**ActionScript 3.0标准与规范（Beta版）**

**1．命名规范（Naming Convention）**

**1.1．命名规则概要（General naming guidelines）**

1.1.1．必须使用英文字符、字符串、数字，不得以数字开头；

suggest：myVariable、myVariable1；

don’t suggest：1myVariable；

1.1.2．英文命名必须简明扼要；

suggest：myVariable、myFunc；

don’t suggest：myV、myF；

1.1.3．常规英文单词缩写必须大写，便于区分识别；

suggest：HTTP、URL；

don’t suggest：http、Url；

1.1.4．推荐驼峰命名法；

suggest：myVariable、myParameter；

don’t suggest：myvariable、myparameter；

**注：其他命名规范细则无特殊说明均自动遵循命名规则概要**

**1.2．变量/属性/参数/常量命名规范（variable/property/parameters/ constant naming convention）**

1.2.1．所有变量必须声明命名空间（public、internal、protected、private）；

suggest：public var myVariable:Object；

don’t suggest：var myVariable；

1.2.2．所有变量必须声明其类型；

suggest：public var myVariable:Object；

don’t suggest：public var myVariable；

1.2.3．命名一组对象时应使用复述形式；

suggest：public var myVariables:Array；

don’t suggest：public var myVariable:Array；

1.2.4．布尔类型变量不得以否定形式命名；

suggest：public var isFound:Boolean；

don’t suggest：public var isNotFound:Boolean；

1.2.5．循环嵌套中用含义清晰的变量定义循环嵌套变量；

suggest：

for(var row:int=0; row <getRows();row ++)

{

for(var col:int=0;col<getCols();col++)

{

//statements

}

}

don’t suggest：

for(var i:int=0;i<getRows();i++)

{

for(var j:int=0;j<getCols();j++)

{

//statements

}

}

1.2.6．以合适的后缀来命名UI组件类型的变量；

suggest：progressSlider、playBtn；

don’t suggest：sliderProgress、btnPlay；

1.2.7．非常量首字母小写，大小字母混排（即驼峰命名法）；

suggest：myUsername、myPassword；

don’t suggest：myusername、mypassword；

1.2.8．参数名可在末尾增加一个后缀“\_“以区分变量与属性；

suggest：

function set password(password\_:Number):void

{

This.\_password=password\_;

}

；

1.2.9．常量必须全英文大写；

suggest：var const MOUSE\_DOWN:String；

don’t suggest：var const mouse\_down:String；

1.2.10．同一类别常量前缀必须相同；

suggest：

var const MOUSE\_DOWN:String；

var const MOUSE\_UP:String；

don’t suggest：

var const MOUSE\_DOWN:String；

var const UP\_MOUSE:String；

1.2.11．private空间下变量/属性必须在首字母前加下划线“\_”；

suggest：private var \_password:Number；

don’t suggest：private var password:Number；

**1.3．函数命名规范（function naming convention）**

1.3.1．所有函数必须声明命名空间（public、internal、protected、private）；

suggest：

public function init():void

{

//statement;

}

don’t suggest：

function init():void

{

//statement;

}

1.3.2．所有函数必须声明返回类型；

suggest：

public public function checkPassword():Boolean

{

//statement;

}

don’t suggest：

public public function checkPassword()

{

//statement;

}

1.3.3．所有函数名必须小写英文字母开头，大小写混排；

suggest：

public public function init():void

{

//statement;

}

don’t suggest：

public public function Init():void

{

//statement;

}

1.3.4．所有函数名必须简明扼要，以动词开头；

suggest：

public public function checkPassword():Boolean

{

//statement;

}

don’t suggest：

public public function passwordCheck()

{

//statement;

}

1.3.5．函数名可忽略类名或对象名关键字，避免重复申明；

suggest：

public class Font

{

public public function getFamily():String

{

//statement;

}

}

don’t suggest：

public class Font

{

public public function getFontFamily():String

{

//statement;

