【完美国际】界面主题

[策划需求 1](#_Toc277000965)

[数据文件格式要求 1](#_Toc277000966)

[主题更换策略 2](#_Toc277000967)

[重要的Callback 3](#_Toc277000968)

[控件资源更换策略 3](#_Toc277000969)

[附录 4](#_Toc277000970)

## 策划需求

允许玩家在登录场景及游戏内的设置界面中切换界面主题，被选中的主题将保存在设置数据中，玩家登录后按照设置载入相应的主题。

## 数据文件格式要求

为了方便制作，同时也为了兼容老的界面格式，支持主题的界面文件数据需要遵循一定规则。

* **DCF文件**

一套界面由多个界面描述文件(xml)组成，文件列表储存在dcf文件中，每个主题都对应一个dcf文件。

目前的实现中，dcf名称由CECUIManager::GetUIDcf()算出，与场景类型和主题id相关：例如游戏内场景的0号主题是ingame.dcf，1号主题是ingame-v1.dcf，2号是ingame-v2.dcf。

* **XML文件**

对于同一窗口的各个主题，需要制作不同的xml，但要求**窗口名必须相同**。

目前的实现中，要求**文件名也必须相同**，且必须储存在**interfaces\**或者**interfaces\Version\*\**格式的路径下。详见各个AUIManager::CreateDlgInstance()方法。

* **举例**

假设计划支持2套主题，游戏内场景共用到2个窗口，Win\_A和Win\_B。其中Win\_A在所有主题下都一样，美术为Win\_B 设计了2套主题，所以应该有3个xml文件。

|  |  |  |
| --- | --- | --- |
| 路径 | 窗口名 | 对应主题编号 |
| Interfaces\DlgA.xml | Win\_A | 所有 |
| Interfaces\DlgB.xml | Win\_B | 0号 |
| Interfaces\Version01\DlgB.xml | Win\_B | 1号 |

对应两套主题的dcf文件描述如下：

|  |  |  |
| --- | --- | --- |
|  | 0号主题 | 1号主题 |
| DCF文件名 | Ingame.dcf | Ingame-v1.dcf |
| DCF内容 | DlgA.xml -1 -1 0  DlgB.xml -1 -1 0 | DlgA.xml -1 -1 0  Version01\DlgB.xml -1 -1 0 |

## 主题更换策略

为了尽量不影响原有的界面代码实现，在主题更换的时候将保留当前窗口的数据，新载入的主题界面描述文件仅影响**控件外观及层次布局(x,y,z-order)**。

目前CECBaseUIMan::ChangeLayout()流程如下（可用阅读版式查看）：



## 重要的Callback

CECBaseUIManager逐个对窗口实例进行换肤操作的时候，会触发一系列callback。

* virtual bool CDlgTheme::OnChangeLayoutBegin()

调用时间：对当前窗口的主题更换操作开始之前。

参数：无

返回值：true（默认）表示可以对此窗口进行主题更换，false表示跳过此窗口的主题更换。

* virtual void CDlgTheme::OnChangeLayoutEnd(bool bAllDone)

调用时间：对当前窗口的主题更换操作结束之后。

参数：bAllDone为true表示此次操作成功，false表示操作失败并且主题已经恢复。

返回值：无

* virtual bool CDlgTheme::OnChangeLayout(PAUIOBJECT pMine, PAUIOBJECT pTheir)

调用时间：针对每个控件进行主题更换之前。

参数：pMine为当前窗口的控件。pTheir为新主题中对应的控件。

返回值：根据参数不同返回值功能也不同。

|  |  |  |  |
| --- | --- | --- | --- |
| pMine | pTheir | 调用含义 | 返回值含义 |
| NULL | !NULL | 新主题中增加了pTheir控件，一般是指增加了装饰型的控件，例如边框。 | true（默认）表示插入之后立即显示，false表示不显示。 |
| !NULL | NULL | 新主题中删除了pMine的同名控件，一般是指删除了装饰型控件，例如边框。 | true（默认）表示同意删除，false表示此控件仍然会被保留。 |
| !NULL | !NULL | 发现同名控件，且资源交换操作已经执行。**注意，此callback发生在交换资源之后！** | true（默认）表示同意交换，false表示需要恢复控件的资源，因此返回false将对pMine和pTheir进行第二次资源交换。 |

## 控件资源更换策略

资源更换基于AUI库提供的AUIObject::GetProperty()和AUIObject::SetProperty()方法，新旧两个窗口中同名控件的同类Property将被互相交换，交换两次相当于恢复原来的Property。因此控件名称非常重要，旧主题中和逻辑相关的控件，在新主题的窗口中不可以改名或者删除。否则发生找不到控件，或者控件被删除的错误。

目前的实现中，有部分包裹相关的界面，例如Skill, Bag, Inventory, Storage等，程序调用SetName()改变了控件在xml中设定的名称。这种情况需要在OnChangeLayoutBegin()中先恢复原有的控件名称，等主题更换完毕后在OnChangeLayoutEnd()中再次设置。

策略原则上仅更换影响外观和布局的Property。针对不同类型的AUIObject，有不同的更换策略，具体实现在CDlgTheme::SwitchResource()中。假如CDlgTheme::SwitchResource()在某个控件上的调用失败返回false，则当前窗口的主题更换操作终止，系统尝试恢复到原主题。在实际部署中不应该存在更换失败的情况，除非用户无法读取资源。

所有的图像资源相关的Property都会被更换，部分类型的Font也会被更换。Text类型的控件由于其内容大部分为程序填充，因此不会更换。具体需要查看CDlgTheme::SwitchResource()的实现。

目前的实现中，有如下几个属性需要注意：

* AUIStillImageButton和AUIRadioButton的Text会根据主题设置，因为按钮有时候用文字表现有时候用图片表现。
* 设置为CanMove=false的AUIDialog，也就是不可移动窗口，其定义在dcf中的位置会根据主题设置。因为不可移动的窗口位置仅和它初始设定有关，而可移动的窗口的位置可能已经被用户改变。

影响外观的Property原则上不应该在程序中再进行二次赋值，这样会造成编辑器中看到的外观和游戏中看到的不一致，例如文本框的SetAlign和按钮的SetText。但客观上存在程序控制这些数据的需求，因此应该将这些操作放入OnChangeLayout()或者OnChangeLayoutEnd()中，每次主题更换之后再重新进行设置。

目前的游戏中，原先调用SetAlign的代码大部分已经被删除，改为直接在xml中定义Align。而Shop, Produce等依靠程序填充按钮文本的窗口，已经在OnChangeLayoutEnd()进行了处理。

由于资源更换策略仅在新旧两个界面的**同名同类型控件**之间进行，所以运行时动态添加的控件无法自动进行资源更新，需要开发者在Callback中自行处理。建议将所有动态创建的操作放入Callback中，根据当前的界面表现生成相应的控件。

受此约束的影响，目前SubDialog控件的更换操作不支持出错后回滚（回滚仅仅为了方便测试，在正式发布版本中不应该存在错误的皮肤）。DlgBBS中动态添加的按钮也未作处理。BBS, Hint, MsgBox的外观需要在各自对应的界面模板中设置，已经创建的则无法更改。

## 附录