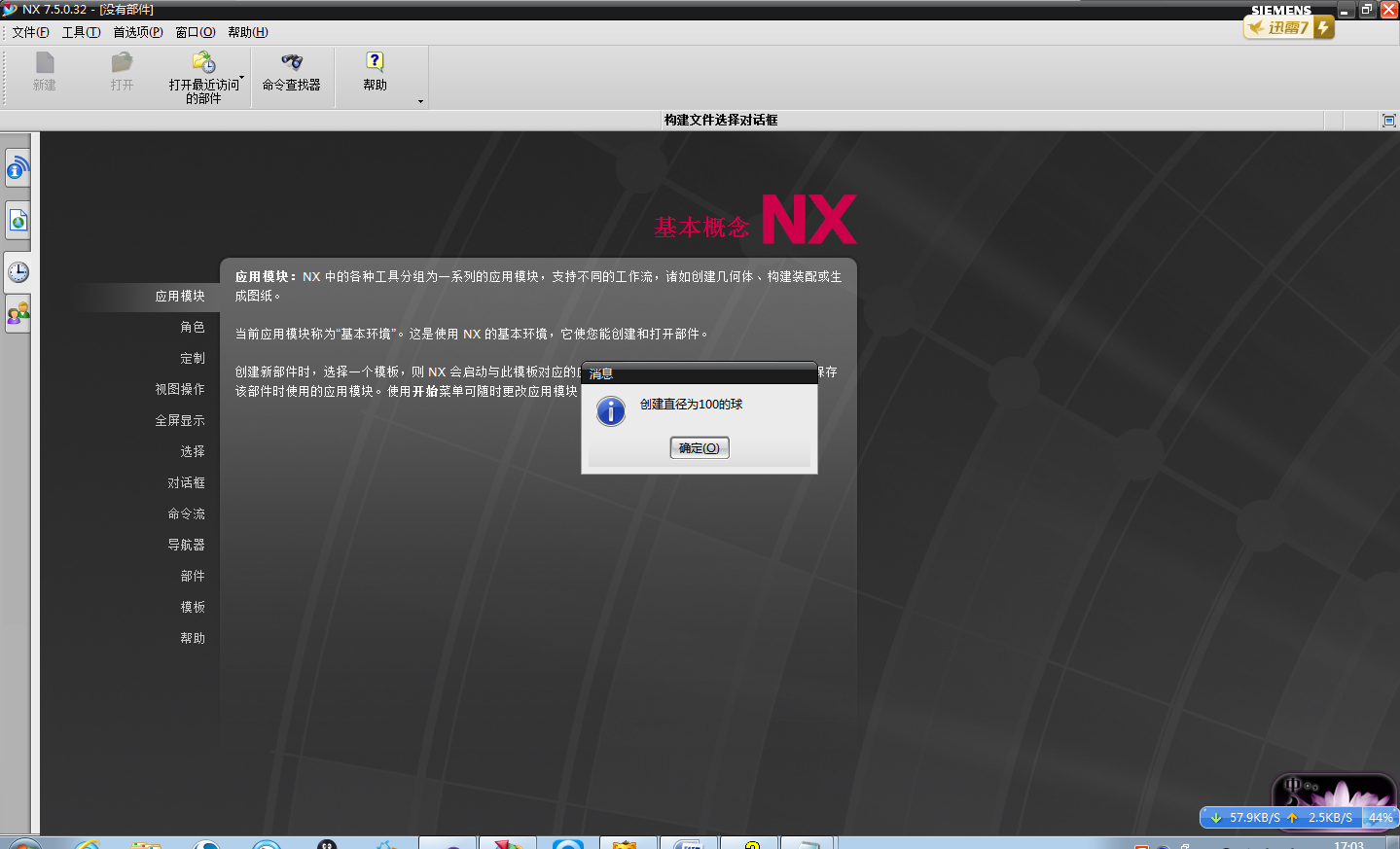
extern int ufusr\_ask\_unload(void)

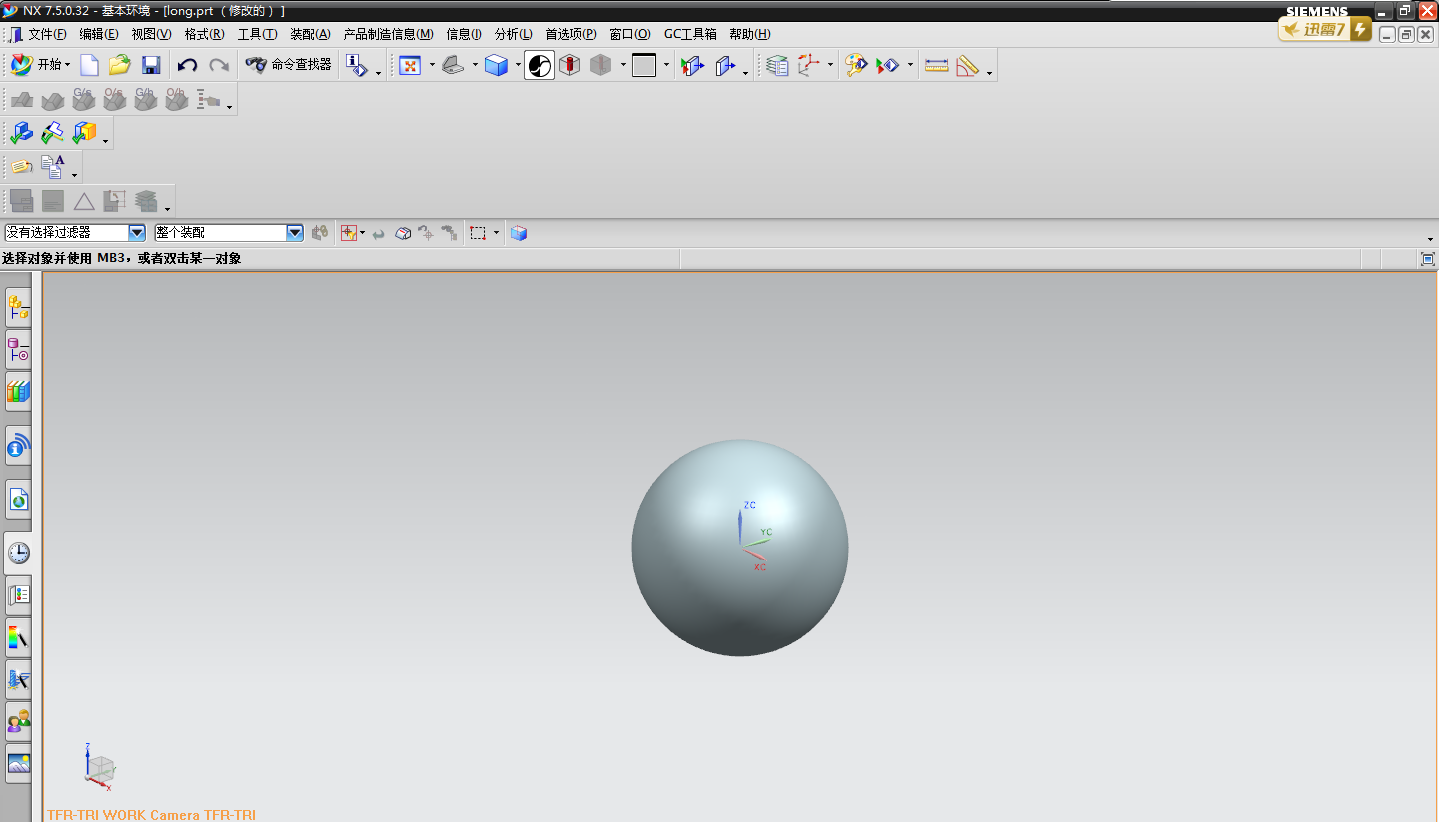
{

return (UF\_UNLOAD\_IMMEDIATELY);

}

**运行结果截图：**





**4、创建一个block，并指定某一特定面高亮显示。**

**代码如下图：**

#include <stdio.h>

#include <uf.h>

#include <uf\_modl.h>

#include<uf\_part.h>

#include<uf\_ui.h>

#include<uf\_obj.h>

#include<uf\_disp.h>

#define UF\_CALL(X) (report( \_\_FILE\_\_, \_\_LINE\_\_, #X, (X)))

static int report( char \*file, int line, char \*call, int irc)

{

if (irc)

{

char messg[133];

printf("%s, line %d: %s\n", file, line, call);

(UF\_get\_fail\_message(irc, messg)) ?

printf(" returned a %d\n", irc) :

printf(" returned error %d: %s\n", irc, messg);

}

return(irc);

}

static void do\_ugopen\_api(void)

{

UF\_FEATURE\_SIGN sign=UF\_NULLSIGN;

tag\_t obj=null\_tag;

double center [3]={0,0,0} ;

char \*edge\_len[3]={"10","10","10"};

tag\_t blk\_feat;

int units=2;//此行代码的赋值，经过调试，貌似只有1和2可以，它们的区别就是生成模型的默认大小不同，可以自己调调看。

char part\_name[13]="F:\\long.part";//此段代码有点小重要，注意一定要保证完整性，不完整的话，估计会出现好多问题，绕好多路。

uc1601("创建一个正方体，并指定下表面高亮显示",1);

UF\_PART\_new(part\_name,units,&obj);

//UF\_MODL\_create\_block(sign,obj,center,edge\_len,&blk\_feat);

UF\_MODL\_create\_block1(sign,center,edge\_len,&blk\_feat);

uf\_list\_p\_t face\_list;//链表

UF\_MODL\_create\_list(&face\_list);//创建链表

UF\_MODL\_ask\_feat\_faces(blk\_feat,&face\_list);//获取实体的个表面特征，并放入链表中

int count=0;

UF\_MODL\_ask\_list\_count(face\_list,&count);

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

{

UF\_MODL\_ask\_list\_item(face\_list,i,&obj);//从链表中取出对象

int type;

double point [20] ;

double dir [20] ;

double box [6];

double radius;

double rad\_data;

int norm\_dir;

UF\_MODL\_ask\_face\_data(obj,&type,point,dir,box,&radius,&rad\_data,&norm\_dir);//这个函数非常重要，可以说是本段代码中最核心的部分。

if(dir[0]==0&&dir[1]==0&&dir[2]==-1)//确定高亮显示的那个面的法向量

{

UF\_DISP\_set\_highlight(obj,1);

//UF\_OBJ\_set\_color(obj, 186);

break;

}

}

}

/\*ARGSUSED\*/

extern void ufusr(char \*param, int \*retcode, int paramLen)

{

if (!UF\_CALL(UF\_initialize()))

{

do\_ugopen\_api();

UF\_CALL(UF\_terminate());

}

}

extern int ufusr\_ask\_unload(void)

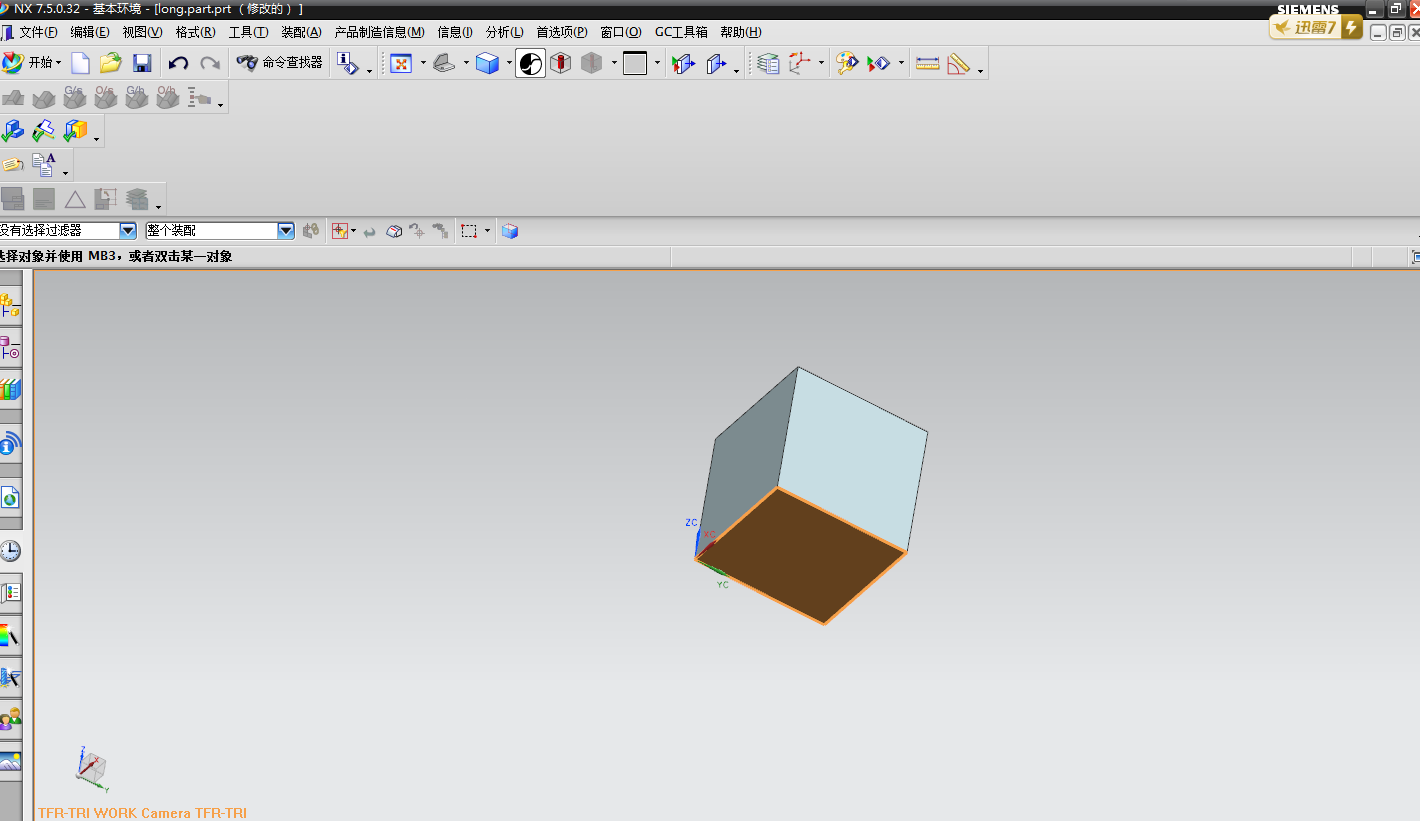
{

return (UF\_UNLOAD\_IMMEDIATELY);

