感觉他们金山的 BKWin 内面包含的这些类是一个专门为开发类似界面而写的.

界面初步分析

主界面用的对话框框架,装载了金山自己定义的窗口.大部分的控件都并非 Windows 控件(除了中间的列表控件),而是 GDI 画出来的.代码只列出了的界面有关部分.

→框架窗口类定义:

```
class KmainDlg//继承以下几个类
   : public CBkDialogImpl<KMainDlg>//KmainDlg主要实现了这个模板功能,定义见下
    , public CWHRoundRectFrameHelper<KMainDlg>   //处理窗口最大化/恢复用
{}
```

-->CbkDialogImpl 的定义:

```
template <class T, class TBkView = CBkDialogView, class TBase = CWindow,
class TWinTraits = CControlWinTraits>
class CBkDialogImpl : public CWindowImpl<T, TBase, TWinTraits>
{
protected:
    TBkView m_richView;// 这里相当于 CbkDialogView m_richView;
                          它所有的控件都是在这个类里面画的.
                          控件信息都在放在 xml 资源文件里边.
}
```

-->m_richView的类定义

```
class CbkDialogView     //只是个空的实现类,主要是CbkDialogViewImpl功能
    : public CBkDialogViewImpl<CBkDialogView>
{
}
```

--> CbkDialogViewImpl的定义:

```
template <class T, class TBkWin = CBkDialog, class TBase = ATL::CWindow,
class TWinTraits = CBkDialogViewTraits>
class  CBkDialogViewImpl
    : public ATL::CWindowImpl<T, TBase, TWinTraits>
    , public CBkViewImpl<T>
{
protected:
    TBkWin m_bkHeader; //根据默认参数,即CbkDialog m_bkHeader;
    TBkWin m_bkBody;    //              CbkDialog m_bkBody;
```

```
    TBkWin m_bkFooter;  //                CbkDialog m_bkFooter;
        //这三个参数分别代表了不同的部分.标题,躯体,以及最下面的部分.如下图
}
```

类继承关系图:

```
            CBkObject
                |--CBKWindow
                        |--CBButton…
                        |--CBKContainerWnd
                                    |--- CBkPanel
                                                |---CBkDialog
```

其中,CbkObject 有个很重要的虚函数 :

```
virtual BOOL Load(TiXmlElement* pXmlElem)
```

从 xml 资源文件中加载信息

当然,它们的 xml 资源文件怎么去写,肯定有自己的一套规则.

CBKWindow 封装了一个默认的窗口.它不是一个 Windows 窗口类,只是画出来的.

如: // BkWindow Handle

typedef DWORD **HBKWND**;

连 OnCreate(),OnDestroy()都是空的

这些类大部分代码都是用来绘制控件,以及响应控件坐标行为变动的.

CbkContainerWnd 内部只有几个空的虚函数而已.

CbkPanel 这个类用得非常多:

```
Class CbkPanel:public CBkContainerWnd
```

```
    {
        CAtlList<CBkWindow *> m_lstWndChild;//这个 list 包含了所有的内部控件
        _CreateBkWindowByName(LPCSTR lpszName)//动态创建类
        {
            CBkWindow *pNewWindow = NULL;

                        //根据xml加载的string来用CheckAndNew动态创建
                pNewWindow = CBkDialog::CheckAndNew(lpszName);
                 if (pNewWindow)
                        return pNewWindow;
          … …
    }
void _ComposingPanel(LPWINDOWPOS lpWndPos);//安置控件
}
```

这个类把一些消息都分发给了 m_lstWndChild 里面的控件,如:MouseMove, OnPaint, OnEraseBkgnd, OnDestroy, OnCreate.

中间的列表也是自绘来的,整个BKWin体系挺复杂烦琐的.正准备用它们的方法模拟个简单的实现,尤于时间不多,原来这 WTL 方面的项目也很少做,所以只刚开了个头.

自定义资源简单使用

自定义资源文件一般放在 xxx.rc2 文件.

