# 完美国际界面内存优化

版本

2014-1-3

2014-1-17修改

2014-3-25 修改

### 当前引擎支持(Angelica 2.0)

1. A2DSpriteTextures \* A2DSprite:: GetTextures()提供了对资源的引用计数，即int A2DSpriteTextures:: m\_nUserRef。
2. 通过A2DSpriteTextures::IncUserRef() 和 A2DSpriteTextures:: DecUserRef() 增减引用数。
3. 通过A2DSpriteBuffer \* A2DSpriteTextures::GetSpriteBuffer() 查询当前是否对象是否加载。
4. 通过A2DSpriteTextures ::UnloadTextures()卸载已加载资源。
5. 通过A2DSpriteTextures:: Init((A3DDevice\* pA3DDevice, const char\* szSpriteMap, int nWidth, int nHeight, A3DCOLOR colorKey)加载已卸载资源。
6. 资源引用计数与资源优化时加载卸载之间独立。由AUInterface界面库和游戏客户端维护引用和资源的优化加载卸载。
7. 在A2DSprite初始化时，自动增加引用。
8. 在A2DSprite释放时，自动减少引用（引擎未处理，2014.1.7后版本已修正）。

### 当前AUInterface界面库中的支持

1. AUIDialog:: SetOptimizeResourceLoad(true)将设置AUIDialog内部优化标识m\_bOptimizeResourceLoad为true
2. AUIDialog::Show(false)及AUIObject::Show(false)时，检测到m\_bOptimizeResourceLoad为true时，将减少资源的引用计数，如果引用计数为空，则删除资源。同时调用各子控件的Show(false)并对其进行优化
3. AUIDialog::Show(true)及AUIObject::Show(true)时，检测到m\_bOptimizeResourceLoad为true时，将增加资源的引用计数，如果资源为空，则加载资源。同时对各子控件做同样处理。
4. AUIDialog及AUIObject内部均有变量m\_nOptimizeResourceState，为1表示资源未优化，为0表示被优化过。

### 问题一（受Show的影响）

AUIDialog \* pDlg;

AUIObject \*pObj = pDlg->GetDlgItem(“XXX”);

pDlg->Show(true); // 所有资源被加载

pObj->Show(false); // 控件资源被优化

pDlg->Show(false); // 对话框资源被优化，控件资源状态保持被优化

pDlg->Show(true); // 所有资源被加载、pObj不可见但资源未被优化

### 问题二（受SetProperty的影响）

1. 所有SetProperty的调用都不考虑资源优化，如SetProperty(“Image File”, &prop) ，会直接减少对原资源的引用并增加对新资源的引用。
2. 资源优化期间调用SetProperty，导致AUIDialog:: m\_nOptimizeResourceState、AUIObject:: m\_nOptimizeResourceState 与实际情况不符，最终导致资源的引用计数不正确。
3. 具体地，首先，调用SetProperty直接对原资源引用减1，实际上在优化时已经减过了，后果是资源可能被提前释放，导致其它控件在需要时发生错误
4. 其次，调用SetProperty还会对新资源引用加1，当恢复优化状态时，会对新资源引用再加1，导致资源不会被释放，即使没人再持有此资源的引用时也得不到释放

### 问题一的解决方案

修改界面库，需保证优化AUIObject资源时，考虑到其可见状态。

1. 方法一

所有AUIObject的Show()的实现中，删除对优化代码的调用，即完全受父对话框控制。此方法改动简单、理解简单，但当控件不显示时，资源无法被优化。如果这种情况发生较少，可采用。

1. 方法二

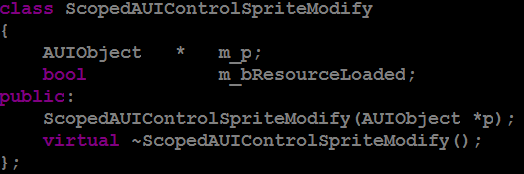
AUIDialog::Show(true)时，对不可见的AUIObject不加载资源；AUIObject的Show(true)时，如果父亲对话框不可见，则不加载资源。采用这种方法，还需要修改AUIDialog:: OptimizeSpriteResource，提供强行加载所有资源功能，在更换AUIDialog界面皮肤前使用