}

**运行结果截图：**

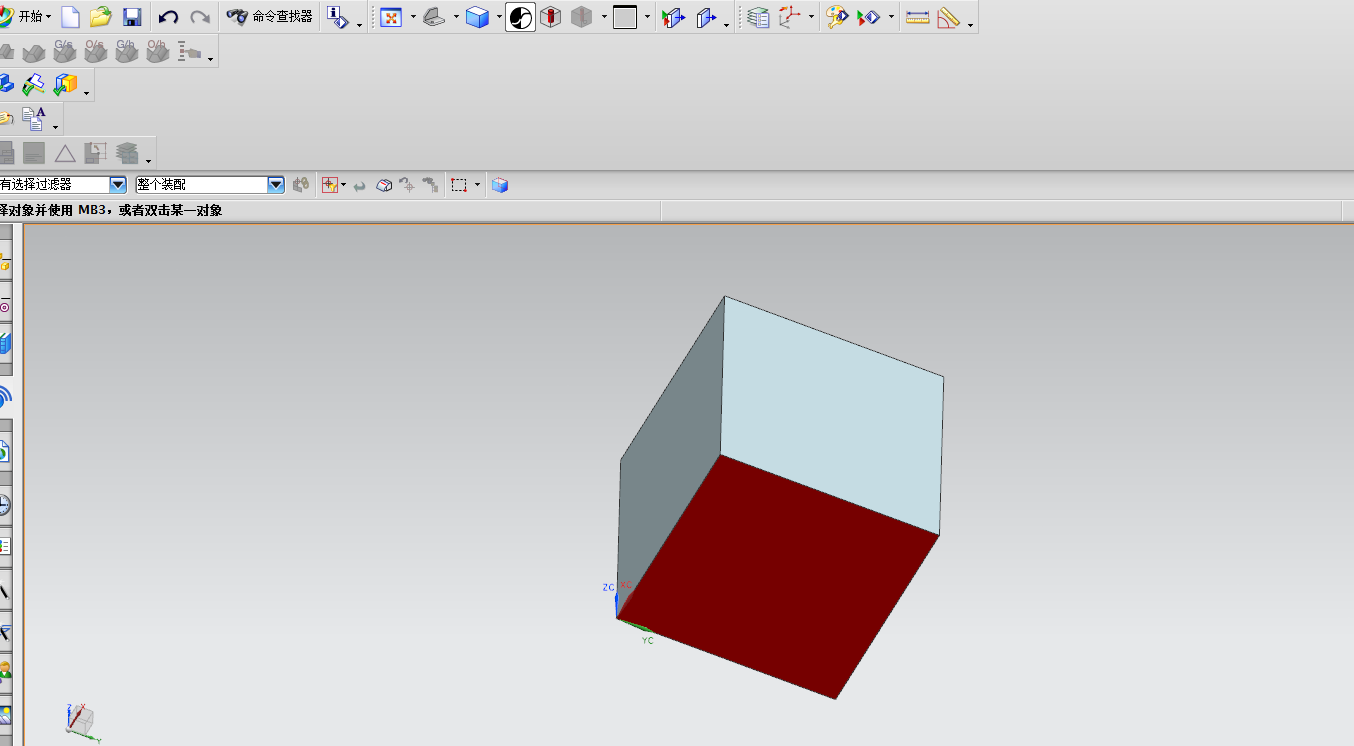




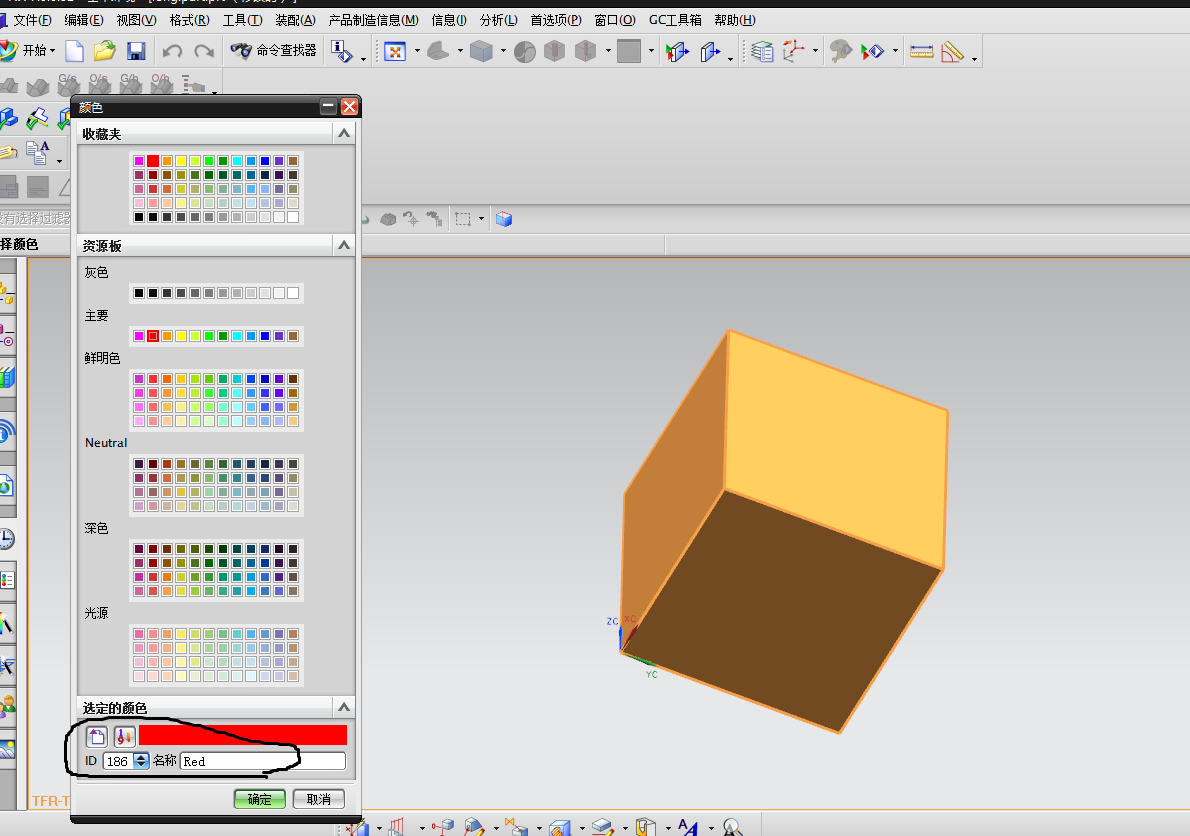
**注：和高亮显示还有一个类似的设置，就是给实体上色，所用到的函数是：**

**UF\_OBJ\_set\_color(obj, 186);**

**上面的这个函数是把对象显示成红色，如下图：**



**“186”是红色的代号。这些数字可以在UG交互模式下的对象显示里查到，如下图：**



**5、用遍历的形式输出当前part所有面的tag至listing window**

**代码如下：**

#include < uf\_assem.h>

#include <uf\_part.h>

#include <uf\_ui\_ugopen.h>

#include <stdio.h>

#include <uf.h>

#include <uf\_ui.h>

#include <uf\_modl.h>

#include <uf\_obj.h>

#define UF\_CALL(X) (report( \_\_FILE\_\_, \_\_LINE\_\_, #X, (X)))

static int report( char \*file, int line, char \*call, int irc)

{

if (irc)

{

char messg[133];

printf("%s, line %d: %s\n", file, line, call);

(UF\_get\_fail\_message(irc, messg)) ?

printf(" returned a %d\n", irc) :

printf("returned error %d: %s\n", irc, messg);

}

return(irc);

}

static void do\_ugopen\_api(void)

{

UF\_FEATURE\_SIGN sign = UF\_NULLSIGN;

double center [ 3 ] = {0,0,0};

char \* edge\_len [ 3 ]={"100","100","100"};

tag\_t blk\_feat =null\_tag;

char \* part\_name="F:\\long";

int units=1;

tag\_t part =NULL\_TAG;

uc1601("创建一个block，并显示每个面的tag",1);

UF\_PART\_new(part\_name,units,&part);//在当前会话窗口中创建一个part，并使其成为工作部件

UF\_CALL(UF\_MODL\_create\_block1(sign,center,edge\_len, &blk\_feat));

if(blk\_feat == NULL\_TAG)

{

uc1601 ( "当前没有任何文件可供操作！",1);

}

else

{

int type=70;

int subtype=2;

tag\_t obj=null\_tag;

tag\_t display\_part=UF\_PART\_ask\_display\_part();

UF\_OBJ\_cycle\_objs\_in\_part(display\_part,70,&obj);

while(obj!=null\_tag)

{

UF\_OBJ\_ask\_type\_and\_subtype(obj,&type,&subtype);//返回一个被标记的对象的对象类型和子类型

UF\_OBJ\_cycle\_objs\_in\_part (display\_part,UF\_solid\_type,&obj);//查询遍历对象

if(subtype== UF\_solid\_face\_subtype)

{

char s[50];

sprintf(s, "%d", obj);

UF\_CALL(UF\_UI\_open\_listing\_window());

UF\_CALL(UF\_UI\_write\_listing\_window(s));

UF\_CALL(UF\_UI\_write\_listing\_window("\n"));

}

}

}

}

void ufusr(char \*param, int \*retcode, int paramLen)

{

if (!UF\_CALL(UF\_initialize()))

{

do\_ugopen\_api();

UF\_CALL(UF\_terminate());

}

}

int ufusr\_ask\_unload(void)

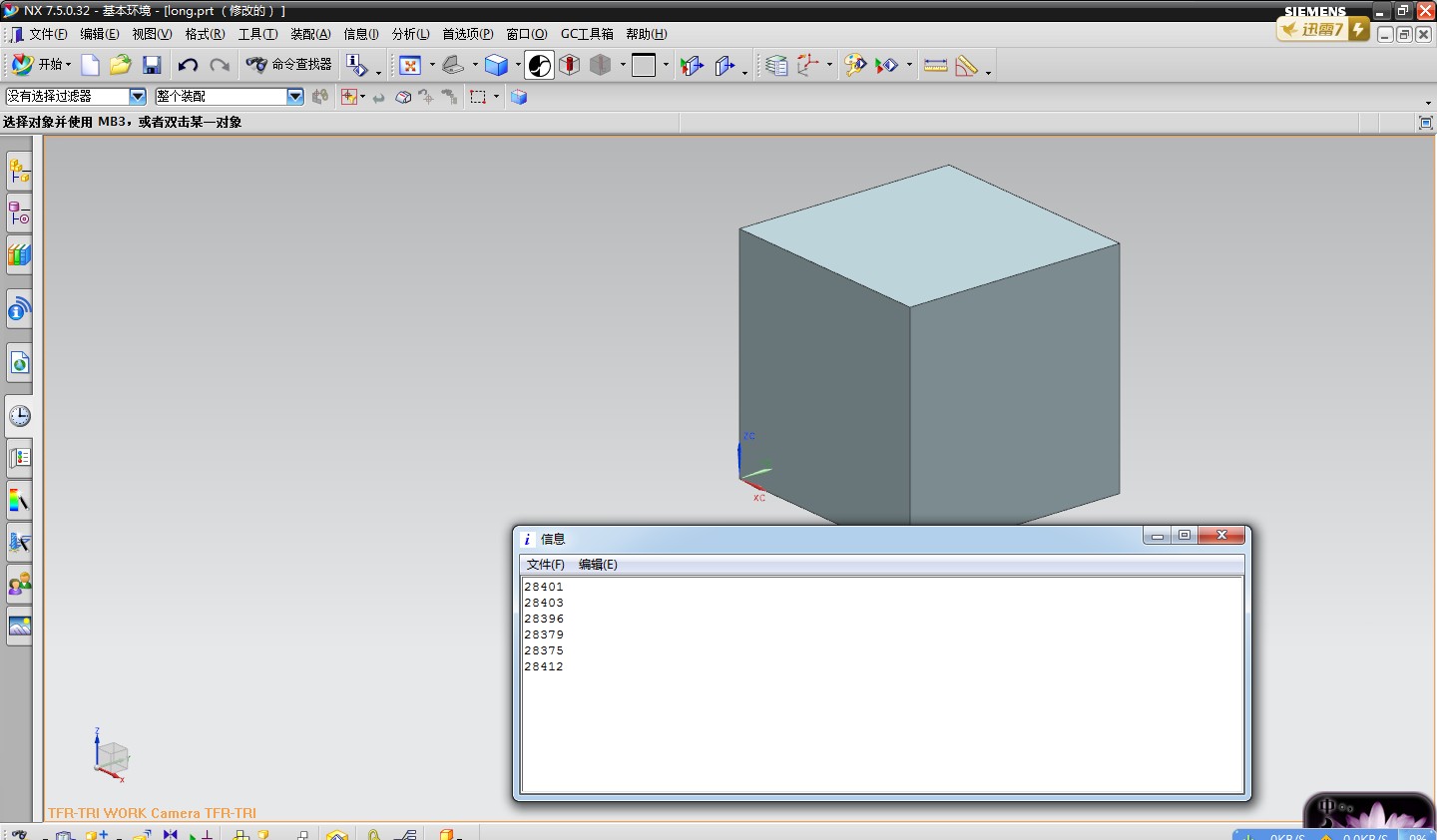
{

return (UF\_UNLOAD\_IMMEDIATELY);

}

**运行结果截屏：**

****

****

**6、遍历当前part中所有features，输出其类型至listing window，UF\_MODL\_ask\_feat\_type()**

**代码如下：**

#include <uf.h>

#include <stdio.h>

#include <uf\_obj.h>

#include <uf\_part.h>

#include <uf\_ui.h>

#include <uf\_modl.h>

#define UF\_CALL(X) (report( \_\_FILE\_\_, \_\_LINE\_\_, #X, (X)))

static int report( char \*file, int line, char \*call, int irc)

{

if (irc)

{

char messg[133];

printf("%s, line %d: %s\n", file, line, call);

(UF\_get\_fail\_message(irc, messg)) ?

printf(" returned a %d\n", irc) :

printf(" returned error %d: %s\n", irc, messg);

}

return(irc);

}

static void do\_ugopen\_api(void)

{

char \* part\_name="F:\\NX";

UF\_FEATURE\_SIGN sign = UF\_NULLSIGN;

tag\_t part=null\_tag;

double center [ 3 ]={0,0,0};

char \* diam="100";

int units =1;

uc1601("创建直径为100的球,并显示其实体特征。",1);

UF\_PART\_new (part\_name, units, &part);//在当前的会话框中创建新的part，并把其作为工作部件

UF\_MODL\_create\_sphere1(sign,center,diam, &part);

int count = 0;

char\* solid\_feature;

tag\_t found = NULL\_TAG;

tag\_t display\_part = UF\_PART\_ask\_display\_part();

UF\_UI\_open\_listing\_window();

UF\_OBJ\_cycle\_objs\_in\_part(display\_part,UF\_feature\_type,&found);

while( found != NULL\_TAG )

{

count++;

UF\_CALL(UF\_MODL\_ask\_feat\_type(found,&solid\_feature));//获取输入的特征的类型

UF\_CALL(UF\_UI\_write\_listing\_window("feature："));

UF\_CALL(UF\_UI\_write\_listing\_window(solid\_feature));

UF\_CALL(UF\_UI\_write\_listing\_window("\n"));

UF\_CALL(UF\_OBJ\_cycle\_objs\_in\_part(display\_part,UF\_feature\_type,&found));

}

}

extern void ufusr(char \*param, int \*retcode, int paramLen)

{

if (!UF\_CALL(UF\_initialize()))

{

do\_ugopen\_api();

UF\_CALL(UF\_terminate());

}

}

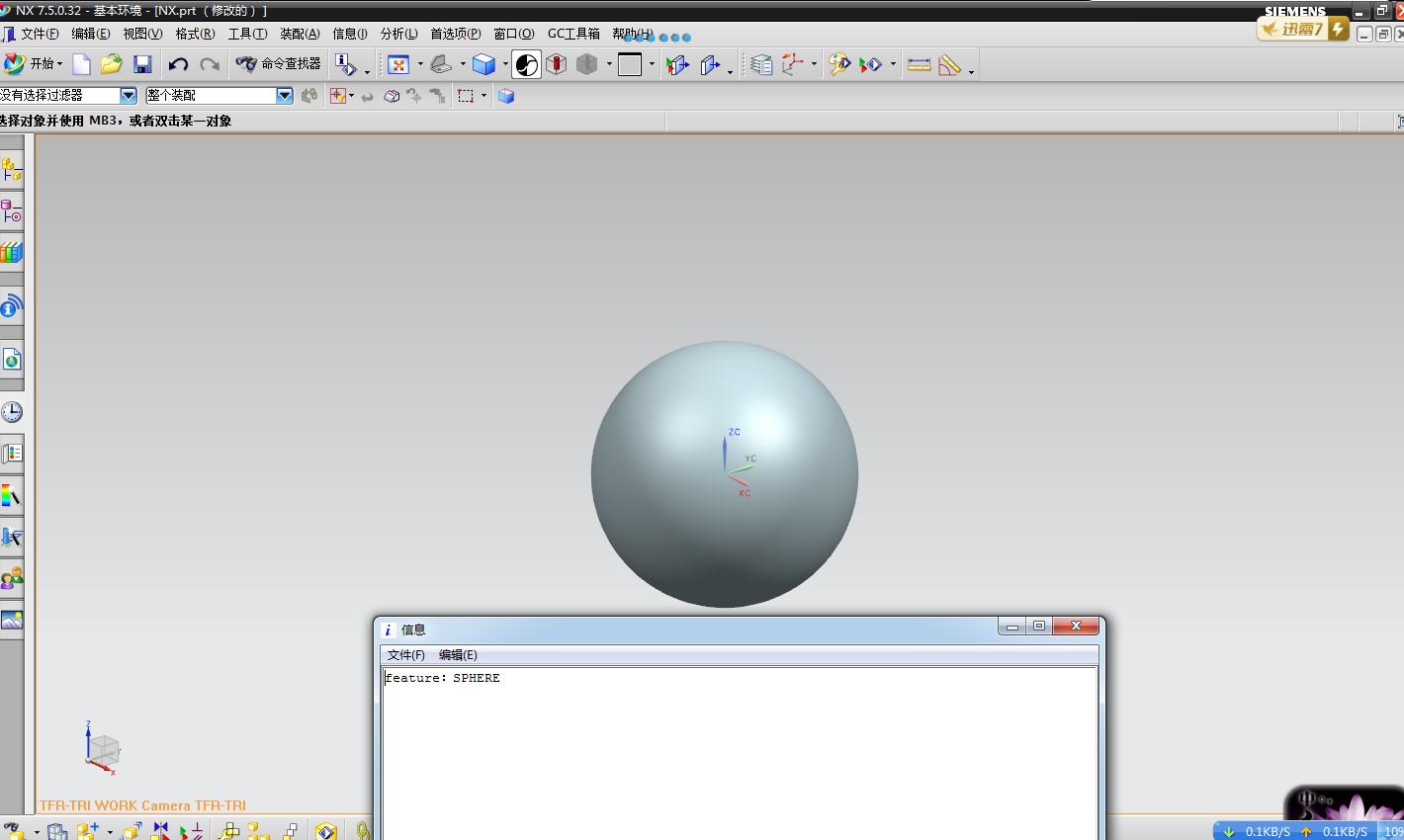
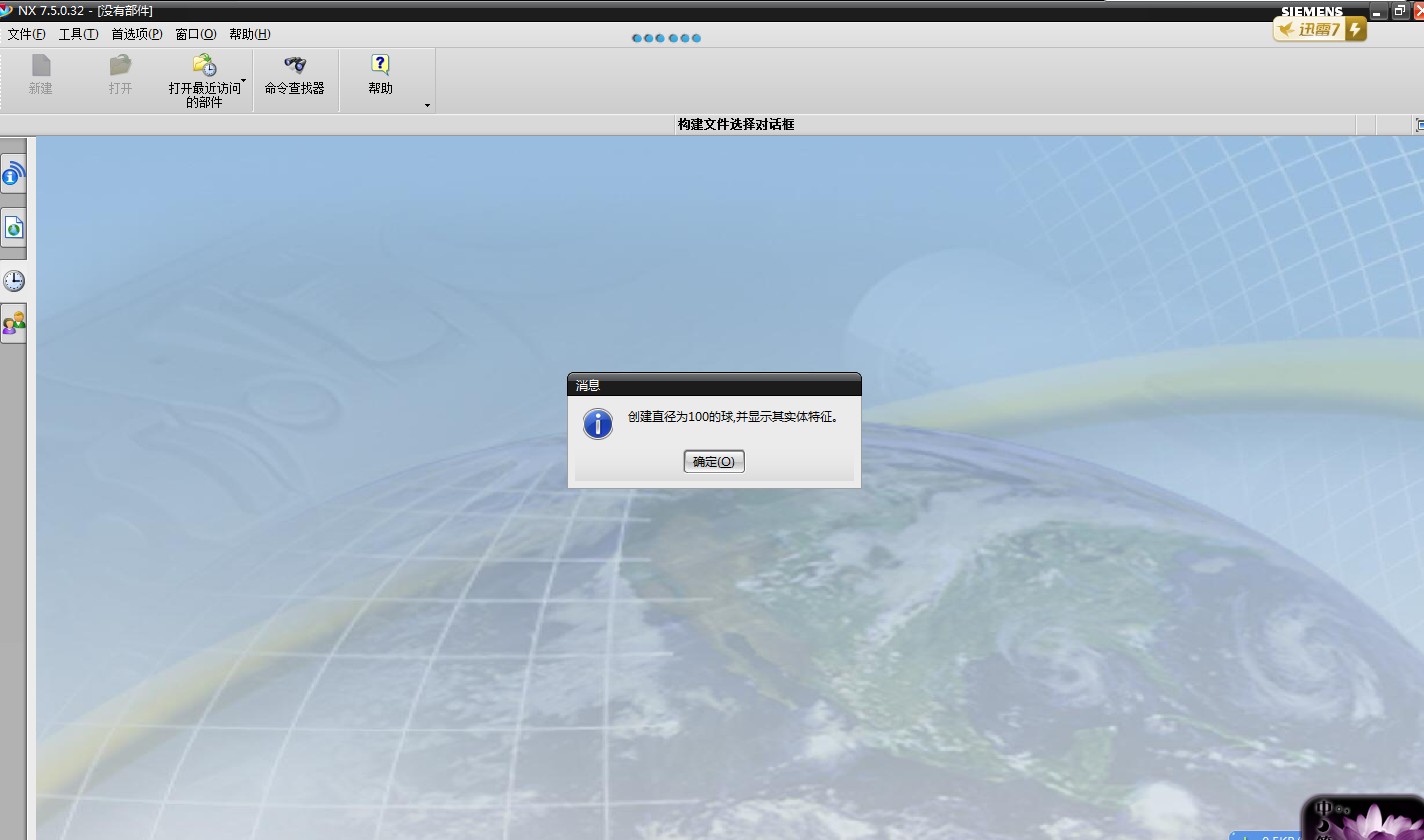
extern int ufusr\_ask\_unload(void)

