**FLASH & SCALEFROM 安装与运行**

1. ***概要：***

这个部分是包含一般的在游戏里使用Flash 和ScaleForm的步骤.

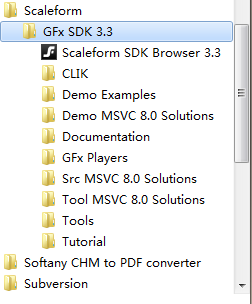
1. ***版本&资源需求：***
2. 在我们游戏（Mars3 Or Mars2）使用的是 GFX 3.3 版本。GFX最后版本的是4.2版本（ADOBE 目前已经停止对GFX的更新）。
3. 工具你可以用最新的FLASH都可以，但是FLASH 的版本不能低于FLASH 8（Action Script 2完全支持是在这版本之后）。
4. FLASH 的破解文件 在百度上随便搜索FLASH CS4 的破解补丁，就会有amtlib.dll的破解补丁，用这个替换原来的就可以的。

这里注意一点就是：破解后不能选择跟新，一旦更新，后果难料。打了这个破解补丁后，每次你点开FLASH，它提示输入激活码界面时，你直接点退出按钮，耐心等待就可以看到FLASH 可以打开了。

1. GFX 的源代码可以在<http://pan.baidu.com/s/1gfJu1RP>  下载ScaleFrom.rar文件。
2. 我在开始用FLASH的时候，网上搜到一个FLASH 功能学习使用网站，简短实用，我觉得不错：<http://www.51zxw.net/list.aspx?cid=11>
3. 在查阅FLASH 里的AS接口的使用和说明，可以打开网站：<http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/2/index.html>，这里包含了所有的FLASH 接口信息。

请注意：GFX 不是完美支持FLASH 的，所以一些功能在GFX 上面是不能用的。具体哪些支持哪些不能用的，可以查阅文档：**Flash Support OverView**(在ScaleForm目录下)

1. 在网上下载一个可以管理AS代码的软件：**FlashDevelopment**，这款软件对Action Script 的支持很好，但是它对中文的支持很糟糕，在AS里尽量不要用中文来注释。
2. ***安装及使用方法：***
3. 下载FLASH 8 以上的版本（我推荐FLASH CS4），安装FLASH,并且破解补丁也打好。
4. 下载GFX 源代码， 安装Scalefrom。
5. 安装完Scalefrom 后再开始菜单里会有Scaleform的标签：



1. 打开Scaleform SDK Browser 3.3,这个里面包含了所有的scaleform 的文档和实例。我们现在先打开里面的：Getting Started With Gfx,根据这个文档里的内容来设置FLASH.(这里我就偷懒了，不详细写出来具体步骤，反正这个文档非常详细)，

需要提醒的是：在设置Flash的包含文件的时候，CLIK的文件包含路径为我们工程里的CLIK路径而不是Scaleform的CLIK文件。

1. 等Scalefrom 和 FLASH 都配置完了，我们可以打开MARS2 OR MARS3 的工程目录。在我们工程里，我们把用scalefrom 做的UI的分成3个部分：

* Scaleform 的CLIK 工程；相对路径：(工程目录)\UnrealEngine3\Development\Flash\CLIK；这个路径用来设置FLASH里编译的包含路劲。
* FLASH 源文件：FLA, SWF, Action Script 文件， 每一个FLA 文件都会对应于一个Action Script 文件，他们的对应关系就像：MarsUI\_Formula.fla,这个fla包含的Action Script 文件就是ActionScript\Formula
* UPK 部分：这部分是我们游戏实际上需要使用的文件，他们是由FLASH SWF 文件导入到UNREAL 里得到的文件。

在导入到UNREAL 里面的时候，可以使用命令行 gfximport,具体使用方法是在我们的(工程目录)\UnrealEngine3\MarsGame\ 目录下面创建一个bat文件，里面的内容是

start /b ..\Binaries\Win32\MarsGame.exe gfximport MarsUI\_BroadCast\Mars\_BroadCast.swf 这行代码里面的MarsUI\_BroadCast\Mars\_BroadCast.swf就是你要导入的文件，

用这行命令行要注意的就是如果这个文件是你第一次导入到Unreal 文件里面，你最好还是通过Unreal Editor 来做的，不然会出现导入的文件位置是Unreal 默认的位置Content\Gfx\