代码如:

```
IDR_XML1 RESXML "res\\warning_dlg.xml"
ISR_XML_DEF_STR RESXML "res\\def_style.xml"
```

格式为:资源 ID　资源类型　　资源位置

程序运行时释放资源:

```
    HINSTANCE hInstance = ::GetModuleHandle(NULL);//当前exe文件的句柄
    HRSRC hResInfo = ::FindResource(hInstance, MAKEINTRESOURCE(IDR_XML1),
_T("RESXML"));
    HGLOBAL hgRes = ::LoadResource(hInstance, hResInfo);
    DWORD cbRes = ::SizeofResource(hInstance, hResInfo);
    void* pvRes = ::LockResource(hgRes);
```

然后根据指针 pvRes,及其长度 cbRes 使用资源文件.

最后调用::FreeResource(hgRes)释放.

Xml 资源文件简单使用

使用 libxml,iconv 库以 UTF-8 格式加载,汉字可使用 iconv 库转换成 GB2312

```
docPtr=xmlReadMemory((const
char*)pvRes,cbRes,NULL,"UTF-8",XML_PARSE_RECOVER);
```

➔ iconv_open(), iconv(), iconv_close() .

金山 XML 资源文件定义格式

附加信息加载
在初始化时一般可以看到如下代码:

```
void KAppModule::_InitUIResource()
{
    BkFontPool::SetDefaultFont(_T("宋体"), -12);
    BkString::Load(IDR_STRING_DEF);
    BkSkin::LoadSkins(IDR_SKIN_DEF);
    BkStyle::LoadStyles(IDR_STYLE_DEF);
}
```

使用BKString::Load(ID):用来加载XML资源文件.储存在容器CAtlMap<UINT, CString> m_mapString中. 使用时利用static函数BKString::Get(ID)释放出来.

```
    static BkString* _Instance()
    {
        if (!ms_pInstance)
            ms_pInstance = new BkString;
        return ms_pInstance;
    }
```

BKString设计为特殊模式类,使用时不需要使用实例.类中函数都使用_Instance()函数来返回唯一对象.BKStyle及BKSkin类也类似.
专门用来存储界面上的出现的有关文字.

BKSKin和BKStyle是紧密相连的.
BKSkin内里的存储容器是CAtlMap<CStringA, CBkSkinBase *> m_mapPool;
我的理解是这样的:BKSKin是为CBKSkinBase提供服务的.加载,动态创建等.
CbkSkinBase是用来定制绘画行为的一组功能类.如颜色,形状等.
CBKSkinBase的派生类如下:

CBKSkinBase的派生类	关键字(用来匹配xml中的节点名,根据此来生成类实例)
CBKImageSkin	imglst
CbkSkinImgFrame	imgframe
CbkSkinImgHorzExtend	imghorzex
CbkSkinButton	button
CbkSkinGradation	gradation
CbkPngSkin	png

一种派生类用于画出一种界面元素.可根据XML节点属性来定制和微调.

```
Skin XML样例:
<skins>
<imghorzex name=mainbghead src=1 left=3/>
<imgframe  name=mainbgbody src=2 top=2 left=1 mode=mask crbg=FBFCFD/>
<imghorzex name=mainbgfoot src=3 left=3/>
<png       name=btnStop    src=3 subwidth=122/>
</skins>
```

根节点为skins,里面是一组节点,src代表.sc2属性值文件中对应BMP资源ID.

BKStyle对应的则是字体形状,鼠标形状,鼠标悬停时颜色,背景颜色,SKIN等控件信息.

```
Style XML样例:
<style>
<class name="msgboxhead" skin="msgboxhead" font="0000" crtext="000000"
crbg=FBFCFD />
<class name="msgboxbody" skin="msgboxbody" font="0000" crtext="000000"
crbg=FBFCFD />
<class name="msgboxfoot" skin="msgboxfoot" font="0000" crtext="000000"
crbg=FBFCFD />
</style>
```

主界面XML加载

根据bkwin/bkdlgview.h文件中的加载xml的代码部分.
主窗口一般继承CbkDialogImpl<T>模板

```
    class CMainDlg
    : public CBkDialogImpl<CMainDlg>
    , public CWHRoundRectFrameHelper<CMainDlg>
{ }
```