}

}

1.3.6．单例类必须统一通过名为getInstance()的静态函数返回其唯一值；

1.3.7．事件处理函数以on加事件类型命名，例如：onClick；

suggest：

myBtn.addEventListener(MouseEvent.CLICK,onClick);

don’t suggest：

myBtn.addEventListener(MouseEvent.CLICK,click);

**1.4．事件命名规范（event naming convention）**

1.4.1．事件均以静态常量定义；

**1.5．接口命名规范（interface naming convention）**

1.5.1．接口以大写字母I开头的名词或形容词，I后面第一个字母均大写；

**1.6．类命名规范（class naming convention）**

1.6.1．名词单数形式开头，首字母大写，大小写混排

1.6.2．表示一组事物的类应用复数形式；

1.6.3．单例类必须统一通过名为getInstance()的静态函数返回其唯一值，建议用包外类实现；

**1.7．包命名规范（package naming convention）**

1.7.1．包的命名均用小写字母或小写字母与数字混排；

1.7.2．包名用名词或动名词；

**1.8．命名空间规范（namespace naming convention）**

1.8.1．命名空间应以小写字母开头，“\_“为分隔符；

**1.9．文件命名规范（filen naming convention）**

1.9.1．所有文件必须采用规范标准的后缀，文件名不要出现中文字符；

**2．文本段落格式规范（Text Statements Format Convention）**

**2.1．包导入（import package）**

2.1.1．只导入需要的类，避免使用通配符；

suggest：

import flash.display.Sprite;

don’t’ suggest：

import flash.display.\*;

2.1.2．自定义类和内置类分开导入，按包路径匹配率高低分块导入；

**2.2．声明（declaration）**

2.2.1．类/接口声明：

a．文档注释；

b．类/接口语句；

c．静态变量顺序；

d．实例变量顺序；

e．构造函数；

f．函数；

g．包外类；

2.2.2．变量的声明应在变量定义的位置（for循环例外）；

**2.3．留白（white space）**

适当的的使用空格可以提高代码的可读性

2.3.1．使用空格的通用规则；

a．操作符，冒号前后均有空格；

b．逗号，分号，AS保留字后留一空格；

2.3.2．逻辑单元以空行分割；

**2.4．括号**

采用Adobe标准风格，括号另起一行；

**2.5．注释**

简明扼要，长话短说

2.5.1．变量/属性的注释；

变量的上一行“//”标注

2.5.2．局部变量的注释

行末“//”注释

2.5.2．函数的注释：

2.5.3．类的注释；

2.5.4．接口的注释；

2.5.5．包的注释；

2.5.6．包外类的注释；

**2.6．编码**

所有文本文件的字符编码采用UTF-8;

**2.7．体积**

单个类文件控制在300-500行为佳，最大不要超过2k行，单行最大不要超过128栏/ 字符（128字符时代码编写和打印的平衡点）；

**2.8．缩进**

使用tab而非space，使用缩进以表明嵌套层次

**3．发开环境规范（develop environment）**

3.1．Flash cs4+FlashDevelop（FD编辑代码，FL编译swf）;

3.2．Flash cs4(swc)+FlashBuilder4（FB绑定swc直接编译swf）;

3.3．FlashBuilder4+Catalyst;

**3．调试（debug）**

3.1．trace信息；

3.2．debug模式及断点调试；

3.3．FF浏览器下，console.log输出面板;

3.4．Alcon客户端，捕获调试信息，不限web或local；

**4．性能优化基本原则（performance）**

**4.1．CPU性能优化**

4.1.1．利用mm.cfg查看AVM运行状况，便于协助改进算法

4.1.2．改进算法

无论对于那一种程序，好的算法总是非常重要的，而且能够极大地提高程序性能，所以任何性能的优化第一步就是从算法或者说程序逻辑的优化开始，检查自己的程序是否有多余的运算，是否在没有必要的时候做了无用功，往往从这些方面就能找到那些导致性能低下的地方