1. 到这里来说，我们游戏里面的大概结构应该是知道了，下面我会讲解一下GFX 在我们工程里的（代码层面）的具体使用方法

=========================================妖娆的分割线=========================================妖娆的分割线============================================================================

**工程细节**

1. ***概要：***

这个部分主要是为了说说我到目前为止对我们工程和GFX的了解，里面主要内容是我在用GFX的时候所遇到的问题及其解决方法。

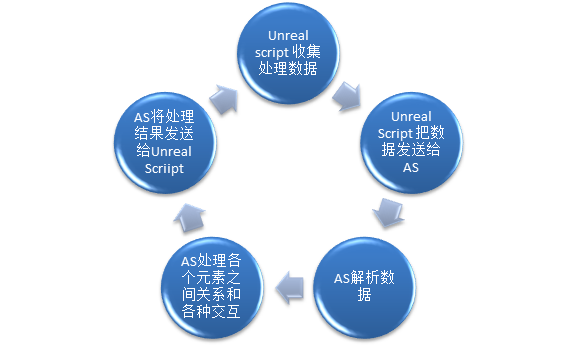
我以前有一个同时在做项目的时候是把GFX3.3完全整合到代码里的。虽然他们不是用的Unreal,但是我觉的这是很有必要的，因为只有完全整合了才能做出我们自己的漂亮的UI，就像DOTA2那样的UI，酷炫！

对于新人（对FLASH 了解不是特别足的）在读下面的内容我强烈介意先看一个ScaleForm的文档，尤其是：

* **CLIK**章节里的各种Demo(这是ScaleForm 自带的各种UI组件，要比野外搜素过来的强多了，在ScaleForm里的很多代码会直接调用CLIK里的函数，在某些时候这也是一个坑）
* **Getting Started with CLIK Buttons** （ScaleForm Browser 里面有）,这个可以让你了解和使用ScaleForm组件里最重要，使用最多，功能最为复杂，坑最多的组件：BUTTON。
* **ActionScript Extensions Reference** (ScaleForm Browser 里面有),这个文件里面讲了很多关于GFX对ScaleForm的基础类和回调函数的扩展，比如movie clip, text Field, mouse,等等，我到目前发现这里面的扩展只有render相关的没什么作用，第一个这个东西可以FLASH里做，第二个这个东西，我们工程里用的很少，第三个我们工程里集成的ScaleForm，只是封装后的集成，在编译代码的时候直接是link lib,没有源代码编译，所以综合来说这个我到现在觉得没什么用。（能打脸时提前告诉我一下，这样我也能学学的。）
* **3DI Overview**,可以先读这个文档来了解GFX 对3D UI的一个扩充。(这不是完美的扩充)

1. ***代码结构***
2. 我的理想
3. 我理想中的UI设计模型是

Unreal Script 只是负责数据的收集处理，然后将最终的数据发送给AS，由AS来解析数据，并且在整个页面里面的所有的元素之间控制，用户交互都是在AS层面执行，然后将最终的执行结果Unreal Script,这样做的原因其实很简单就是将属性（数据）和行为分开，保证程序的健壮性，可读性。优点多多。

简单的模式是不是？不过好像执行的不怎么好，因为所有的UI不是我一个人写的

1. 我理想中代码规范

* 如果Unreal Script 里需要调用AS里的函数，要在这个AS函数前面加上’UC\_’(不包含‘’)，
* 如果在AS里需要调用Unreal Script 里的函数需要在Unreal Script 的函数前面加上所有’AS\_’(不包含‘’)，
* 在所有的函数只要函数里面有逻辑代码就要加上注释，
* 所有的成员变量之前都要写上这个变量使用注释（成员变量前面没有注释，看的真的好累），尤其是那些机构体变量，枚举值

看很简单基础规范，但是执行不怎么好。最上面的两个作用是我可以读Unreal Script和Action Script 的代码不需要参阅另一半的代码，来解读这个函数在哪里用到的，是干什么用的。

   这些规范不涉及变量函数名称的命名什么的，说真的谁会记得那些用20张A4纸打印都不打印不完的规则。

1. 我理想中的Flash制作：

           所有的FLA 文件都有自己的数据来做debug，这数据是写在AS里面，这样在我打开一个FLA文件能够不用通过编译，导入，打开游戏（这是能让人逼疯的一件事情）就能看到最为直观的效果。