并在构造函数里设置主界面资源ID,如:

```
CMainDlg:: MainDlg()
    : CBkDialogImpl<CMainDlg>(IDR_BK_MAIN_DIALOG)()
```

XML:根节点必须为layer,appwin和resize一般为1,header,footer,body分别代表标题,中间,状态栏.

```
<layer title="金山卫士 2.1" width="800" height="570" appwin="1"
resize="1">
    <header class="msgboxhead" width="full" height="32">
        <dlgfile pos="0,0,-0,-0" src="509" />
    </header>
    <footer class="mainfoot" width="full" height="23">
        <dlgfile pos="0,0,-0,-0" src="508" />
    </footer>
    <body class="dlgfoot" width="full" height="full">
        <dlgfile pos="5,0,-5,-5" src="504" />
    </body>
</layer>
```

Dlgfile代表使用另外一个资源xml来加载子元素.对应类CbkDialogFile

属性	属性值代表的意思
src	资源ID
show	1,可见;0,不可见
width	代表控件长度;full,最长;否则,自动适应;
height	同上
id	控件唯一的核心整数值,类似window控件的HANDLE,可根据此值设置消息映射
class	style标识
pos	以’,’分开 ,标识控件位置.

Tab控件的定义,以tabctrl为父节点.tab节点代表子页面.如下:

```
<tabctrl>
    <tab id=" " />
    <tab id="1234" >
    </tab>
</tabctrl>
```

属性	属性值代表的意思
tabspacing	tab标签之间的空格长度
tabpos	最左tab标签应右移的距离
width	代表控件长度;full,最长;否则,自动适应;
height	同上
id	控件唯一的核心整数值,类似window控件的HANDLE,可根据此值设置消息映射
class	style标识
pos	以',' 分开 ,标识控件位置.左,顶,长,高.负数代表父控件的最大值减去此值

金山库使用步骤

1. 主对话框设计

主对话框类需从以下模板派生:CBkDialogImpl<T> CWHRoundRectFrameHelper<T> CbkDialogImpl 封装了金山对话框绝大部分特性.CWHRoundRectFrameHelper 则提供了最大化/恢复时对话框大小变动时里面的控件的控制,使用把消息链入 CWHRoundRectFrameHelper 基类,可以使对话框里面 m_richView 视图控制的控件轻松应对最大化/恢复的情形.

2.消息宏.

使用应用程序的时候，你必须在你的类中定义“消息映射”,类似普通 WTL 程序。如下示例代码:

```
BEGIN_MSG_MAP_EX(KMainDlg)
    MSG_BK_NOTIFY(IDC_RICHVIEW_WIN)
    CHAIN_MSG_MAP(CBkDialogImpl<KMainDlg>)
    CHAIN_MSG_MAP(CWHRoundRectFrameHelper<KMainDlg>)
    MSG_WM_INITDIALOG(OnInitDialog)
    MSG_WM_TIMER(OnTimer)
    REFLECT_NOTIFICATIONS_EX()
END_MSG_MAP()
```

在主对话框类里加入 wtl 增强的对消息的控制宏 BEGIN_MSG_MAP_EX,END_MSG_MAP 对,加入一些对话框常规初始化的消息控制,如 MSG_WM_INITDIALOG(OnInitDialog)等.
然后使用把消息链入基类实现 CHAIN_MSG_MAP(baseClass)

MSG_BK_NOTIFY(IDC_RICHVIEW_WIN)把消息传入 BK_NOTIFY_MSP()
,BK_NOTIFY_MAP_END()宏对.

```
BK_NOTIFY_MAP(IDC_RICHVIEW_WIN)
    BK_NOTIFY_ID_COMMAND(IDC_BTN_SYS_CLOSE, OnBtnClose)
    BK_NOTIFY_ID_COMMAND(IDC_BTN_SYS_MAX, OnBtnMax)
    BK_NOTIFY_ID_COMMAND(IDC_BTN_SYS_MIN, OnBtnMin)
    BK_NOTIFY_MAP_END()
```

使用 BK_NOTIFY_ID_COMMAND(控件 ID,响应函数)为你所感兴趣的控件添加事件控制.
一般为左键单击消息.