{

return(UF\_UNLOAD\_IMMEDIATELY);

}

**运行结果截屏：**

****

**6、创建人机交互界面新建圆柱体，并可以随意选择原点和矢量方向。**

**代码如下：**

#include <uf.h>

#include <stdio.h>

#include <uf\_defs.h>

#include <uf\_exit.h>

#include <uf\_styler.h>

#include <uf\_mb.h>

#include <uf\_disp.h>

#include <uf\_part.h>

#include <uf\_ui.h>

#include <uf\_modl.h>

#include <uf\_obj.h>

#include "cylinder.h"

#define CHANGE\_CB\_COUNT ( 3 + 1 ) /\* Add 1 for the terminator \*/

tag\_t blk\_tag= null\_tag;

double blk\_orig[] = {0,0,0};

double direction [ 3 ];

char \* part\_name="F:\\long";

int units=1;

static UF\_STYLER\_callback\_info\_t CHANGE\_cbs[CHANGE\_CB\_COUNT] =

{

{UF\_STYLER\_DIALOG\_INDEX, UF\_STYLER\_APPLY\_CB , 0, CHANGE\_apply\_cb},

{CHANGE\_ACTION\_2 , UF\_STYLER\_ACTIVATE\_CB , 1, CHANGE\_action\_2\_act\_cb},

{CHANGE\_ACTION\_3 , UF\_STYLER\_ACTIVATE\_CB , 1, CHANGE\_action\_3\_act\_cb},

{UF\_STYLER\_NULL\_OBJECT, UF\_STYLER\_NO\_CB, 0, 0 }

};

static UF\_MB\_styler\_actions\_t actions[] = {

{ "cylinder.dlg", NULL, CHANGE\_cbs, UF\_MB\_STYLER\_IS\_NOT\_TOP },

{ NULL, NULL, NULL, 0 } /\* This is a NULL terminated list \*/

};

static int init\_proc(UF\_UI\_selection\_p\_t select,void \*user\_data);

extern void ufusr (char \*param, int \*retcode, int rlen)

{

uc1601("创建一个圆柱,并指明原点和矢量",1);

UF\_PART\_new(part\_name,units,&blk\_tag);

int response = 0;

int error\_code = 0;

if ( ( UF\_initialize() ) != 0 )

return;

if ( ( error\_code = UF\_STYLER\_create\_dialog ( "cylinder.dlg",

CHANGE\_cbs, /\* Callbacks from dialog \*/

CHANGE\_CB\_COUNT, /\* number of callbacks\*/

NULL, /\* This is your client data \*/

&response ) ) != 0 )

{

char fail\_message[133];

/\* Get the user function fail message based on the fail code.\*/

UF\_get\_fail\_message(error\_code, fail\_message);

UF\_UI\_set\_status (fail\_message);

printf ( "%s\n", fail\_message );

}

UF\_terminate();

return;

}

extern int ufusr\_ask\_unload (void)

{

return ( UF\_UNLOAD\_IMMEDIATELY );

}

extern void ufusr\_cleanup (void)

{

return;

}

int CHANGE\_apply\_cb ( int dialog\_id,void \* client\_data,UF\_STYLER\_item\_value\_type\_p\_t callback\_data)

{

if ( UF\_initialize() != 0)

return ( UF\_UI\_CB\_CONTINUE\_DIALOG );

//直径的相关参数

UF\_STYLER\_item\_value\_type\_t data;

data.item\_id = "REAL\_0";

data.item\_attr = UF\_STYLER\_VALUE;

data.indicator = UF\_STYLER\_REAL\_VALUE;

UF\_STYLER\_ask\_value(dialog\_id,&data);//用来查询对话框控件的属性值。

char diameter[133];

sprintf(diameter,"%f",data.value.real);

//高度的相关参数

data.item\_id = "REAL\_1";

data.item\_attr = UF\_STYLER\_VALUE;

data.indicator = UF\_STYLER\_REAL\_VALUE;

UF\_STYLER\_ask\_value(dialog\_id,&data);

char height[133];

sprintf(height,"%f",data.value.real);

UF\_FEATURE\_SIGN sign = UF\_NULLSIGN;

//tag\_t blk\_tag= null\_tag;

//UF\_PART\_new(part\_name,units,&blk\_tag);

UF\_MODL\_create\_cyl1(sign,blk\_orig,diameter,height,direction , &blk\_tag);

UF\_STYLER\_free\_value(&data);//释放字符串或数组申请的内存

UF\_STYLER\_ask\_value(dialog\_id,&data);

UF\_MODL\_update();

UF\_STYLER\_free\_value(&data);

UF\_terminate ();

return (UF\_UI\_CB\_CONTINUE\_DIALOG);

}

//选择圆柱中心点,选用点构造器UF\_UI\_point\_construct（）

int CHANGE\_action\_2\_act\_cb ( int dialog\_id,void \* client\_data,UF\_STYLER\_item\_value\_type\_p\_t callback\_data)

{

if ( UF\_initialize() != 0)

return ( UF\_UI\_CB\_CONTINUE\_DIALOG );

char \*cue = "点构造器";

UF\_UI\_POINT\_base\_method\_t method = UF\_UI\_POINT\_INFERRED;

tag\_t point\_tag = NULL\_TAG;

//double sBasePoint[3];;

int response;

int error = UF\_UI\_point\_construct(cue,

&method,

&point\_tag,

blk\_orig,

&response);

UF\_terminate ();

return (UF\_UI\_CB\_CONTINUE\_DIALOG);

}

//选择圆柱方向（拾取向量对话框）

int CHANGE\_action\_3\_act\_cb ( int dialog\_id,void \* client\_data,UF\_STYLER\_item\_value\_type\_p\_t callback\_data)

{

if ( UF\_initialize() != 0)

return ( UF\_UI\_CB\_CONTINUE\_DIALOG );

double vec[3], pnt[3];

int mode = UF\_UI\_ZC\_AXIS;//默认矢量方向

int disp\_flag = UF\_UI\_DISP\_NO\_VECTOR;//显示临时坐标，是比较UF\_UI\_DISP\_NO\_VECTOR

int response = 0;

int ifail;

UF\_initialize();

ifail = UF\_UI\_specify\_vector( "选择一个矢量",

&mode,

disp\_flag,

vec,

pnt,

&response );

if ( ifail != 0 || response != UF\_UI\_OK )

printf( "No vector selected \n" );

else

printf( "Vect base (%f, %f, %f), direction (%f, %f, %f) \n",

pnt[0], pnt[1], pnt[2], vec[0], vec[1], vec[2] );

fflush( stdout );

UF\_terminate ();

//double direction[3];

for(int i=0;i<3;i++)

{

direction[i]=vec[i]-pnt[i];

}

/\* Callback acknowledged, do not terminate dialog \*/

return (UF\_UI\_CB\_CONTINUE\_DIALOG);

}

static int init\_proc(UF\_UI\_selection\_p\_t select,void\* user\_data)

{

//指明了mask\_triple数组的长度

int num\_triples = 1;

UF\_UI\_mask\_t mask\_triples[] = {UF\_solid\_type, 0, UF\_UI\_SEL\_FEATURE\_SOLID\_BODY};

if((UF\_UI\_set\_sel\_mask(select, UF\_UI\_SEL\_MASK\_CLEAR\_AND\_ENABLE\_SPECIFIC,

num\_triples,

mask\_triples)) == 0)

{

return (UF\_UI\_SEL\_SUCCESS);

}

else

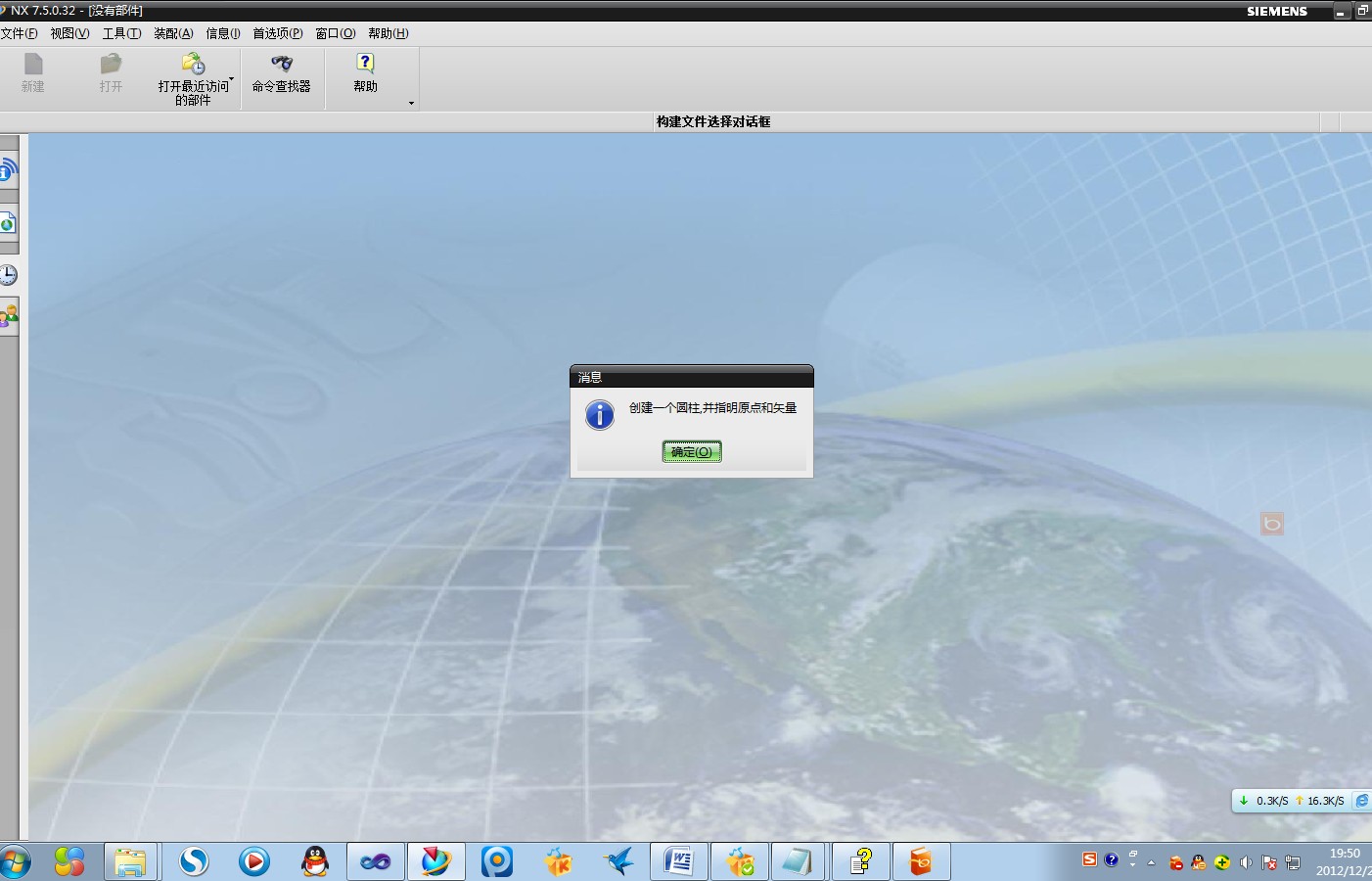
{

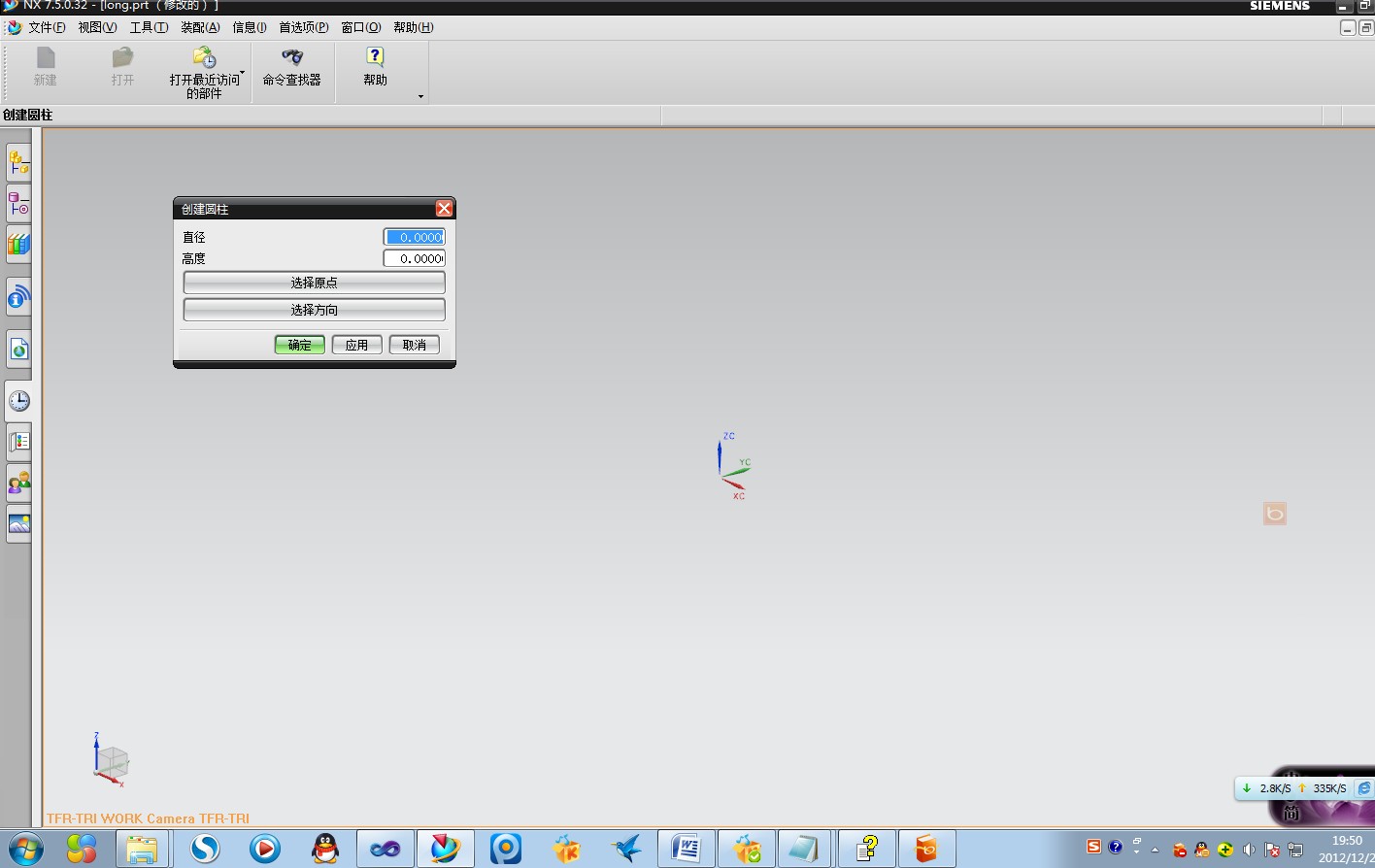
return (UF\_UI\_SEL\_FAILURE);