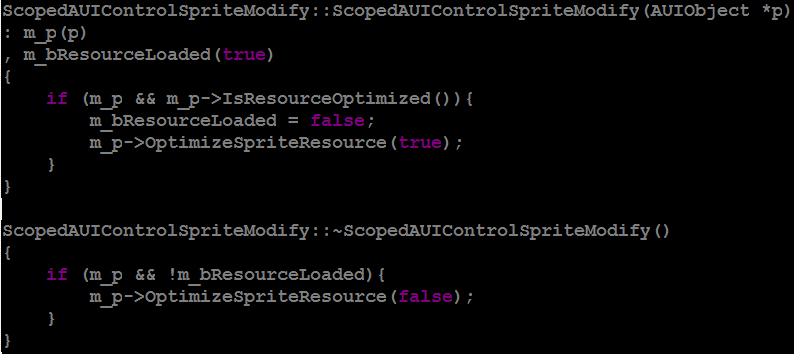
我们采用方法一

### 问题二的解决方案

客户端增加了ScopedDialogSpriteModify和ScopedAUIControlSpriteModify类，在构造函数中加载对话框及控件的所有资源，当SetProperty调用后，再恢复对象的优化状态。

ScopedAUIControlSpriteModify 代码如下：

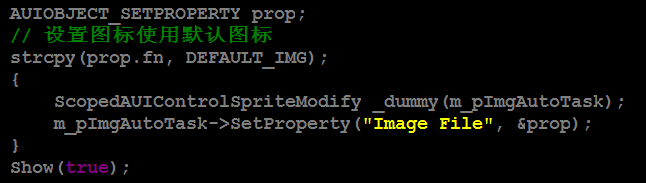




其中，IsResourceOptimized()为在AUIDialog及AUIObject中新增的方法：



ScopedAUIControlSpriteModify类的示例用法如下：



### 正确使用ScopedDialogSpriteModify、ScopedAUIControlSpriteModify

上面两个类应与SetProperty调用前后直接绑定，在设置完成后即恢复资源优化情况。看如下针对控件的优化反例（对ScopedDialogSpriteModify也类似）：

{

PAUIIMAGEPICTURE pImage=dynamic\_cast<PAUIIMAGEPICTURE>(GetDlgItem(“XXX”));

AUIOBJECT\_SETPROPERTY prop;

strcpy(prop.fn, DEFAULT\_IMG);

ScopedAUIControlSpriteModify \_dummy(pImage);

pImage ->SetProperty(“Image File”, &prop);

pImage->Show(true);

}

假设进入上述代码域前，pImage已经被优化，则有如下过程：

1. 当ScopedAUIControlSpriteModify构造函数被调用时，资源被加载进入、资源优化状态清除
2. 调用SetProperty后，正确减少对原资源的引用，增加对新资源的引用
3. 再调用Show(true)，由于当前资源未优化，Show(true)对资源优化不产生效果
4. 当ScopedAUIControlSpriteModify析构函数被调用时，资源被优化。这将导致资源在使用的时候却未加载。当新资源为pImage独占却被优化掉时，pImage->Render中会出现assert；如果此新资源为共享对象，则当其它对象持有新资源引用时pImage->Render不会失败。

### 图片文件路径应满足的规则

试验中发现，A2DSprite对文件路径的大小写不敏感，但是对”/”和”\”敏感：

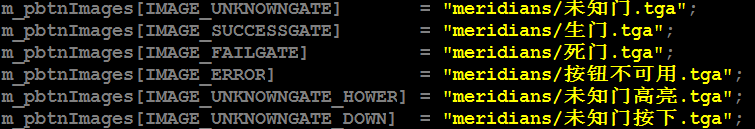
"meridians\未知门.tga" (1)

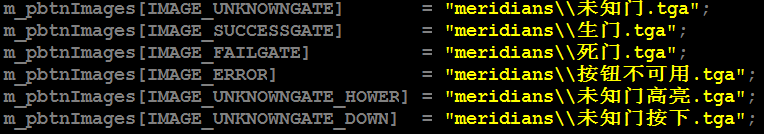
"MERDIANS\未知门.tga" (2)

"meridians/未知门.tga" (3)

(1)和(2)在A2DSprite底层对应同一个A2DSpriteTextures对象，而它们和(3)是不同的。因此，为避免内存浪费，应使用同一规则在代码中引用路径。由于界面文件引用图片路径使用的是”\”，包括ecm等文件中也是使用了”\”，因此，客户端需要被迫也使用这种方式。

基于这个原因，DlgMeridiansImpact.cpp中的下述代码，被转换：

 （转换前）

 （转换后）

客户端中其它地方也已经做了修改。**后续代码引用文件路径，应采用相同规则**。