文档 2

在主消息映射表中加入:MSG_BK_NOTIFY(IDC_RICHVIEW_WIN)
使 NOTIFY 消息进入的 BK_NOTIFY_MAP　BK_NOTIFY_MAP_END 结构.

在消息映射表中加入
BK_NOTIFY_MAP(IDC_RICHVIEW_WIN)
BK_NOTIFY_ID_COMMAND(STOP_BTN, OnStop)
BK_NOTIFY_MAP_END()
使流入IDC_RICHVIEW_WIN的消息转到View里边的控件.

BK_NOTIFY_ID_COMMAND此宏只接收WM_NOTIFY消息结构中的code为BKNM_COMMAND的消息. 在此一般为CLICK 消息,

```
BKWIN_DECLARE_ATTRIBUTES_BEGIN()
    BKWIN_XXX_ATTRIBUTES(“name”,m_nValueOfName,BOOL)
BKWIN_DECLARE_ATTRIBUTES_END()
```

此宏定义了虚函数SetAttributes.作用是设定m_nValueOfName为XML中name属性所对应的值.

简单实例及步骤

金山的 google code 页面:http://code.google.com/p/bkwin-tutorial/wiki/Index

资源文件样例及效果图如下:

```
<layer title="bkwin" width="600" height="470" appwin="1">
	<header class="mainhead" width="full" height="46" crbg=D8FEFB>
		<icon src="101" pos="5,4"/>
		<text class="dlgtitle" pos="25,6">bkwin</text>
		<imgbtn id="60003" class="linkimage" skin="minbtn" pos="-90,1"/>
		<imgbtn id="60002" class="linkimage" skin="maxbtn" pos="-60,1"/>
		<imgbtn id="60001" class="linkimage" skin="closebtn" pos="-30,1"/>
	</header>
	<body class="mainbody" width="full" height="full">
		<dlg pos="0,0,-0,-0" crbg=F7FBBF>
			<text class="hellowordstyle" pos="100,100">hello world!</text>
		</dlg>
	</body>
	<footer class="mainfoot" width="full" height="29" crbg=FFB9B9>
	</footer>
</layer>
```

```
<style>
<class name="hellowordstyle" font=2004 crtext=FF0000 />
<class name="linkimage" cursor="hand"/>
<class name="mainhead" font=0004 crtext=000000 crbg=FBFCFD />
</style>
```

```
<skin>
	<imglst name="closebtn" src="101" subwidth="30"/>
	<imglst name="maxbtn" src="103" subwidth="30"/>
	<imglst name="minbtn" src="104" subwidth="30"/>
</skin>
```

颜色及字体皆在 class 值对应的 style 文件中设置.具体如下:
颜色和 font 以十六进制方式读入.
颜色值:共六个数,如:FBFCFD.二个为一组,共有三组,分别代表 16 进制的 RGB 值.
我做了个小程序生成他的 xml 里面使用的颜色值:　FTP://10.1.0.117/liaogang/bkwin/

Font 的四个值分别表示: 字体大小;无;无;
最后一个值的二进制则表示 无,粗体,下画线,斜体;

故字体大小在第一个值中设置:
最后一个值为以下:

0	正常
1	斜体
2	下画线
3	下画线+斜体
4	粗体
5	粗体+斜体
¬6	粗体+下画线
7	粗体+下画线+斜体

说明:font 为字体样式;crtext 为字体颜色,crbg 为背景色,x-margin 为客户区水平边空白,类似页边距,y-margin 为上下边.margin 为左右上下相同值

分页程序样例 2 及效果如下:

```
<layer title="title" width="800" height="570" appwin="1" resize="1" show="1">
<!----------------------header,鼠标可按此区域拖动整个窗口------------------------>
    <header class="mainhead" width="full" height="32">
        <text    pos="25,6">%str1% - build 1.0.0</text>
        <imgbtn id="60003" class="linkimage" skin="minbtn" pos="-105,0"/>
        <imgbtn id="60002" class="linkimage" skin="maxbtn" pos="-73,0"/>
```