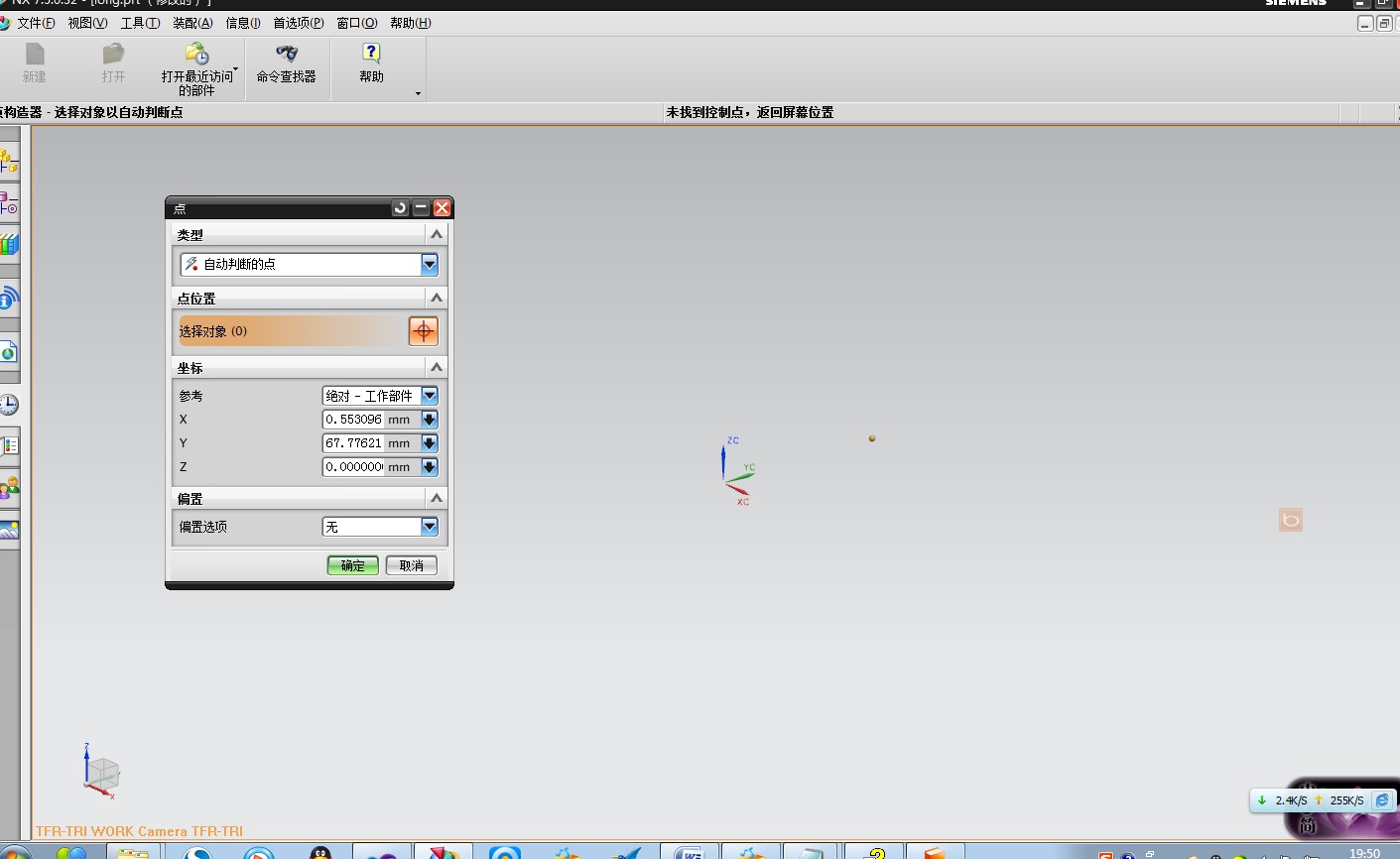
}

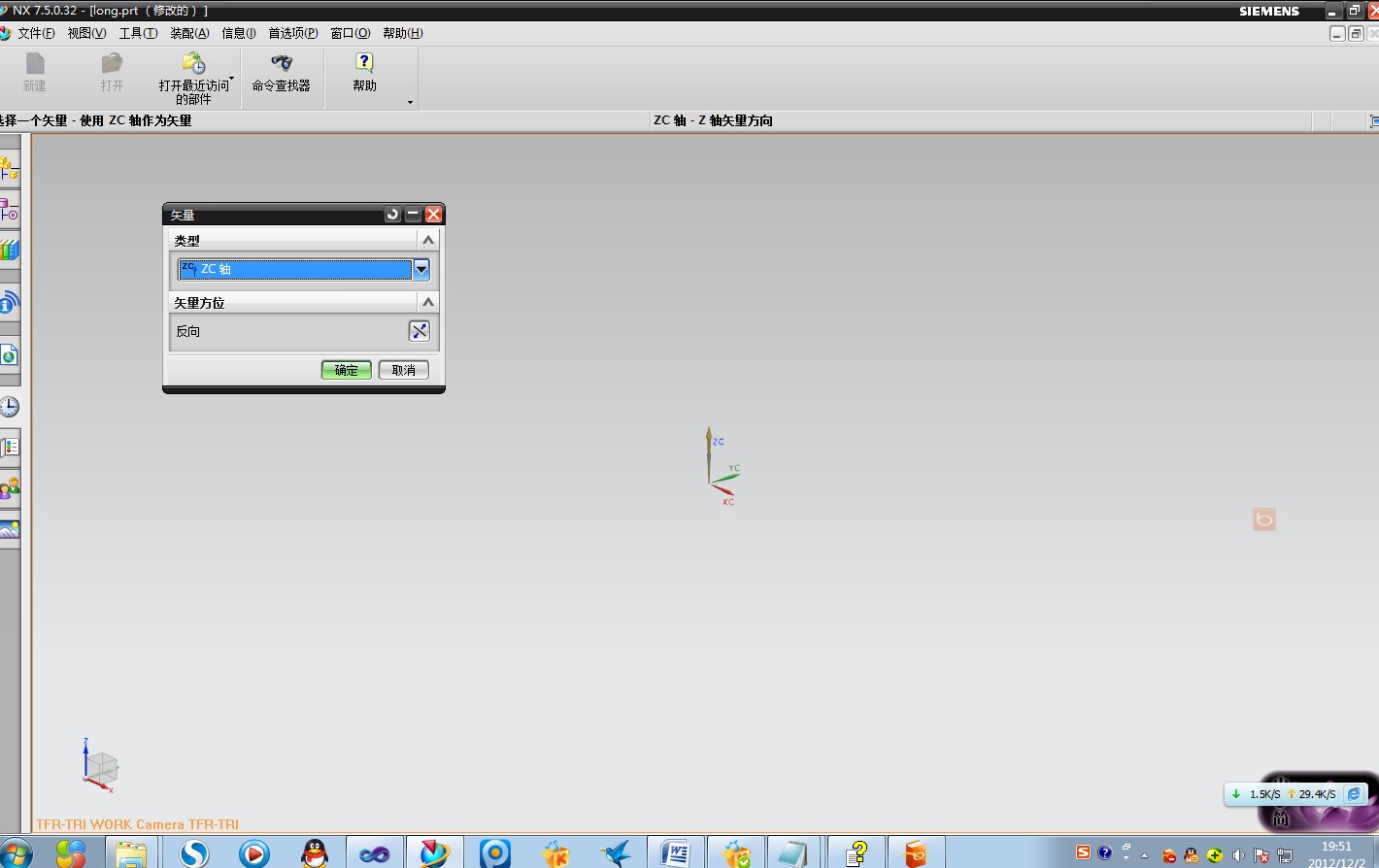
}

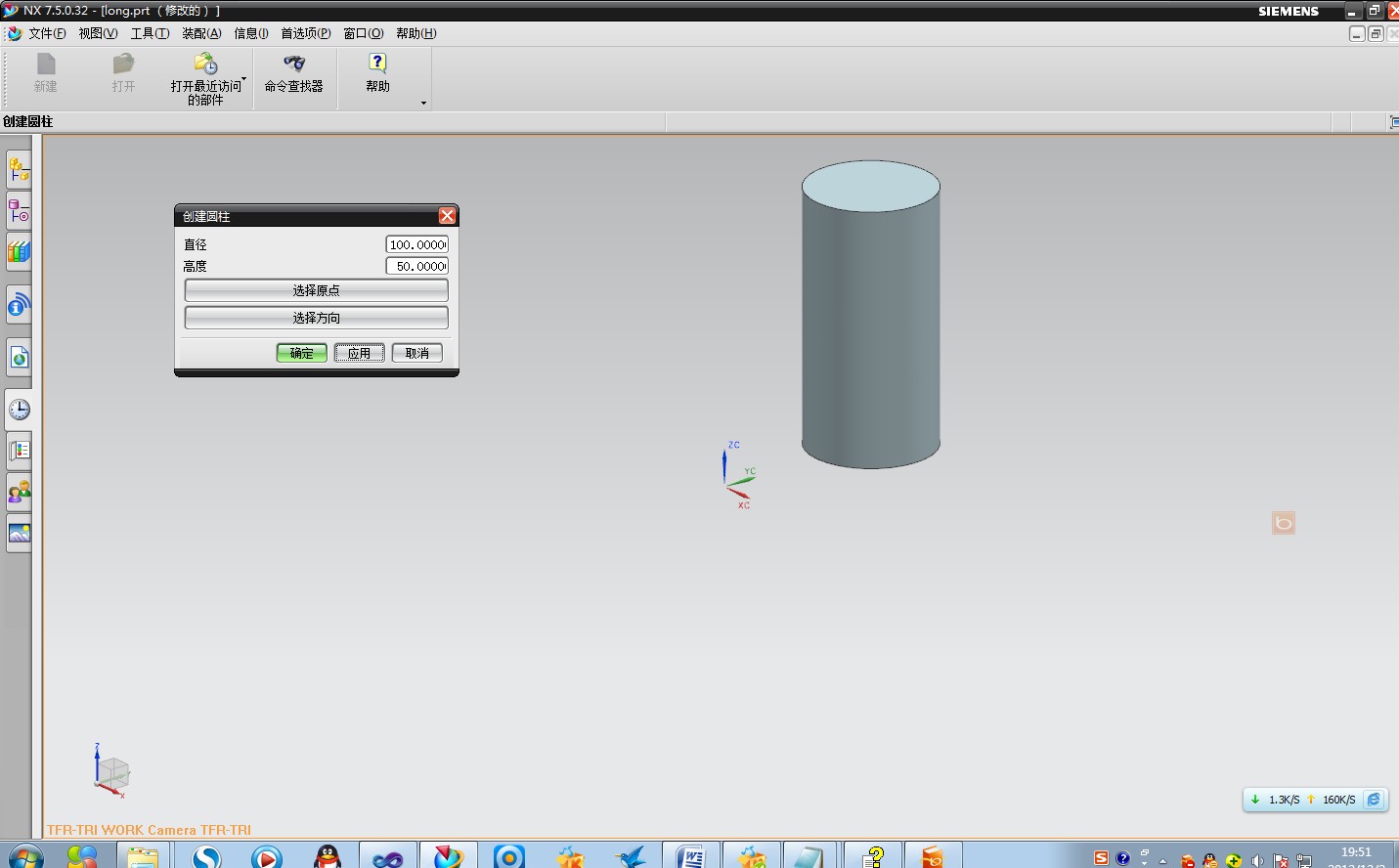
**运行结果截屏**

****

****

****

****

****

**7、（1）新建一个对象，选择对象的面，返回这个面的tag值。**

**代码如下：**

#include <stdio.h>

#include <uf.h>

#include <uf\_ui.h>

#include <uf\_ui\_types.h>

#include <uf\_disp.h>

static int sel\_init\_proc(UF\_UI\_selection\_p\_t select,void \*user\_data);

#define UF\_CALL(X) (report( \_\_FILE\_\_, \_\_LINE\_\_, #X, (X)))

static int report( char \*file, int line, char \*call, int irc)

{

if (irc)

{

char messg[133];

printf("%s, line %d: %s\n", file, line, call);

(UF\_get\_fail\_message(irc, messg)) ?

printf(" returned a %d\n", irc) :

printf(" returned error %d: %s\n", irc, messg);

}

return(irc);

}

static void do\_ugopen\_api(void)

{

//类选择对话框相关变量

char cue[] = "Select Objects";

char title[] = "Select faces";

int scope=UF\_UI\_SEL\_SCOPE\_NO\_CHANGE;

int response, count, i;

tag\_p\_t objects;

int error\_code = 0;

char\* str = (char\*)UF\_allocate\_memory(sizeof(unsigned int)+1,&error\_code);

UF\_UI\_open\_listing\_window();

if((UF\_CALL(UF\_UI\_select\_with\_class\_dialog(

cue, title, scope,sel\_init\_proc, NULL, &response, &count, &objects))) == 0)

{

printf("object count = %d\n",count);

if (response == UF\_UI\_OK && count > 0)

{

for (i=0; i<count; i++)

{

sprintf(str,"%d\n",objects[i]);

UF\_UI\_write\_listing\_window(str);

UF\_DISP\_set\_highlight(objects[i], 0);//把选中的对象高亮显示

}

UF\_free(objects);

}

}

}

/\* selection initialization procedure \*/

static int sel\_init\_proc(UF\_UI\_selection\_p\_t select,void\* user\_data)

{

int num\_triples=1;

UF\_UI\_mask\_t mask\_triples[] = {UF\_solid\_type, 0, UF\_UI\_SEL\_FEATURE\_PLANAR\_FACE};

/\* enable only lines and edges \*/

if((UF\_CALL(UF\_UI\_set\_sel\_mask(select,UF\_UI\_SEL\_MASK\_CLEAR\_AND\_ENABLE\_SPECIFIC,

num\_triples, mask\_triples))) == 0)

{

return (UF\_UI\_SEL\_SUCCESS);

}

else

{

return (UF\_UI\_SEL\_FAILURE);

}

}

/\*ARGSUSED\*/

void ufusr(char \*param, int \*retcode, int param\_len)

{

if (!UF\_CALL(UF\_initialize()))

{

do\_ugopen\_api();

UF\_CALL(UF\_terminate());

}

}

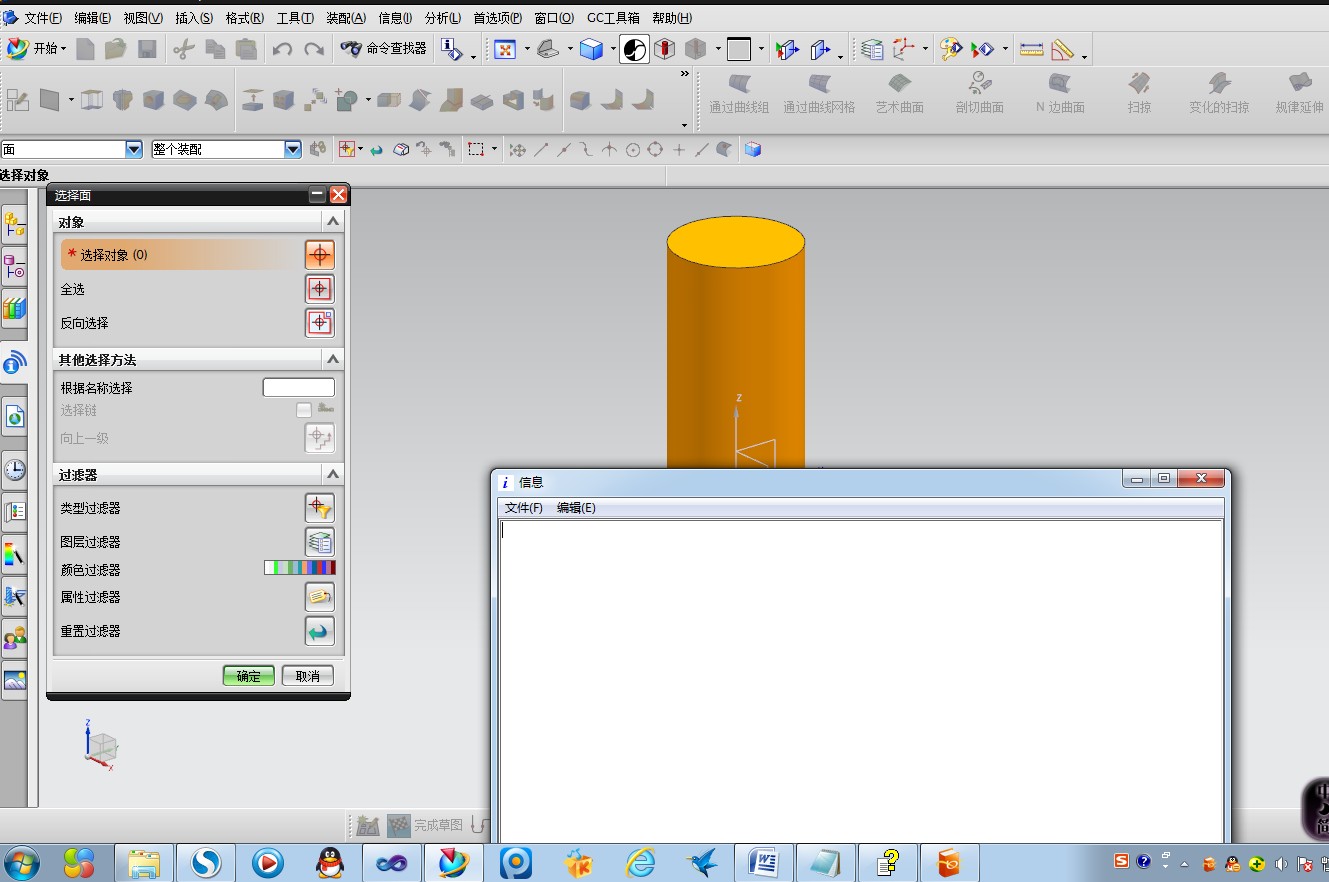
int ufusr\_ask\_unload(void)