           套用一句话：理想的伟大在于现实的骨干。

1. 简要说明

在我们现在的工程里（代码部分），UI的主要分为三个部分：

**Unreal Native Code & Unreal Script & Action Script 2.0.**

在ADOBE 把GFX 整合到Unreal 里面的时候它也提供了很多在Unreal 层面的元件（元件的定义查看附录）控制代码。但是我们工程里没有用，因为如果用这些控制代码去在Unreal Script 上来生成创建movie clip或者其他的元件，会让代码非常乱。而且维护性，可读性都非常差。（不管怎么说，AS的可读性我觉得比UC的强多了）

下面我就分别简略说明一下：

1. *Unreal Native Code*

在我们这个工程里只有GFX的lib文件，并且这个lib文件是没有DEBUGINFO的，而在UnrealEngine3\Development\Src\GFX下面的.cpp文件是Unreal 和GFX交互的文件，其中我看的比较多的就是：GFxUIEngine.cpp.这个文件里包含了所有Unreal 到Gfx的实现接口。比如说loadimage, externalcall, advance,等等。 我看到先人改动比较多的就是GFxUIEngine.cpp里的input函数。

如果说你想要知道GFX内部应该怎么实现的，不知Action Script 怎么实现的？为什么GfxObject可以当作一个容器的？元件是怎么控制的？GfxEvent 到底在GFX怎么运作的，你可以打开开头让你下载的GFX的工程软件，自己查看，就我目前看到的，很多在功能包括枚举值什么的在两个处是完全一样的。很多功能只是在Unreal里面加了一个壳子，作为Unreal数据结构的转换用的。

在这里我推荐你查看ScaleForm里面的文档：**Integration Tutorial,**这个文档会按部告诉你如何把Gfx插入DX中，在这些步骤中你会对GFX的流程与类的组成和使用有一个基础性的了解。

在这个部分因我们游戏需求而需要我改的几乎没有（我们游戏的UI设计的非常简单）所以我主动修改的地方没有多少，也没有多少陷阱让我头疼的.按照标准的流程做我们现在的UI，这个地方让你可以不必去关心。

（我想在以后的时间里我会在这里添加所有我对ScaleForm 工程的新认知的）

1. *Unreal Script*

我们工程里的Unreal Script 就我个人感觉就比较乱了的。

在我们游戏里有两种UI实现方式一个是通过UIScene来实现的还有一个是通过GFX来实现的

UIScene其实是需要放弃的，但是现在我们游戏还有一些这些代码来显示UI，这其实是不对的。UIScene的代码很乱，因为他的数据处理，页面显示全部糅杂在一起，如果这个UI不是你做的，你想找一个信息很难。我没用UIScene来做过UI，不知道具体UIScene的工作流程是什么，但是我通过看代码如果让我来做，我做的也不会好太多。

      GFX做的UI现在是我们游戏里主要的UI实现方式。所以我下面主要讨论的就是关于GFX的代码

1. *Action Script 2.0*

在我工程里最早期使用GFX的人应该是一个专业的FLASH程序员，因为我看到他原来用FLASH来做UI的所有的AS（Action Script）代码都是自己写的一些功能组件（也许是从其他地方copy过来的）；因为是自己写的组件，所以比GFX自带的CLIK要简单的很多，功能上没有CLIK完善，而且跟ScaleForm 不是完全的兼容很多特性它里面是没有支持代码的。而且这个对于以后招一个另外一个用FLASH 做UI的程序员来说不算友好。所以我在新做的UI都是用CLIK里提供的基础组件来搭建。CLIK易用但是出问题了就是比较头疼的问题。我会后面说一说主要在用CLIK时候我遇到头疼的问题。

1. Unreal Script里的UI代码
2. Action Script 里的UI代码

==========================================妖娆的分割线=================================================妖娆的分割线======================================================================

**附录**

1. FLASH 中各种专业名词解释

当初ADOBY 在开发FLASH 的时候把FLASH的定位为可操作的舞台剧（我个人理解就是小型的GAME ENGINE，不过跟不上潮流了），在FLASH 里所有的物件在术语上都是套用的舞台剧然后做一个变通。

他们都把第一层的画布叫做舞台。然后在舞台上的所有物件叫做元件（如果你看GFX 代码的话会发现它是被命名为Character（对，就是舞台的人物）的），时间轴上的Action Script代码在有些文档里也叫做剧本。