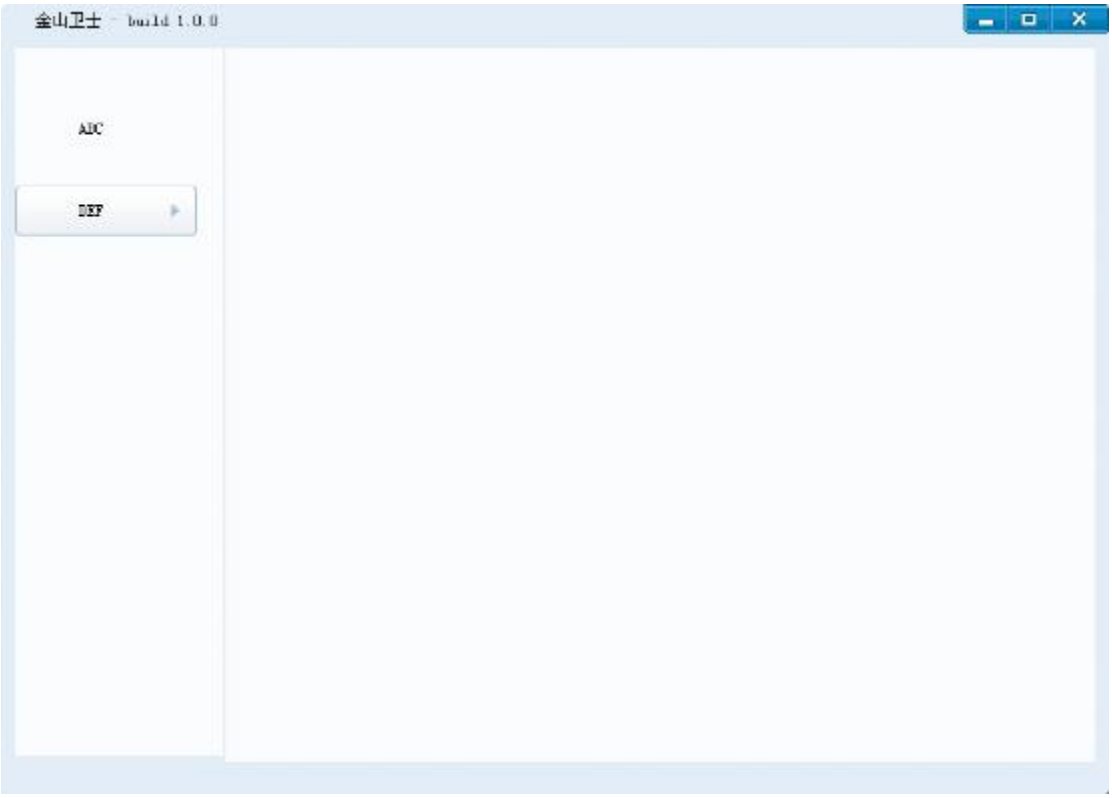
```
        <imgbtn id="60001" class="linkimage" skin="closebtn" pos="-43,0"/>
    </header>
    <!----------------------footer------------------------>
    <footer    class="mainfoot" width="full" height="23"/>
    <!----------------------body-------------------------->
    <body    class="mainbody" width="full" height="full">
    <!----------------------整个面板-------------------->
        <dlg pos="0,0,-0,-0">
    <!----------------------左边的蓝色面板-------------------->
            <dlg pos="0,0,150,-5" crbg="FBFCFD">
                <!--在这里加入 tabctrl 控件,及 tab 子标签控件→
            </dlg>
    <!----------------------右边的蓝色面板-------------------->
            <dlg pos="152,0,-0,-0" crbg="FBFCFD">
            </dlg>

        </dlg>
    </body>
</layer>
```

```
<style>
<class name="tabstyle" skin="tabskin" font=0000 crtext=000000 crbg=F7F995/>
<class name=mainhead    font=0000 crtext=000000 crbg=E2EDF7/>
<class name=mainbody font=0000 crtext=000000 crbg=E2EDF7 x-margin=10/>
<class name=mainfoot    font=0000 crtext=000000 crbg=E2EDF7 x-margin=10 y-margin=1/>
</style>
```

```
<skins>
<button name="tabskin" bguphover=F7F995/>
<imghorzex name=mainbghead    left=3/>
<imgframe name=mainbgbody    top=2 left=1 crbg=E2EDF7 />
<imghorzex name=mainbgfoot    left=3/>
</skins>
```