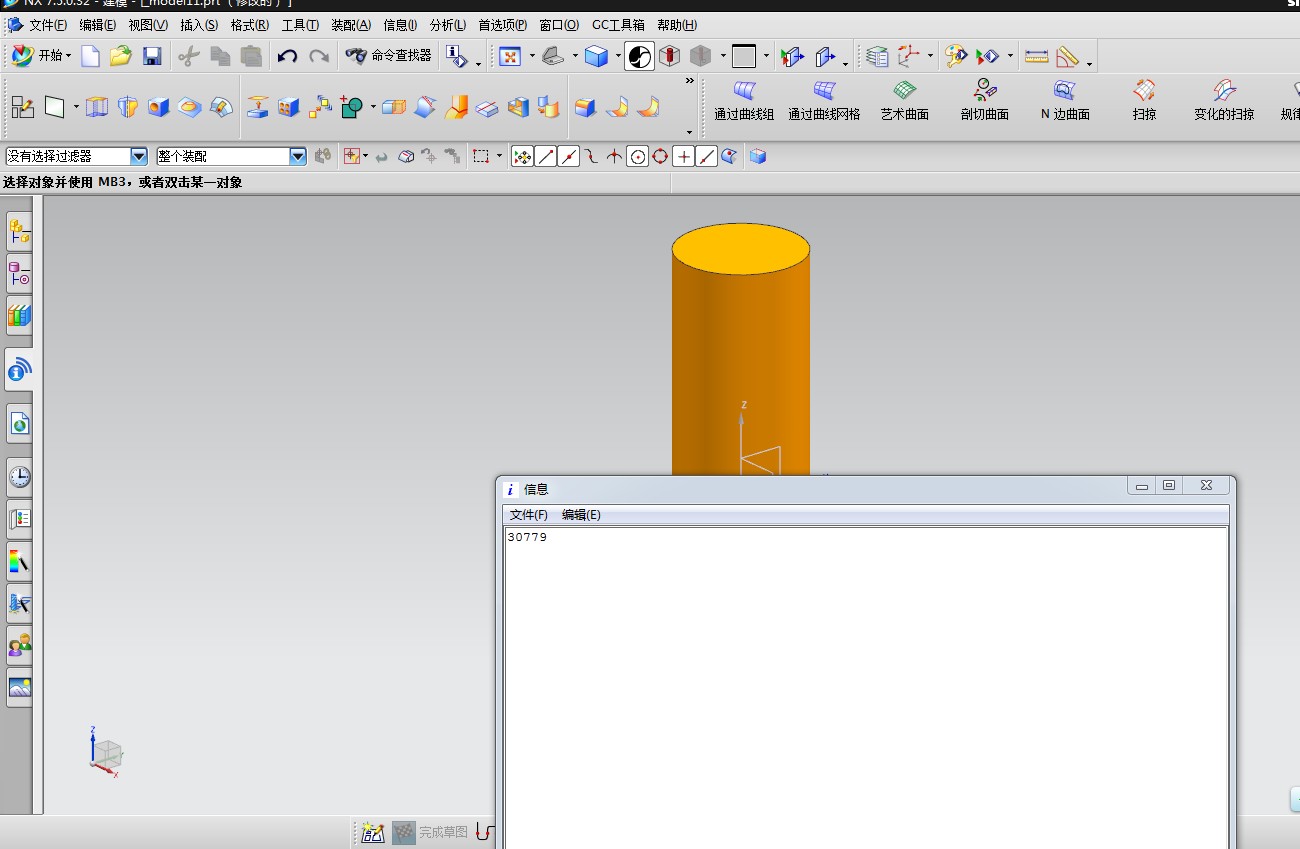
{

return (UF\_UNLOAD\_IMMEDIATELY);

}

**运行结果截屏：**

****

****

**（2）读取文本文件里的内容和往文本文件中写入信息。**

**代码如下：**

#include <stdio.h>

#include <uf.h>

#include <uf\_defs.h>

#include <uf\_exit.h>

#include <uf\_ui.h>

#include <uf\_styler.h>

#include <uf\_mb.h>

#include <uf\_modl.h>

#include <uf\_disp.h>

#include <uf\_obj.h>

#include <fstream>

#include "Multiple\_text.h"

using namespace std;

#define CHANGE\_CB\_COUNT ( 3 + 1 ) /\* Add 1 for the terminator \*/

char prompt\_string[133]="请选取文本文件";

char title\_string[133]="选取文本文件";

char filter\_string[133] = "文本文件";

char default\_name[133]=" ";

char filename[133];

int response=0;

static UF\_STYLER\_callback\_info\_t CHANGE\_cbs[CHANGE\_CB\_COUNT] =

{

{UF\_STYLER\_DIALOG\_INDEX, UF\_STYLER\_APPLY\_CB , 0, CHANGE\_apply\_cb},

{CHANGE\_ACTION\_1 , UF\_STYLER\_ACTIVATE\_CB , 1, CHANGE\_action\_1\_act\_cb},

{CHANGE\_ACTION\_2 , UF\_STYLER\_ACTIVATE\_CB , 1, CHANGE\_action\_2\_act\_cb},

{UF\_STYLER\_NULL\_OBJECT, UF\_STYLER\_NO\_CB, 0, 0 }

};

static UF\_MB\_styler\_actions\_t actions[] = {

{ "Multiple\_text.dlg", NULL, CHANGE\_cbs, UF\_MB\_STYLER\_IS\_NOT\_TOP },

{ NULL, NULL, NULL, 0 } /\* This is a NULL terminated list \*/

};

extern void ufusr(char \*param, int \*retcode, int rlen)

{

int response = 0;

int error\_code = 0;

if ( ( UF\_initialize() ) != 0 )

return;

if ( ( error\_code = UF\_STYLER\_create\_dialog ( "Multiple\_text.dlg",

CHANGE\_cbs, /\* Callbacks from dialog \*/

CHANGE\_CB\_COUNT, /\* number of callbacks\*/

NULL, /\* This is your client data \*/

&response ) ) != 0 )

{

char fail\_message[133];

/\* Get the user function fail message based on the fail code.\*/

UF\_get\_fail\_message(error\_code, fail\_message);

UF\_UI\_set\_status (fail\_message);

printf ( "%s\n", fail\_message );

}

UF\_terminate();

return;

}

extern int ufusr\_ask\_unload (void)

{

/\* unload immediately after application exits\*/

return ( UF\_UNLOAD\_IMMEDIATELY );

/\*via the unload selection dialog... \*/

/\*return ( UF\_UNLOAD\_SEL\_DIALOG ); \*/

/\*when UG terminates... \*/

/\*return ( UF\_UNLOAD\_UG\_TERMINATE ); \*/

}

extern void ufusr\_cleanup (void)

{

return;

}

int CHANGE\_apply\_cb ( int dialog\_id,

void \* client\_data,

UF\_STYLER\_item\_value\_type\_p\_t callback\_data)

{

/\* Make sure User Function is available. \*/

if ( UF\_initialize() != 0)

return ( UF\_UI\_CB\_CONTINUE\_DIALOG );

/\* ---- Enter your callback code here ----- \*/

UF\_terminate ();

/\* Callback acknowledged, do not terminate dialog \*/

/\* A return value of UF\_UI\_CB\_EXIT\_DIALOG will not be accepted \*/

/\* for this callback type. You must respond to your apply button.\*/

return (UF\_UI\_CB\_CONTINUE\_DIALOG);

}

int CHANGE\_action\_1\_act\_cb ( int dialog\_id,

void \* client\_data,

UF\_STYLER\_item\_value\_type\_p\_t callback\_data)

{

/\* Make sure User Function is available. \*/

if ( UF\_initialize() != 0)

return ( UF\_UI\_CB\_CONTINUE\_DIALOG );

/\* ---- Enter your callback code here ----- \*/

UF\_UI\_create\_filebox ( prompt\_string,

title\_string,

filter\_string,

default\_name,

filename,

&response);

/\* 从文件中读信息\*/

int error;

fstream file1(filename);

char \*\*file\_contents = (char\*\*)UF\_allocate\_memory(sizeof(char\*)\*16,&error);

int i=0;

while(!file1.eof())

{

file\_contents[i] = (char\*)UF\_allocate\_memory(MAX\_LINE\_SIZE+1,&error);

file1>>file\_contents[i];

UF\_UI\_write\_listing\_window(file\_contents[i]);

UF\_UI\_write\_listing\_window("\n");

i++;

}

UF\_STYLER\_item\_value\_type\_t data;

data.item\_id = "MTXT\_0";

data.item\_attr = UF\_STYLER\_VALUE;

data.indicator = UF\_STYLER\_STRING\_PTR\_VALUE;

data.count = i;

data.value.strings = file\_contents;

UF\_STYLER\_set\_value(dialog\_id,&data);

UF\_STYLER\_free\_value(&data);

file1.close();

UF\_terminate ();

/\* Callback acknowledged, do not terminate dialog \*/

return (UF\_UI\_CB\_CONTINUE\_DIALOG);

/\* or Callback acknowledged, terminate dialog. \*/

/\* return ( UF\_UI\_CB\_EXIT\_DIALOG ); \*/

}

int CHANGE\_action\_2\_act\_cb ( int dialog\_id,

void \* client\_data,

UF\_STYLER\_item\_value\_type\_p\_t callback\_data)

{

/\* Make sure User Function is available. \*/

if ( UF\_initialize() != 0)

return ( UF\_UI\_CB\_CONTINUE\_DIALOG );

/\* ---- Enter your callback code here ----- \*/

UF\_STYLER\_item\_value\_type\_t data;

data.item\_id = "MTXT\_0";

data.item\_attr = UF\_STYLER\_VALUE;

UF\_STYLER\_ask\_value(dialog\_id,&data);

char prompt\_string[]="Select File";

char title\_string[]="File Access";

char filter\_string[UF\_CFI\_MAX\_PATH\_NAME\_SIZE+1]="\*.txt";

char default\_name[133]="";

char filename[UF\_CFI\_MAX\_PATH\_NAME\_SIZE];

int response;

int error = UF\_UI\_create\_filebox(prompt\_string,

title\_string,

filter\_string,

default\_name,

filename,

&response);

if(error !=0 )

{

return UF\_UI\_CB\_CONTINUE\_DIALOG;

}

fstream file2(filename);

for (int i=0;i<data.count;i++)

{

if(!(file2<<data.value.strings[i]&&file2<<endl))

file2.close();

UF\_STYLER\_free\_value(&data);

UF\_terminate ();

/\* Callback acknowledged, do not terminate dialog \*/

return (UF\_UI\_CB\_CONTINUE\_DIALOG);

/\* or Callback acknowledged, terminate dialog. \*/

/\* return ( UF\_UI\_CB\_EXIT\_DIALOG ); \*/

}

}

**注意：这个项目需要在UG中创建一个人机交互界面**

**8、（1）创建点收集器，创建参考点并返回点的坐标**

**代码如下：**

#include <stdio.h>

#include <uf.h>

#include <uf\_ui.h>

#define UF\_CALL(X) (report( \_\_FILE\_\_, \_\_LINE\_\_, #X, (X)))

static int report( char \*file, int line, char \*call, int irc)

{

if (irc)

{

char messg[133];

printf("%s, line %d: %s\n", file, line, call);

(UF\_get\_fail\_message(irc, messg)) ?

printf(" returned a %d\n", irc) :

printf(" returned error %d: %s\n", irc, messg);

}

return(irc);

}

static void do\_ugopen\_api(void)

{

//相关知识在P76

char message[]="点收集器";

logical coincident\_points = TRUE;

UF\_UI\_chained\_points\_p\_t points;

int count;

int response;

UF\_UI\_select\_point\_collection(message,coincident\_points,&points,&count,&response);

points[0].object;

UF\_UI\_open\_listing\_window();

char str [133];

for(int i=0;i<3;i++)

{

sprintf(str,"%f",points[0].pt[i]);//关键部分

UF\_UI\_write\_listing\_window(str);

UF\_UI\_write\_listing\_window("\n");

}

UF\_free(points);

}

void ufusr(char \*param, int \*retcode, int paramLen)

{

if (!UF\_CALL(UF\_initialize()))

{

do\_ugopen\_api();

UF\_CALL(UF\_terminate());

}

}

int ufusr\_ask\_unload(void)

{

return (UF\_UNLOAD\_IMMEDIATELY);

}

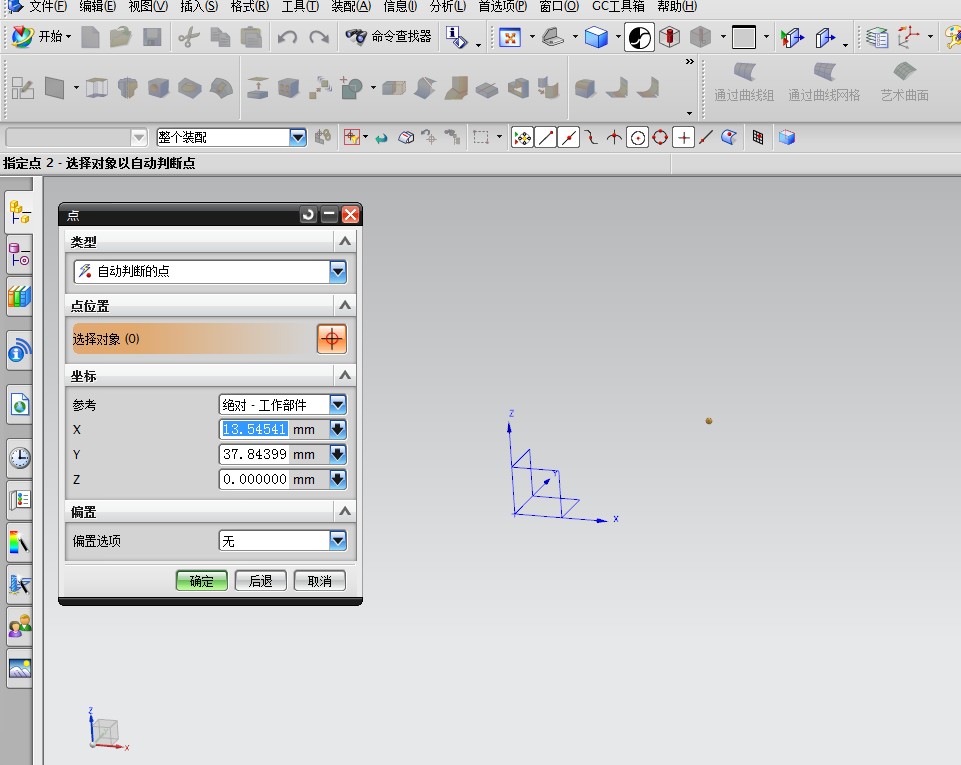
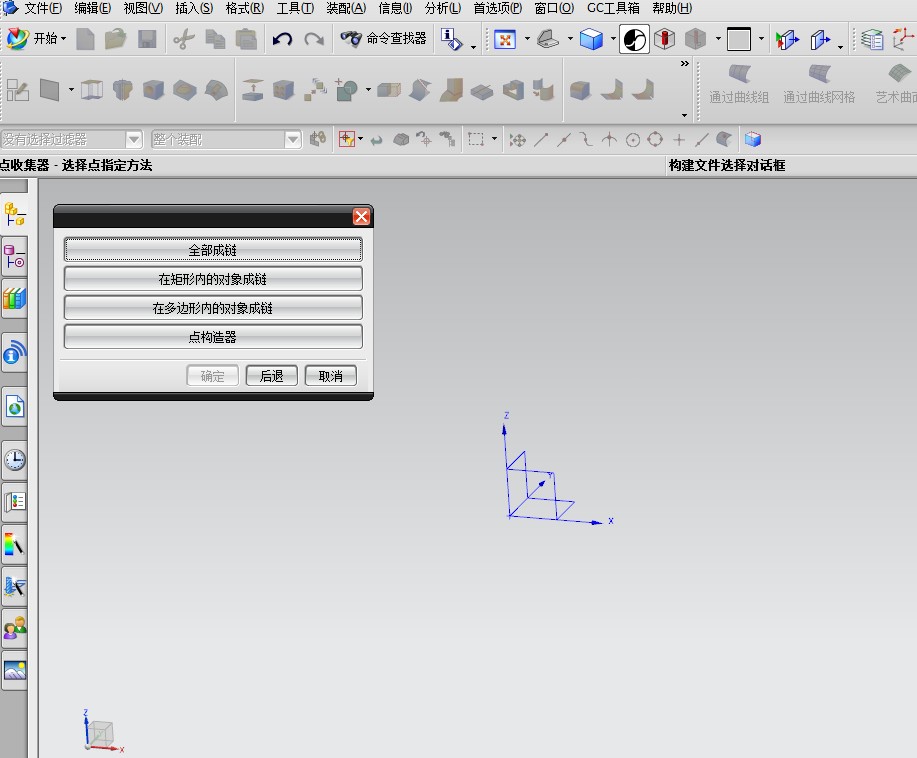
void ufusr\_cleanup(void)