在主对话框样式 xml 中插入下面的代码.以实现 tab 控件.效果如下图:

```
<tabctrl id="4001" tabwidth="131" tabalign="left" tabheight="37" tabspacing="22" tabpos="40" framepos="10" pos="0,0,-0,-0" text-x="-10" crbg1="F7F995" tabskin="tableft">
                        <tab title="ABC" class="tabstyle" width="full" height="full" valign="middle">
                        </tab>
                        <tab title="DEF" width="full" height="full" valign="middle">
                        </tab>
</tabctrl>
```



消息宏中插入对标签控件控制:BK_NOTIFY_TAB_SELCHANGE(4001,OnTabSelectChanged)
再在右边的空白区域加入两个文本控件以响应标签页的更改

在右边的<!----------右边的蓝色面板---------->这行下面的 dlg 控件中加入如下代码,并设置好 id 值,好在程序中加以控制.

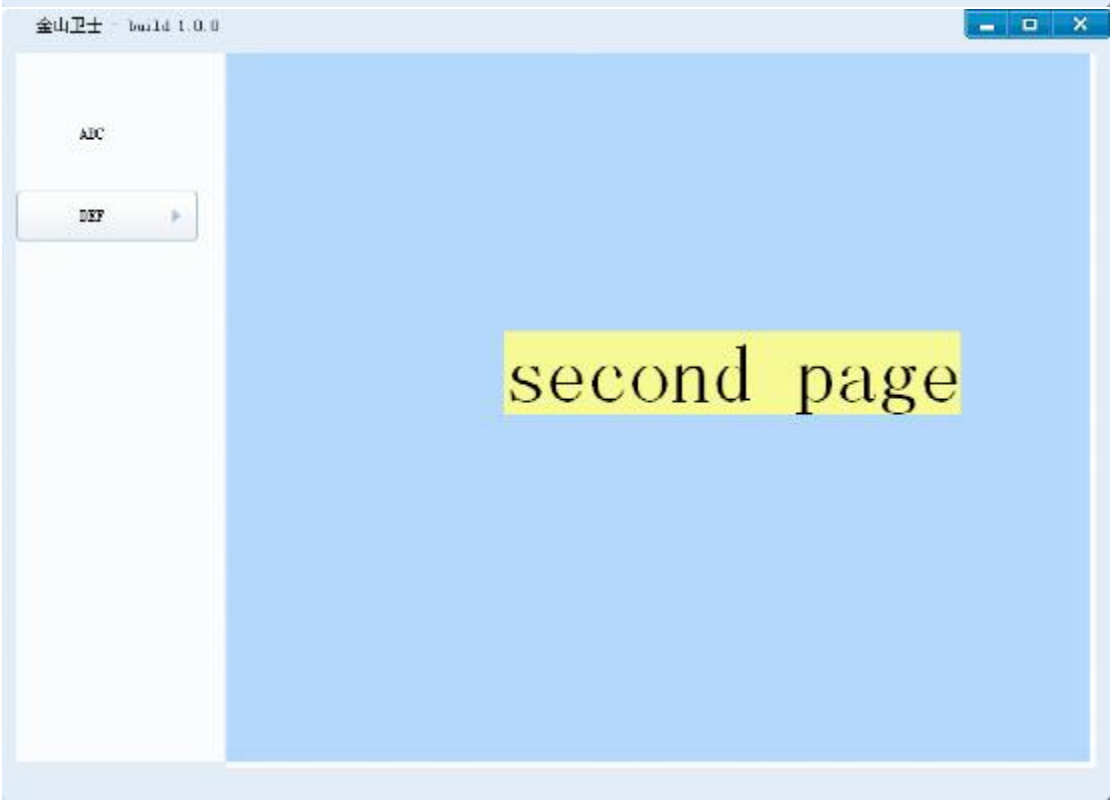
```
<dlg id="11001" pos="0,0,-5,-5" crbg="F8DBD1">
                        <text pos="200,200" class="tab1">first page</text>
</dlg>
<dlg id="11002" pos="0,0,-5,-5" crbg="B4D8FC">
                        <text pos="200,200" class="tab2">second page</text>
</dlg>
```