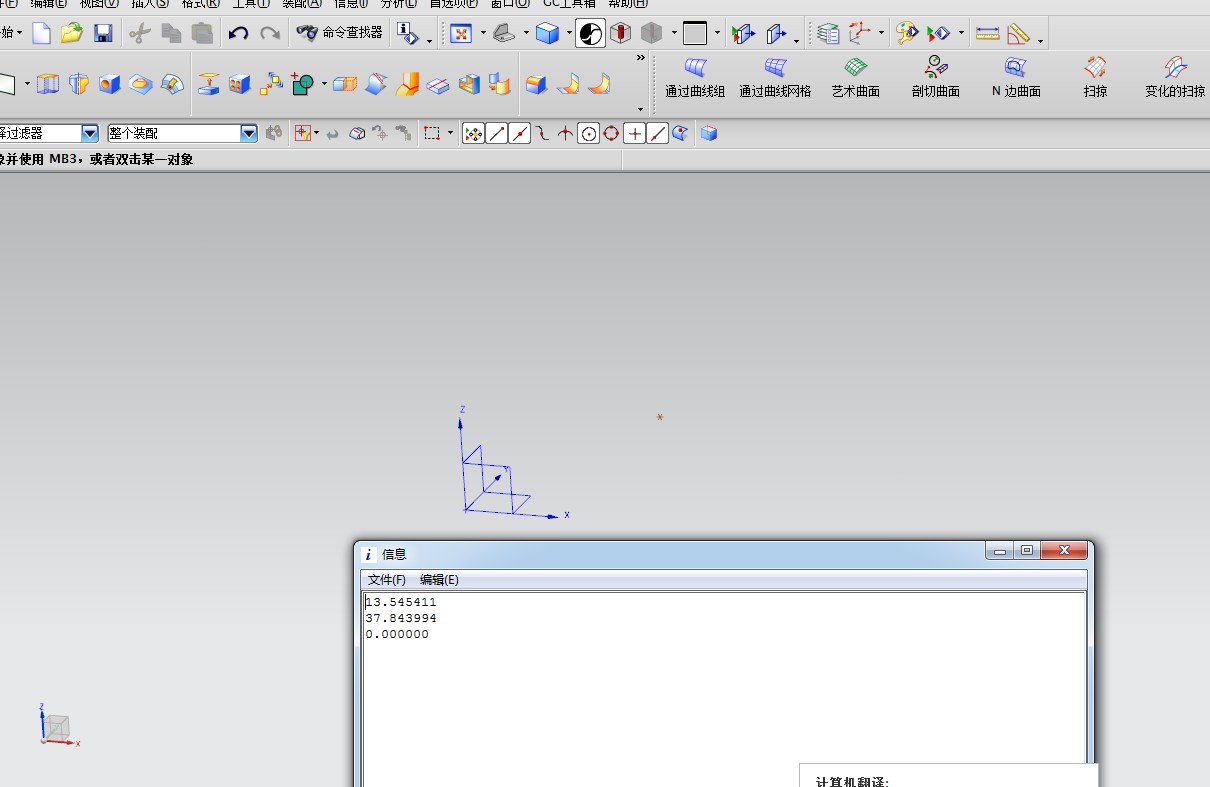
{

/\* Perform cleanup \*/

}

**运行结果截屏：**

****

****

**（2）创建一个对象，用单对象选择对话框选择对象的边，并返回该边的长度。**

代码如下：

#include <uf.h>

#include <stdio.h>

#include <uf\_modl.h>

#include <uf\_ui.h>

#include <uf\_defs.h>

#include <uf\_obj.h>

#include <uf\_gexp.h>

#include <uf\_part.h>

#include <uf\_object\_types.h>

static int init\_proc(UF\_UI\_selection\_p\_t select,void \*user\_data);

#define UF\_CALL(X) (report( \_\_FILE\_\_, \_\_LINE\_\_, #X, (X)))

static int report( char \*file, int line, char \*call, int irc)

{

if (irc)

{

char messg[133];

printf("%s, line %d: %s\n", file, line, call);

(UF\_get\_fail\_message(irc, messg)) ?

printf(" returned a %d\n", irc) :

printf(" returned error %d: %s\n", irc, messg);

}

return(irc);

}

static void do\_ugopen\_api(void)

{

//单对象选择对话框变量参数

char cue[] = "单对象选择对话框";

char title[] = "单对象选择对话框";

int iScope=UF\_UI\_SEL\_SCOPE\_NO\_CHANGE;

int response;

tag\_t view;

tag\_t object;

double cursor[3];

tag\_t xform =null\_tag;

tag\_t feature\_tag;

tag\_t exp\_tag ;

double length ;

char buf[UF\_UI\_MAX\_STRING\_LEN+1];

if(!UF\_CALL(UF\_UI\_select\_with\_single\_dialog(cue,title,

iScope, init\_proc, NULL,

&response, &object, cursor, &view)))

{

if (response == UF\_UI\_OBJECT\_SELECTED ||

response == UF\_UI\_OBJECT\_SELECTED\_BY\_NAME)

{

UF\_GEXP\_create\_length(object,xform,&feature\_tag,&exp\_tag);//创建一个测量长度的表达式

UF\_CALL(UF\_MODL\_ask\_exp\_tag\_value(exp\_tag,&length));//调用表达式的tag值

sprintf(buf, "The length of the linearedge is:%f\n", length);

UF\_UI\_open\_listing\_window();

UF\_UI\_write\_listing\_window(buf);

}

/\* unhighlight selected object \*/

UF\_DISP\_set\_highlight(object,0);

}

}

/\* selection initialization procedure \*/

static int init\_proc(UF\_UI\_selection\_p\_t select,void\* user\_data)

{

int num\_triples = 1;//1代表对象的边

UF\_UI\_mask\_t mask\_triples[] = {UF\_solid\_type, 0, UF\_UI\_SEL\_FEATURE\_LINEAR\_EDGE};

/\* enable only lines and edges \*/

if((UF\_CALL(UF\_UI\_set\_sel\_mask(select,

UF\_UI\_SEL\_MASK\_CLEAR\_AND\_ENABLE\_SPECIFIC,

num\_triples, mask\_triples))) == 0)

{

return (UF\_UI\_SEL\_SUCCESS);

}

else

{

return (UF\_UI\_SEL\_FAILURE);

}

}

void ufusr(char \*param, int \*retcode, int paramLen)

{

if (!UF\_CALL(UF\_initialize()))

{

do\_ugopen\_api();

UF\_CALL(UF\_terminate());

}

}

int ufusr\_ask\_unload(void)

{

return (UF\_UNLOAD\_IMMEDIATELY);

}

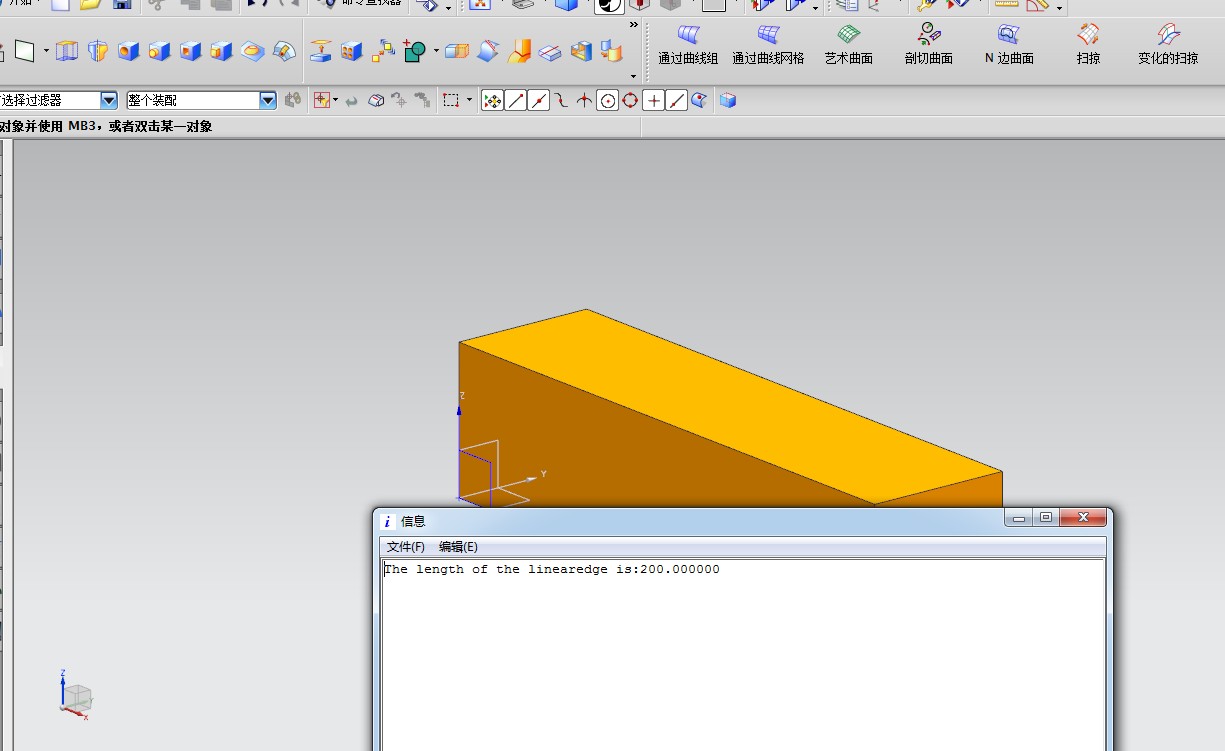
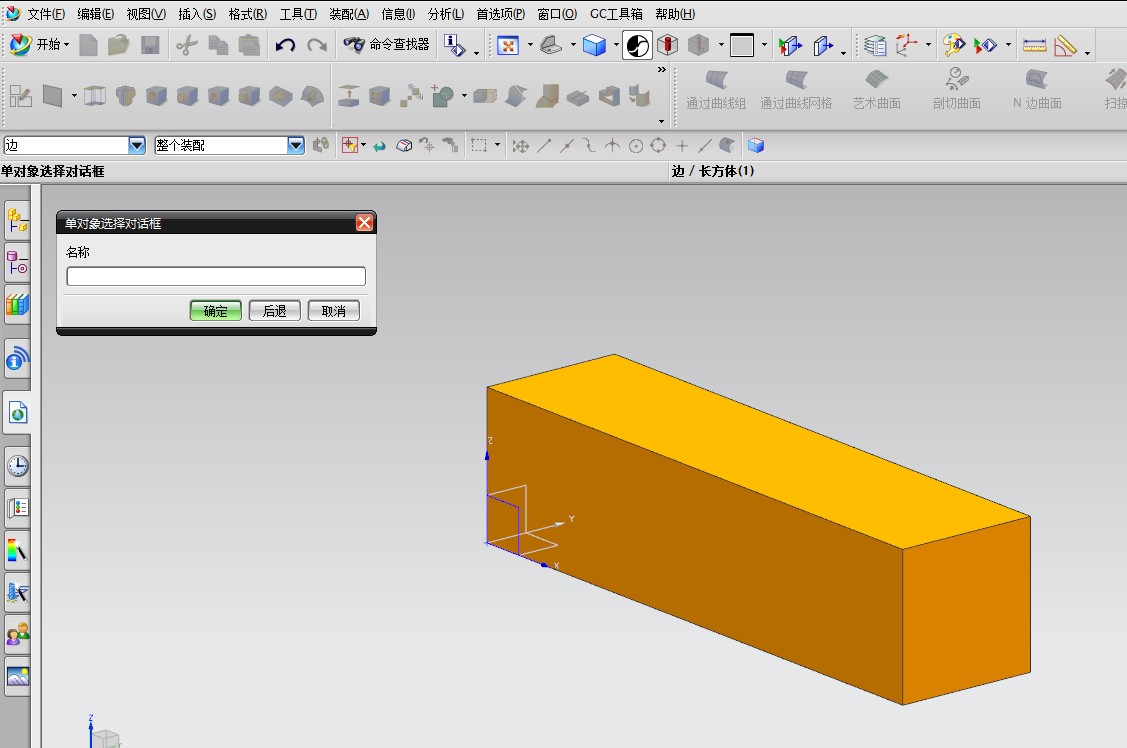
void ufusr\_cleanup(void)

{

/\* Perform cleanup \*/

}

**运行结果截屏如下：**

****

**9、访问外部数据源（哥们的电脑做不起来这个项目，所以没有截屏，但代码是正确的）**

代码如下：

#include <stdio.h>

#include <stdarg.h>

#include <uf.h>

#include <uf\_ui.h>

#include <uf\_object\_types.h>

#include <uf\_disp.h>

#include <uf\_obj.h>

#include <uf\_part.h>

#include <uf\_modl.h>

#include <uf\_assem.h>

#include <afxdb.h>

#include <uf\_csys.h>

#include <uf\_mb.h>

#include <uf\_drf.h>

#include <uf\_xs.h>

#define UF\_CALL(X) (report( \_\_FILE\_\_, \_\_LINE\_\_, #X, (X)))

static int report( char \*file, int line, char \*call, int irc)

{

if (irc)

{

char sFailMessage[133];

UF\_get\_fail\_message(irc,sFailMessage);

char sWholeMessage[256];

sprintf(sWholeMessage,

"File Name:%s\nLine:%d\nFunction:%s\nError Code:%d\nMessage:%s",

file,line,call,irc,sFailMessage);

MessageBox( NULL, sWholeMessage,

"Error in Function", MB\_OK|MB\_ICONWARNING|MB\_SYSTEMMODAL);

}

return(irc);

}

static void error\_return(char \*msg)

{

fprintf(stderr,"\n+++ERROR : %s.\n\n",msg);

exit(1);

}

static void report\_error(int result,char \*alt\_msg)

{

char err\_message[200];

int found = UF\_get\_fail\_message(result,err\_message);

if (found != 0)

error\_return(alt\_msg);

else

error\_return(err\_message);

}

static void create\_blo( double origin\_y, char \*blo\_length,char\*blo\_width,char\*blo\_height )

{

UF\_FEATURE\_SIGN sign = UF\_NULLSIGN;

tag\_t taget = NULL\_TAG;

double blo\_origin [3] = {0,0,0};

//double cyl\_direction[ 3 ] = { 0.0, 0.0, 1.0 };

tag\_t blk\_feat = NULL\_TAG;

char\* edge\_lengths[] ={blo\_length,blo\_width,blo\_height};

blo\_origin[1] = origin\_y;

UF\_MODL\_create\_block (sign,taget,blo\_origin,edge\_lengths,&blk\_feat);

}

static void do\_ugopen\_api(void)

{

CDatabase Database; // 定义一个MFC的CDatabse数据库类对象Databse

CString SQLCommand = "select \* from screw"; // SQL查询语句

DWORD dwOptions = 0; // 设置连接的建立方式

CDBVariant temp; // 数据库数据通用类型

int result; // 储存返回值的变量

char blo\_length[133];

char blo\_width[133];

char blo\_height[133];

double \*blo = NULL;

try

{

result = Database.OpenEx( "DSN=screw", dwOptions ); // 打开数据库

if( result != 0 ) // 如果成功打开

{

CRecordset rs( &Database ); // 定义记录集

if ( rs.Open( CRecordset::snapshot, SQLCommand ) != false )

{

rs.MoveFirst( );

short nFields = rs.GetODBCFieldCount( ); // 获取记录的字段数目

while( !rs.IsEOF( ) )

{

blo =(double \*)malloc( nFields \* sizeof(double) );

for( short index = 0; index < nFields; index++ )

{

rs.GetFieldValue( index, temp );

blo[index] = temp.m\_dblVal;

}

sprintf(blo\_length,"%f",blo[0] );

sprintf(blo\_width,"%f",blo[1] );

sprintf(blo\_height,"%f",blo[2] );

create\_blo( blo[1]\*3,blo\_length,blo\_width,blo\_height);

rs.MoveNext( );

free(blo);

}

rs.Close( ); // 关闭记录集

}

Database.Close(); // 关门数据库连接

}

}

catch( CDBException \*pe ) // 出错处理

{

AfxMessageBox( "Exception!" );

AfxMessageBox( pe->m\_strError );

pe -> Delete( );

}

}

/\*ARGSUSED\*/

void ufusr(char \*param, int \*retcode, int param\_len)

{

if (!UF\_CALL(UF\_initialize()))

{

do\_ugopen\_api();

UF\_CALL(UF\_terminate());

}

}

/\*----------------------------------------------------------------------------\*/

extern int ufusr\_ask\_unload (void)

{

/\* unload immediately after application exits\*/

return ( UF\_UNLOAD\_IMMEDIATELY );

**}**

**10、创建一个拉伸体。**

**代码如下：**

#include <uf.h>

#include <stdio.h>

#include <uf\_obj.h>

#include <stdlib.h>

#include <uf\_view.h>

#include <uf\_part.h>

#include <uf\_modl.h>

#define UF\_CALL(X) (report( \_\_FILE\_\_, \_\_LINE\_\_, #X, (X)))

static int report( char \*file, int line, char \*call, int irc)

{

if (irc)

{

char messg[133];

printf("%s, line %d: %s\n", file, line, call);

(UF\_get\_fail\_message(irc, messg)) ?

printf(" returned a %d\n", irc) :

printf(" returned error %d: %s\n", irc, messg);

}

return(irc);

}

static void do\_ugopen\_api(void)

{

int i;

tag\_t view\_tag = NULL\_TAG;

//定义三根线，作为扫描的截面线串

UF\_CURVE\_line\_t sline1,sline2,sline3;

tag\_t line1,line2,line3;

tag\_t generators[3];

//定义拉特征参数

uf\_list\_p\_t objects;

char \*taper\_angle = "0.0";

char \*limit[ 2 ] = {"0.0","10.0"};//拉伸起点和终点，及长度

double point[ 3 ] = {0.0,0.0,0.0};

double direction[ 3 ]={0.0,0.0,1.0}; //拉伸方向

UF\_FEATURE\_SIGN sign = UF\_NULLSIGN;

uf\_list\_p\_t features;

if(!UF\_initialize())

{

//创建线1

sline1.start\_point[ 0 ] = 0.0;

sline1.start\_point[ 1 ] = 0.0;

sline1.start\_point[ 2 ] = 0.0;

sline1.end\_point[ 0 ] = 0.0;

sline1.end\_point[ 1 ] = 5.0;

sline1.end\_point[ 2 ] = 0.0;

UF\_CURVE\_create\_line(&sline1,&line1);

//创建线2

sline2.start\_point[ 0 ] = 0.0;

sline2.start\_point[ 1 ] = 5.0;

sline2.start\_point[ 2 ] = 0.0;

sline2.end\_point[ 0 ] = 5.0;

sline2.end\_point[ 1 ] = 5.0;

sline2.end\_point[ 2 ] = 0.0;

UF\_CURVE\_create\_line(&sline2,&line2);

//创建线3

sline3.start\_point[ 0 ] = 5.0;

sline3.start\_point[ 1 ] = 5.0;

sline3.start\_point[ 2 ] = 0.0;

sline3.end\_point[ 0 ] = 0.0;

sline3.end\_point[ 1 ] = 0.0;

sline3.end\_point[ 2 ] = 0.0;

UF\_CURVE\_create\_line(&sline3,&line3);

//把三根线的tag放入数组

generators[0] = line1;

generators[1] = line2;

generators[2] = line3;

//创建一个链表，以存放扫描的截面线串

UF\_MODL\_create\_list( &objects);

for(i=0;i<3;i++)

{

UF\_MODL\_put\_list\_item(objects,generators[i]);

}

UF\_MODL\_create\_extruded(objects,taper\_angle,limit,point,direction,sign,&features);

UF\_MODL\_delete\_list(&objects);

UF\_VIEW\_ask\_work\_view(&view\_tag);

UF\_VIEW\_fit\_view(view\_tag,0.7);

UF\_PART\_save();

UF\_terminate();

**}**

**}**

void ufusr(char \*param, int \*retcode, int paramLen)

{

if (!UF\_CALL(UF\_initialize()))

{

do\_ugopen\_api();

UF\_CALL(UF\_terminate());

}

}

int ufusr\_ask\_unload(void)

{

return (UF\_UNLOAD\_IMMEDIATELY);

}

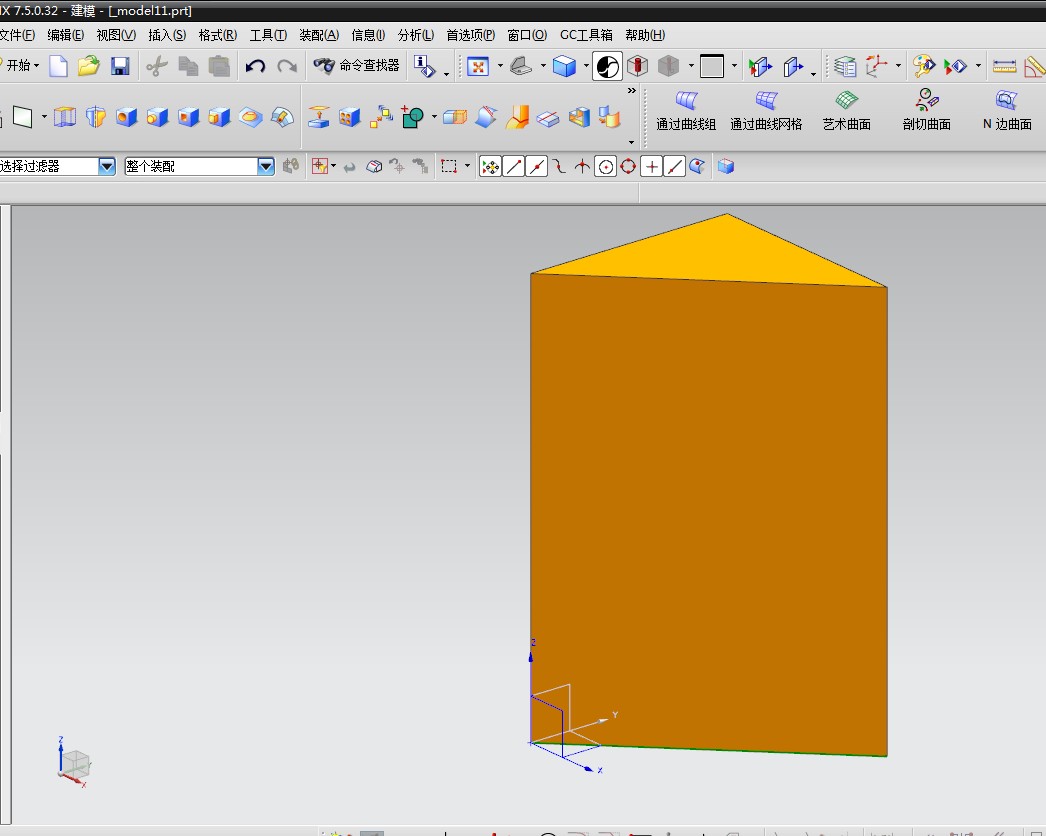
void ufusr\_cleanup(void)

{

/\* Perform cleanup \*/

}

**运行结果截屏如下：**

****

**11、在两个平面的中间创建一个参考平面。**

**代码如下：**

#include <stdio.h>

#include <uf.h>

#include <uf\_ui.h>

#include <uf\_disp.h>

#include<uf\_modl.h>

#define UF\_CALL(X) (report( \_\_FILE\_\_, \_\_LINE\_\_, #X, (X)))

static int report( char \*file, int line, char \*call, int irc)

{

if (irc)

{

char messg[133];

printf("%s, line %d: %s\n", file, line, call);

(UF\_get\_fail\_message(irc, messg)) ?

printf(" returned a %d\n", irc) :

printf(" returned error %d: %s\n", irc, messg);

}

return(irc);

}

static int init\_proc(UF\_UI\_selection\_p\_t select,void\* user\_data)

{

int num\_triples = 2;//2代表面，1代表对象的边

UF\_UI\_mask\_t mask\_triples[] = {

UF\_line\_type, 0, 0,

UF\_solid\_type, 0, UF\_UI\_SEL\_FEATURE\_ANY\_FACE};

/\* enable only lines and edges \*/

if((UF\_CALL(UF\_UI\_set\_sel\_mask(select,UF\_UI\_SEL\_MASK\_CLEAR\_AND\_ENABLE\_SPECIFIC,num\_triples, mask\_triples))) == 0)

{

return (UF\_UI\_SEL\_SUCCESS);

}

else

{

return (UF\_UI\_SEL\_FAILURE);

}

}

static void do\_ugopen\_api(void)

{

char cue[] = "按类选择对话框";

char title[] = "按类选择对话框";

int response, count, i;

tag\_p\_t objects;

/\* Use multiple class selection with scope set to any object

in the work part and without using an initialization procedure.

\*/

if((UF\_CALL(UF\_UI\_select\_with\_class\_dialog(cue, title, UF\_UI\_SEL\_SCOPE\_NO\_CHANGE ,init\_proc, NULL, &response, &count, &objects))) == 0)

{

// printf("object count = %d\n",count);

if (response == UF\_UI\_OK && count > 0)

{

for (i=0; i<count; i++)

{

// printf("object tag = %d\n", objects[i]);

UF\_DISP\_set\_highlight(objects[i],1);

}

//相对参考平面的参数

int num\_refs =2;//两个约束面，最大为3

int point\_select [3] ;

int which\_plane=1;

double reference\_point [3];

char\* angle\_string="0";

char\* offset\_string="0";

int num\_dplanes;

tag\_t dplane\_tag [2] ;

UF\_MODL\_create\_relative\_dplane(num\_refs,objects ,point\_select , which\_plane, reference\_point,angle\_string, offset\_string, &num\_dplanes, dplane\_tag);

UF\_free(objects);

}

}

}

/\*ARGSUSED\*/

void ufusr(char \*param, int \*retcode, int param\_len)

{

if (!UF\_CALL(UF\_initialize()))

{

do\_ugopen\_api();

UF\_CALL(UF\_terminate());

}

}

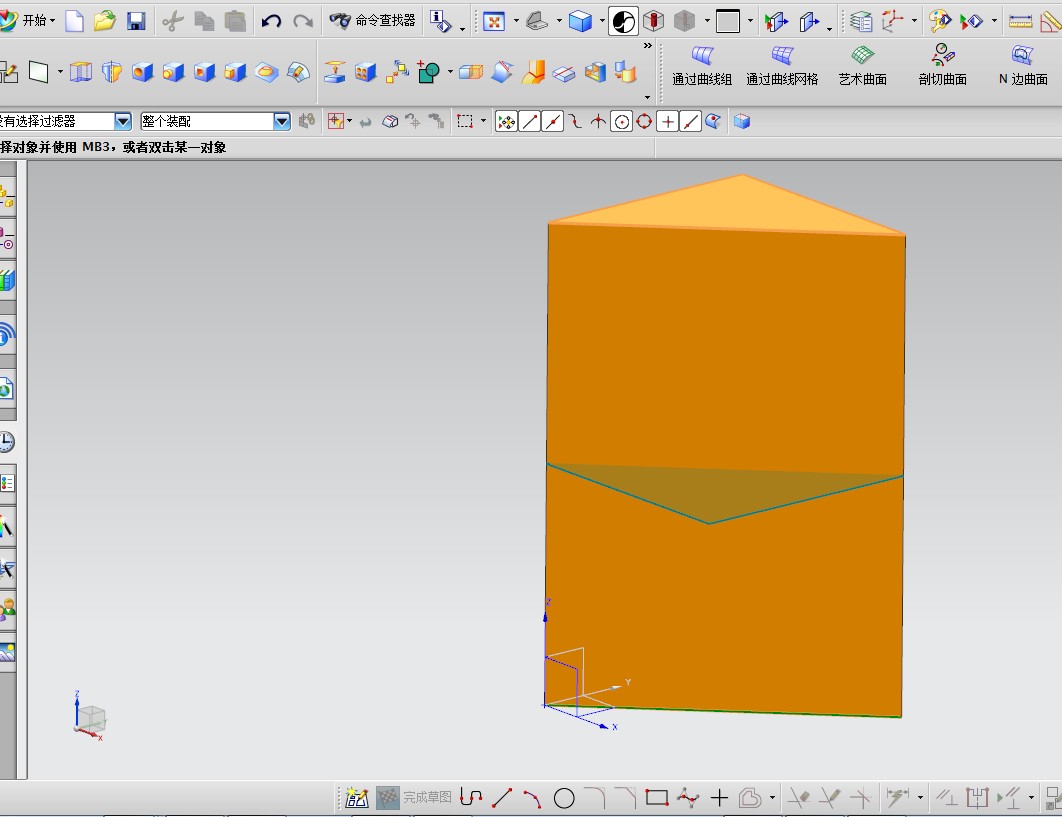
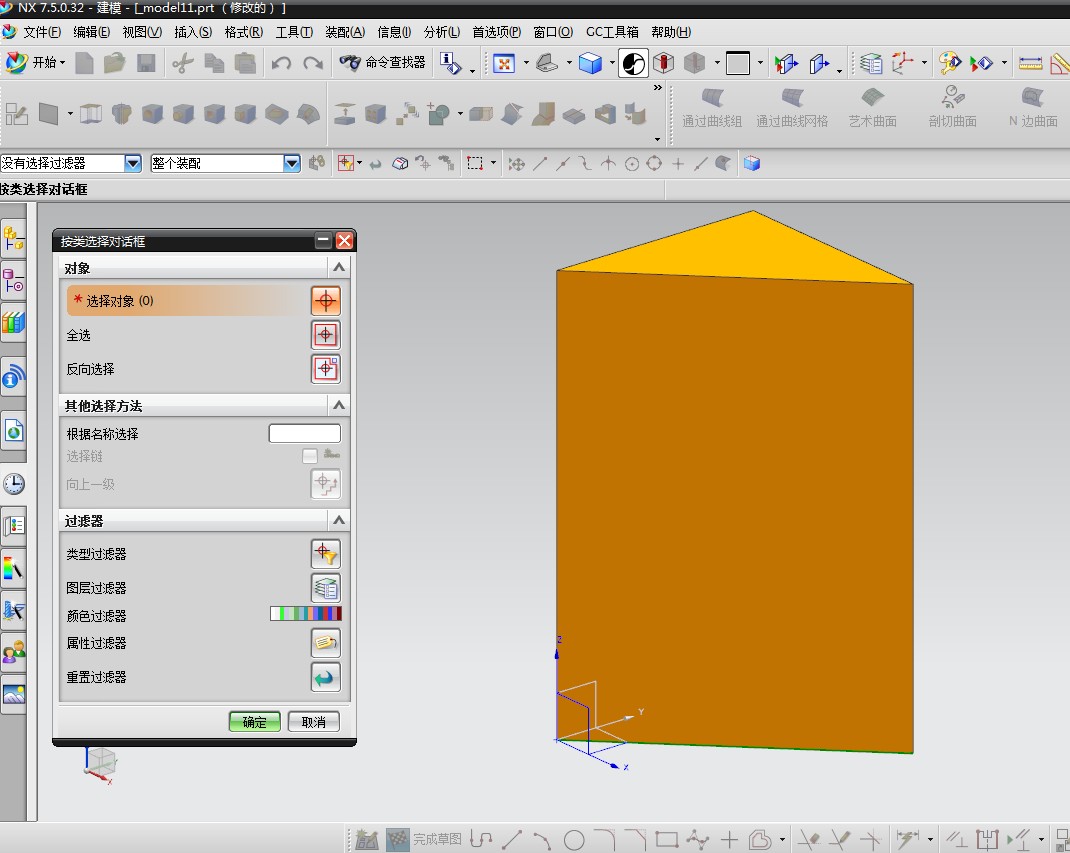
int ufusr\_ask\_unload(void)

{

return (UF\_UNLOAD\_IMMEDIATELY);

**}**

**运行结果截屏如下：**

****

**12、（1）创建一个圆柱体，并利用相对参考平面知识创建一个孔特征（P74）。**

**代码如下：**

#include <stdio.h>

#include <stdlib.h>

#include <uf.h>

#include <uf\_modl.h>

#include <uf\_view.h>

#include <uf\_obj.h>

#include <uf\_layer.h>

#define UF\_CALL(X) (report( \_\_FILE\_\_, \_\_LINE\_\_, #X, (X)))

static int report( char \*file, int line, char \*call, int irc)

{

if (irc)

{

char messg[133];

printf("%s, line %d: %s\n", file, line, call);

(UF\_get\_fail\_message(irc, messg)) ?

printf(" returned a %d\n", irc) :

printf(" returned error %d: %s\n", irc, messg);

}

return(irc);

}

static void do\_ugopen\_api(void)

{

//创建圆柱

UF\_FEATURE\_SIGN sign = UF\_NULLSIGN;

tag\_t target = NULL\_TAG;

double cyl\_origin[3] = {0.0,0.0,0.0};

char \* cyl\_height ="100";

char \* cyl\_diam ="50";

double cyl\_direction[3] = {0.0,0.0,1.0};

tag\_t cyl\_obj;

UF\_MODL\_create\_cylinder(sign,target,cyl\_origin,cyl\_height,cyl\_diam,cyl\_direction,&cyl\_obj);

//创建相对基准平面的变量声明区

int num\_refs = 1;

tag\_t object\_tags[3];

tag\_t object\_tags1[3];

int point\_select[3] = {0,1,0};

int which\_plane = 1;

char \* offset\_string = "25";

int num\_dplanes;

tag\_t dplane\_tag[2];

tag\_t dplane\_tag1[2];

int layer\_fea\_nu;

tag\_t \* layer\_fea\_id;

//创建孔特征的变量声明区

double location[3] = {25.0,0.0,50.0};

double direction[3] = {-1.0,0.0,0.0};

char \* diame ="20";

char \* depth ="80";

char \* angle ="0.0";

tag\_t face\_li;

tag\_t face\_t1;

tag\_t hole\_obj\_id;

//获取圆柱体所有的面，存入链表

uf\_list\_p\_t face\_list;

UF\_MODL\_ask\_feat\_faces(cyl\_obj,&face\_list);

UF\_MODL\_ask\_list\_item(face\_list,2,&object\_tags[0]);

UF\_MODL\_delete\_list(&face\_list);

//设置61层为工作层，将以下创建的基准平面放到61层

UF\_LAYER\_set\_status(61,UF\_LAYER\_WORK\_LAYER);

//创建基准平面

UF\_MODL\_create\_relative\_dplane(num\_refs,object\_tags,point\_select,which\_plane,NULL,NULL,"0",&num\_dplanes,dplane\_tag);

//获取基准平面object tag

object\_tags1[0]=dplane\_tag[0];

UF\_MODL\_ask\_feat\_object(dplane\_tag[0],&layer\_fea\_nu,&layer\_fea\_id);

UF\_OBJ\_set\_blank\_status(layer\_fea\_id[0],UF\_OBJ\_BLANKED);

UF\_free(layer\_fea\_id);

//由第一个基准平面偏置出第二个基准平面，使第二个基准平面与圆柱平面相切

UF\_MODL\_create\_relative\_dplane(1,object\_tags1,point\_select,which\_plane,NULL,NULL,offset\_string,&num\_dplanes,dplane\_tag1);

//设置层1为工作层

UF\_LAYER\_set\_status(1,UF\_LAYER\_WORK\_LAYER);

//创建孔特征

face\_li=dplane\_tag1[0];

face\_t1=object\_tags[0];

UF\_MODL\_create\_simple\_hole(location,direction,diame,depth,angle,face\_li,face\_t1,&hole\_obj\_id);

UF\_terminate();

**}**

void ufusr(char \*param, int \*retcode, int paramLen)

{

if (!UF\_CALL(UF\_initialize()))

{

do\_ugopen\_api();

UF\_CALL(UF\_terminate());

}

}

int ufusr\_ask\_unload(void)

{

return (UF\_UNLOAD\_IMMEDIATELY);