好,再在 OnTabSelectChanged 函数下面写入控制代码:

```
BOOL OnTabSelectChanged(int old,int newid)
    {
        //old,newid表示旧/新选中的以0开始的标签序号
        if (newid==0)
        {
            SetItemVisible(11001,TRUE);
            SetItemVisible(11002,FALSE);
        }
        else if(newid==1)
        {
            SetItemVisible(11001,FALSE);
```

```
            SetItemVisible(11002,TRUE);
        }
        return TRUE;
    }
```

效果图如下:



加入 List 控件

在第一个标签页对应的右边面板,即 ID11001 的 DLG 内插入一个自绘 list 控件为

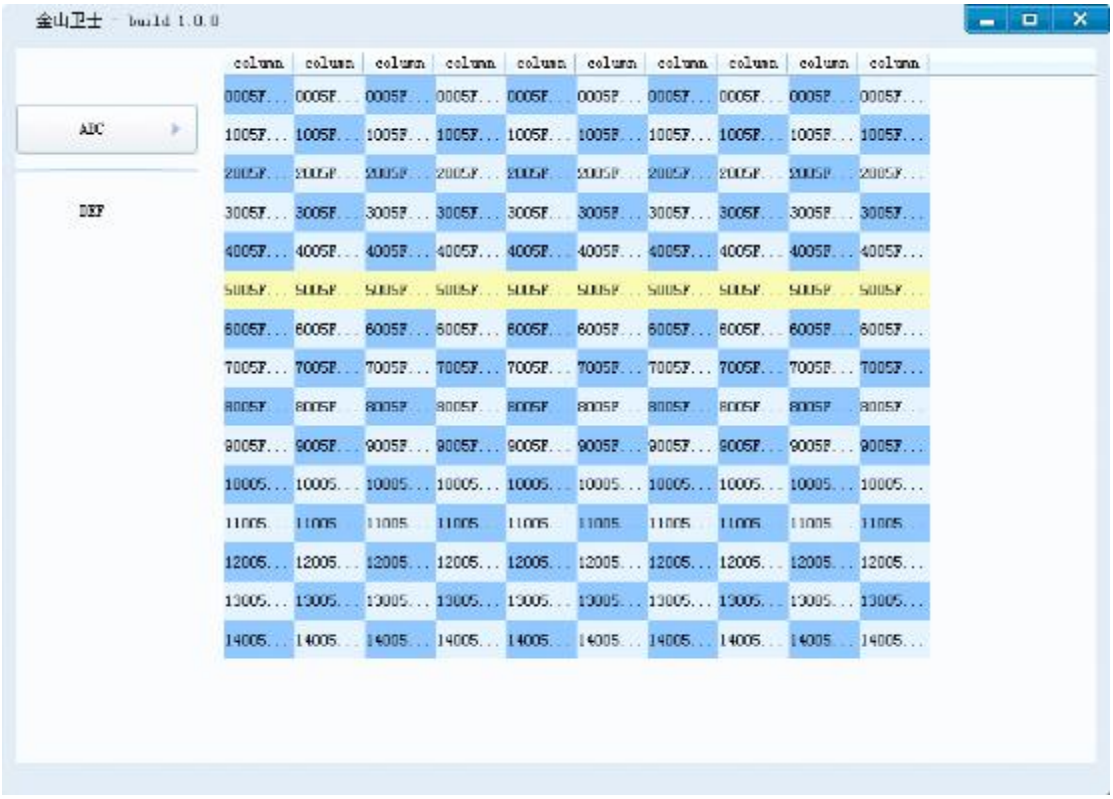
```
<realwnd id="2001" pos="0,0,-0,-0" >
```

在 OnInitDialog 函数里加入一些初始化操作

```
    m_list.Create( GetViewHWND(), NULL, NULL,
WS_VISIBLE|WS_CHILD|LVS_REPORT|LVS_SHOWSELALWAYS|LVS_SINGLESEL|LVS_OWNERDR
AWFIXED, 0, 2001, NULL);

    for (int i=0;i<10;i++)
    {
        m_list.AddColumn(_T("column"),i);
    }
```

使 Create 函数的控件 ID 参数与上面的 ID 属性值一致即可.