}

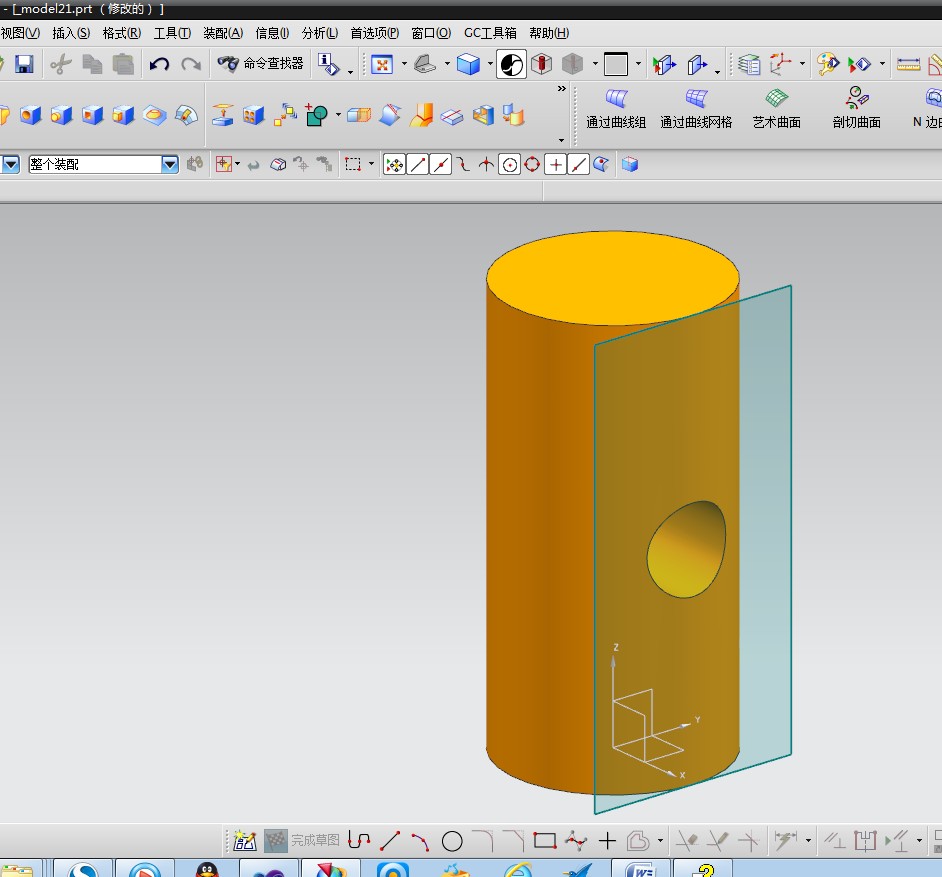
void ufusr\_cleanup(void)

{

/\* Perform cleanup \*/

}

**运行结果截屏：**

****

**（2）在装配环境中导入一个工作部件。**

**代码如下：**

#include <uf.h>

#include <uf\_ui.h>

#include <uf\_modl.h>

#include <stdio.h>

#include <uf\_obj.h>

#include <uf\_part.h>

#include <uf\_disp.h>

#include<uf\_obj.h>

#include<uf\_assem.h>

#include<uf\_layer.h>

void ufusr(char \*param, int \*retcode, int paramLen)

{

if(UF\_initialize()!=0)

return;

tag\_t parent\_part=UF\_ASSEM\_ask\_work\_part();

char part[133]="E:\\workdir\\model1.prt";

char refset\_name[133]="";

const char instance\_name = null\_tag;

double origin[3]={0.0,0.0,0.0};

double csys\_matrix[6]={1,0,0,0,1,0};

int layer;

UF\_LAYER\_ask\_work\_layer(&layer);

tag\_t instance;

UF\_PART\_load\_status\_t error\_status;

UF\_ASSEM\_add\_part\_to\_assembly(parent\_part,part,refset\_name,instance\_name,origin,csys\_matrix,layer,&instance,&error\_status);

UF\_free\_string\_array(error\_status.n\_parts,error\_status.file\_names);

UF\_free(&error\_status);

UF\_terminate();

}

int ufusr\_ask\_unload(void)

{

return (UF\_UNLOAD\_IMMEDIATELY);

}

void ufusr\_cleanup(void)

{

}

**13、在装配环境下打开两个部件，并对其进行“对齐”约束。**

**代码如下：**

#include <uf.h>

#include <uf\_ui.h>

#include <uf\_modl.h>

#include <stdio.h>

#include <uf\_obj.h>

#include <uf\_part.h>

#include <uf\_disp.h>

#include <uf\_obj.h>

#include <iostream>

#include <uf\_assem.h>

using namespace std;

static int init\_proc(UF\_UI\_selection\_p\_t select,void\* user\_data)

{

int num\_triples = 1;

UF\_UI\_mask\_t mask\_triples[] = {UF\_solid\_type,

0,

UF\_UI\_SEL\_FEATURE\_PLANAR\_FACE};

/\* enable only lines and edges \*/

if(UF\_UI\_set\_sel\_mask(select,

UF\_UI\_SEL\_MASK\_CLEAR\_AND\_ENABLE\_SPECIFIC,

num\_triples,

mask\_triples) == 0)

return (UF\_UI\_SEL\_SUCCESS);

else

return (UF\_UI\_SEL\_FAILURE);

}

extern DllExport void ufusr(char \*param, int \*retcode, int paramLen)

{

if(UF\_initialize()!=0)

return;

tag\_t display\_part = NULL\_TAG;

display\_part = UF\_PART\_ask\_display\_part();/\*Returns the tag of the current display part. In a non-assembly part, this is the

same as the work part. If there currently isn't a displayed part, a

NULL\_TAG is returned. \*/

tag\_t root\_occ = UF\_ASSEM\_ask\_root\_part\_occ(display\_part);

if(root\_occ==null\_tag)//判断是否是装配环境

{

uc1601("Please return to the assembly environment",1);

return;

}

int response,count=0;

tag\_p\_t objs;

UF\_UI\_select\_with\_class\_dialog("Select planar faces",

"Align",

UF\_UI\_SEL\_SCOPE\_ANY\_IN\_ASSEMBLY,

init\_proc,

NULL,

&response,

&count,

&objs);

if(response!=UF\_UI\_OK||count<2)

{

uc1601("Please select faces to create align constraint!",1);

return;

}

tag\_t tool\_body=null\_tag,target\_body=NULL\_TAG;

UF\_MODL\_ask\_face\_body(objs[0],&target\_body);

UF\_MODL\_ask\_face\_body(objs[1],&tool\_body);

if( tool\_body == target\_body )

{

uc1601("Please select create!",1);

return;

}

//创建配合关系

UF\_ASSEM\_mating\_condition\_t mc;

tag\_t from\_part\_occ = NULL\_TAG;

tag\_t to\_part\_occ = NULL\_TAG;

//初始化约束

UF\_ASSEM\_init\_mc(&mc);

//Return the tag of the part\_occurrence

UF\_ASSEM\_ask\_parent\_component(objs[0],&from\_part\_occ);

UF\_ASSEM\_ask\_parent\_component(objs[1],&to\_part\_occ);

mc.mated\_object = UF\_ASSEM\_ask\_inst\_of\_part\_occ(from\_part\_occ);//Returns the instance tag of a part occurrence

mc.part\_occurrence = UF\_ASSEM\_ask\_root\_part\_occ(display\_part);//Returns the tag of the root part occurrence

mc.num\_constraints = 1;//约束关系的个数

mc.constraints[0].from\_part\_occ = from\_part\_occ;

mc.constraints[0].from\_status = UF\_ASSEM\_ok;

mc.constraints[0].from = UF\_ASSEM\_ask\_prototype\_of\_occ( objs[0] );

mc.constraints[0].from\_type = UF\_ASSEM\_planar\_face;

mc.constraints[0].to = UF\_ASSEM\_ask\_prototype\_of\_occ( objs[1] );

mc.constraints[0].to\_part\_occ = to\_part\_occ;

mc.constraints[0].to\_status = UF\_ASSEM\_ok;

mc.constraints[0].to\_type = UF\_ASSEM\_planar\_face;

mc.constraints[0].mate\_type = UF\_ASSEM\_align;

UF\_ASSEM\_mc\_status\_t status;

UF\_ASSEM\_dof\_t dof;

double transform\_matrix[4][4];

//求解约束

int result = UF\_ASSEM\_solve\_mc(&mc,&status,&dof,transform\_matrix);

if( result == 0 && status == UF\_ASSEM\_mc\_solved )

{

UF\_ASSEM\_mc\_structure\_state\_t struct\_state;

UF\_ASSEM\_apply\_mc\_data(&mc,&struct\_state,&status);

//更新模型

UF\_DISP\_refresh();

UF\_MODL\_update();

}

UF\_terminate();

}

int ufusr\_ask\_unload(void)

{

return (UF\_UNLOAD\_IMMEDIATELY);

}

void ufusr\_cleanup(void)

{

}

**14（1）创建一个圆柱，并打一个穿孔，倒圆角。**

**代码如下：**

#include <stdio.h>

#include <uf.h>

#include <uf\_ui.h>

#include <uf\_disp.h>

#include<uf\_modl.h>

#include<uf\_layer.h>

#include<uf\_assem.h>

#include<uf\_obj.h>

#include<uf\_part.h>

#include<iostream>

#define UF\_CALL(X) (report( \_\_FILE\_\_, \_\_LINE\_\_, #X, (X)))

static int report( char \*file, int line, char \*call, int irc)

{

if (irc)

{

char messg[133];

printf("%s, line %d: %s\n", file, line, call);

(UF\_get\_fail\_message(irc, messg)) ?

printf(" returned a %d\n", irc) :

printf(" returned error %d: %s\n", irc, messg);

}

return(irc);

}

static void do\_ugopen\_api(void)

{

UF\_FEATURE\_SIGN sign = UF\_NULLSIGN;

tag\_t target = NULL\_TAG;

//创建圆柱的变量声明区

double origin [] = {0.0,0.0,0.0};

char\* height = "100";

char\* cyl\_diam = "80";

double cyl\_direction [] = {0.0,0.0,1.0};

tag\_t cyl\_feat\_tag = NULL\_TAG;

UF\_MODL\_create\_cylinder(sign,target,origin,height,cyl\_diam,cyl\_direction,&cyl\_feat\_tag);

double value;

tag\_t target1;

UF\_MODL\_create\_exp\_tag(height,&target1);//Creates a parameter expression, based upon the input string.

UF\_MODL\_ask\_exp\_tag\_value(target1,&value);//Asks the value of an expression tag.

//创建孔特征的变量声明区

double location[3]={0.0,0.0,0.0} ;

double direction [] = {0.0,0.0,1.0};

tag\_t obj=null\_tag;

char \*diam="30";

char \*angle="0";

tag\_t bottom,top;

tag\_t feature\_obj\_id ;

//链表,把圆柱所有的面放进去

uf\_list\_p\_t face\_list;

UF\_MODL\_create\_list(&face\_list);//Retrieves an object from a linked list of objects

UF\_MODL\_ask\_feat\_faces(cyl\_feat\_tag,&face\_list);

int count=0;

UF\_MODL\_ask\_list\_count(face\_list,&count);//Retrieves the count from a linked list of objects

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

{

UF\_MODL\_ask\_list\_item(face\_list,i,&obj);

int type;

double point [3] ;

double dir [3] ;

double box [6] ;

double radius;

double rad\_data;

int norm\_dir ;

UF\_MODL\_ask\_face\_data(obj,&type,point,dir,box,&radius,&rad\_data,&norm\_dir);//Queries the data associated with a face

if(type==22)//“22”代表有界平面

{

if(fabs(point[2]-origin[2])<0.0005)

{

bottom=obj;

}

if(fabs(point[2]-value)<0.0001)

{

UF\_MODL\_ask\_feat\_location(obj,location);

top=obj;

}

}

}

UF\_MODL\_create\_simple\_hole(location,direction,diam,height,angle,top,bottom,&feature\_obj\_id);

UF\_MODL\_ask\_feat\_faces(feature\_obj\_id,&face\_list);

UF\_MODL\_ask\_list\_count(face\_list,&count);

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

{

UF\_MODL\_ask\_list\_item(face\_list,i,&obj);//Retrieves an object from a linked list of objects

int typel;

UF\_MODL\_ask\_face\_type(obj,&typel);

//倒圆角

if(typel== UF\_MODL\_CYLINDRICAL\_FACE )

{

uf\_list\_p\_t facesl;

UF\_MODL\_create\_list(&facesl);

UF\_MODL\_ask\_shared\_edges(top,obj,&facesl);

const char \* radius="3";

int smooth\_overflow=0;

int cliff\_overflow=0;

int notch\_overflow=0;

double vrb\_tool=3;

tag\_t feature\_obj\_idl;

UF\_MODL\_create\_blend(radius, facesl, smooth\_overflow, cliff\_overflow, notch\_overflow, vrb\_tool, &feature\_obj\_id );

}

}

}

void ufusr(char \*param, int \*retcode, int param\_len)

{

if (!UF\_CALL(UF\_initialize()))

{

do\_ugopen\_api();

UF\_CALL(UF\_terminate());

}

}

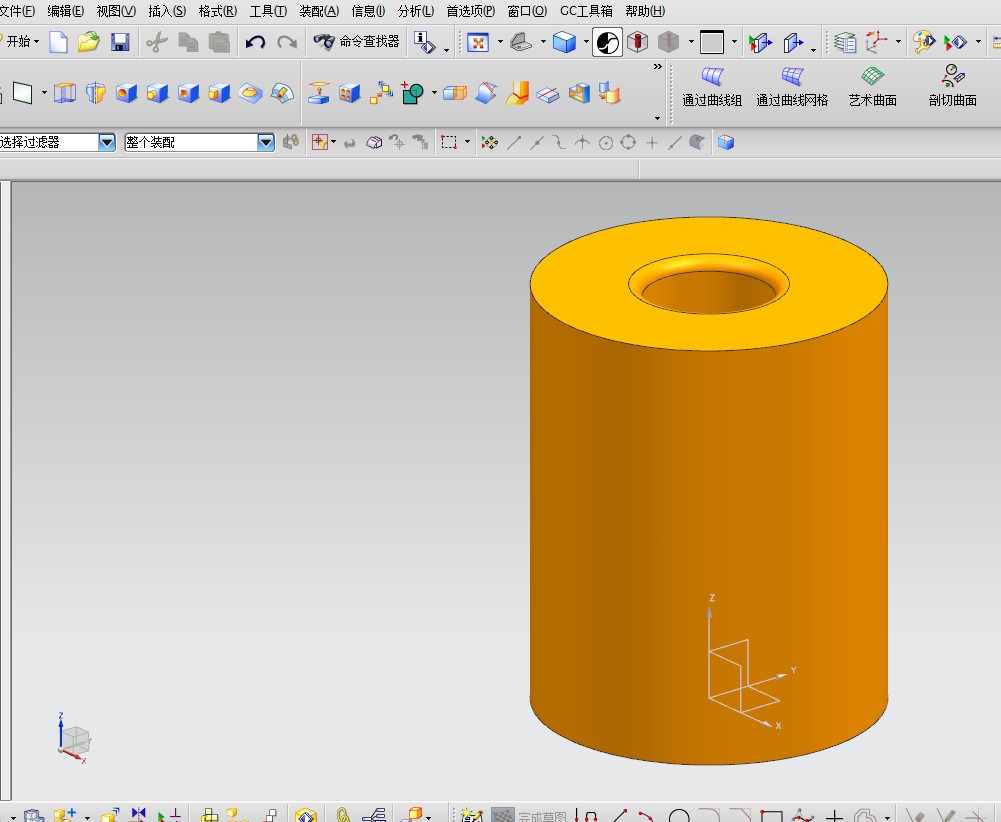
int ufusr\_ask\_unload(void)

{

return (UF\_UNLOAD\_IMMEDIATELY);

}

运行结果截屏如下：



**（2）创建一个对象，使其沿着X轴平移100，以Z轴为旋转轴旋转90度。**

**代码如下：**

#include <uf.h>

#include<stdio.h>

#include<uf\_part.h>

#include<uf\_modl.h>

#include<uf\_obj.h>

#include<uf\_ui.h>

#include<uf\_trns.h>

extern void ufusr(char \*param, int \*retcode, int paramLen)

{

if(UF\_initialize()!=0)

return;

//获取实体

tag\_t display\_part = UF\_PART\_ask\_display\_part();

tag\_t objects = NULL\_TAG;

UF\_OBJ\_cycle\_objs\_in\_part(display\_part,UF\_solid\_type,&objects);

//返回平移的矩阵

double translation[3]={100.0,0.0,0.0};

double matrix1[16];

uf5943(translation,matrix1);

//返回旋转的矩阵

double origin[3]={0.0,0.0,0.0};

double direction[3]={0.0,0.0,1.0};

double degrees\_rotation=90.0;

double matrix2[16];

int status1;

uf5945(origin,direction,&degrees\_rotation,matrix2,&status1);

//实现平移及旋转

const int n\_objects=1 ;

const int move\_or\_copy=1;

const int dest\_layer=0 ;

const int trace\_curves=2 ;

tag\_t copies;

tag\_t trace\_curve\_group;

int status2;

uf5947(matrix1,&objects,&n\_objects,&move\_or\_copy,&dest\_layer,

&trace\_curves,&copies,&trace\_curve\_group,&status2);

uf5947(matrix2,&objects,&n\_objects,&move\_or\_copy,&dest\_layer,

&trace\_curves,&copies,&trace\_curve\_group,&status2);

UF\_terminate();

}

int ufusr\_ask\_unload(void)

{

return (UF\_UNLOAD\_IMMEDIATELY);

}

void ufusr\_cleanup(void)

{

}