这样 list 控件就会随着父 dlg 的显示也显示出来.如图:



加入一些常用按钮与图标控件

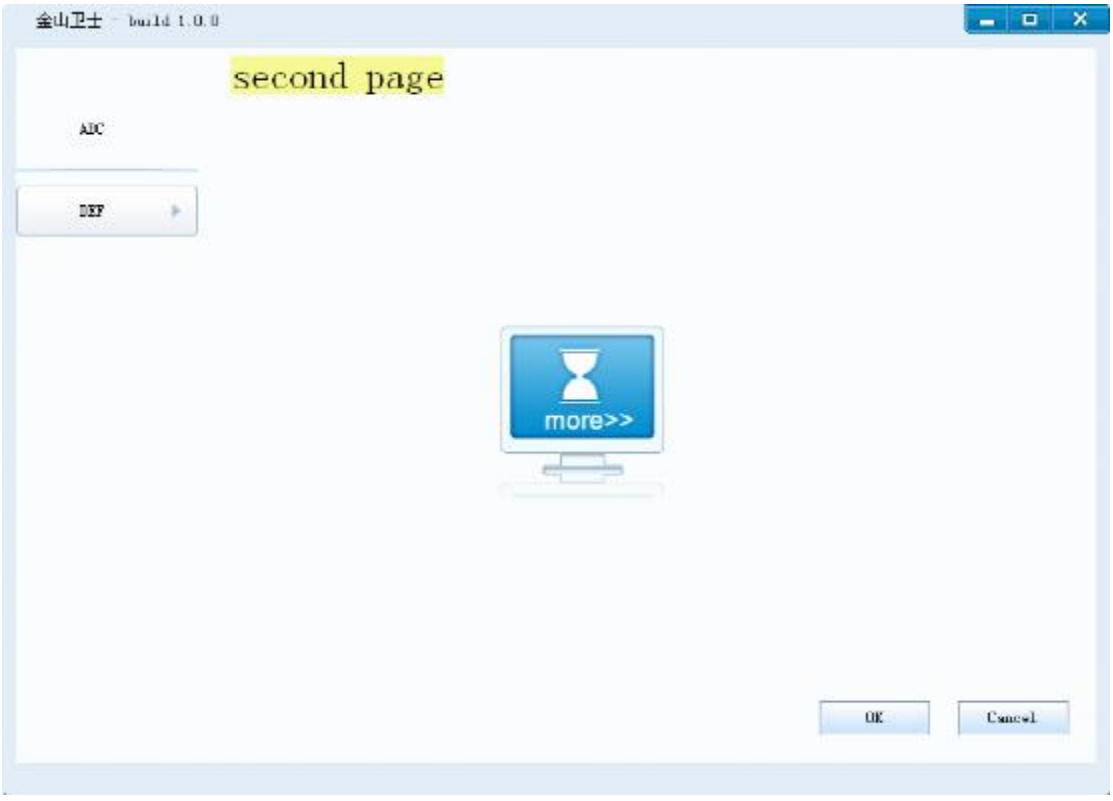
]xml 代码:

```
<text pos="5,5" class="tab2">second page</text>
<button id="60004" pos="-200,-45,-120,-20" class="normalbtn">OK</button>
<button id="60005" pos="-100,-45,-20,-20" class="normalbtn">Cancel</button>
<img pos="200,200" skin="sysopt_more" />                                    //图标
```

//normalbtn style
<class name=normalbtn skin=normalbtn font=0000 crtext=000000 textmode=25 cursor=hand x-margin=5/>

//normalbtn skin
<button name=normalbtn border=7D9EBC bg=FBFCFD bgup=FEFEFE bgdown=C6E2FD bguphover=FEFEFE bgdownhover=DBEDFE bguppush=C6E2FD bgdownpush=FEFEFE/>

效果如图:



金山开源的几个例子里面还没有发现使用了菜单及右键的地方.

也没发现使用菜单及右键的地方及代码.