4.1.3．优化细节代码

1．尽量少用Timer、setInterval、enterframe，频率尽可能低，不用及时注销

2．flash常见帧频为12fps、24fps、25fps、30fps、50pfs，通常以25fps为标准

3．用 var obj:Object = {}; 要比 var obj:Object = new Object();要好；

4．用var arr:Array = []; 要比 var arr:Array = new Array(); 要好；

5．用for (var i:int=0, len=arr.length; i<len; i++) 要比 for (var i:int=0; i<arr.length; i++) 要好；

6．如果不是为了保存颜色值请不要适用uint这个类型，他的速度比起 int 要慢多了；

7．Array的遍历要比Object或者Dictionary的枚举要快得多；

8．if (myObj != null) 要比 if (myObj) 的速度要快；

9．for (var i:\* in myObj) 比 for (var i:String in myObj) 要快；

10．Dictionary当 weakkey设置为 true 的时候要比 false 慢；

11．var myText:String = “a” + “b” + “c”;var myText2:String = [ "a", "b", "c" ].join(”");

在JavaScript里面在IE下后者要更快，但是在AS里面，前者更快；

12．在循环体内声明变量和在循环体外声明变量其实速度上不会有太大的区别；

4.1.4．所有不在舞台显示列表的MC帧动画需从显示列表中移除，性能消耗对比removeChild<visible<alpha=0<alpha=1<alpha半透明，不用的MC帧动画尽量做到stop每一帧

4.1.5．重绘

使用Flash Player的Debugger版本，打开显示重绘区域，将迅速定位到舞台上有那些地方被重绘，找出没有显示任何东西却不断重绘的地方，Flash Player很笨，不会说你把一个DisplayObject的visible设置成false就放弃重绘那个显示对象。所以请保证你的 MovieClip在visible=false的时候为停止状态。有一点很有意思，假设两个现实物体存在 hitTest = true 这样的关系，那么重绘的区域的面积很有可能 > 两者的面积总和；

4.1.6．cacheAsBitmap慎用

一般情况下除非确定这个显示对象不可能发生变化那么用用也无妨，不过更推荐自己手动的用BitmapData将该对象Draw一遍，然后让这个对象彻底消失。否则每次的变动都是巨大的性能消耗

4.1.7．记得销毁无用的对象

对于非常驻的对象使用完之后记得消除其引用，防止出现内存溢出的问题，往往要做到这一点需要有一个良好的编程习惯

4.1.8．以空间换时间

听起来挺虚，实则很简单，说白了就是以内存换CPU，例如将不变动的值进行保存，免去下次需要此数据的时候进行再次计算，虽然原理很简单，但是有的时候却很容易疏忽掉，而这个往往就造成你的算法效率低下的问题。

4.1.9．清除冗余的代码

有些代码可能你的程序一辈子也不会执行到，请把这些没有用的代码或者对象清理掉，否则内存会被偷偷的蚕食掉

4.1.10．正式上线的产品必须注销掉所有调试信息

4.1.11．MC、Sprite嵌套层级尽可能少

4.1.12．swf文件在web页面引用wmode参数设为“window”CPU消耗最低；

4.1.13．权衡程序的结构

程序的架构也非常重要，良好的结构会带来性能和程序健壮性的提升，但是有的时候又是相互矛盾的，例如代码写得过于健壮，反而会影响性能，这个地方需要开发者自己去权衡

**4.2．内存性能优化**

4.2.1．被删除对象在外部的所有引用一定要被删除干净才能被系统当成垃圾回收处理掉；

4.2.2．父对象内部的子对象被外部其他对象引用了，会导致此子对象不会被删除，子对象不会被删除又会导致了父对象不会被删除；

4.2.3．如果一个对象中引用了外部对象，当自己被删除或者不需要使用此引用对象时，一定要记得把此对象的引用设置为null;

4.2.4．本对象删除不了的原因不一定是自己被引用了，也有可能是自己的孩子被外部引用了，孩子删不掉导致父亲也删不掉；

4.2.5．除了引用需要删除外，系统组件或者全局工具、管理类如果提供了卸载方法的就一定要调用删除内部对象，否则有可能会造成内存泄露和性能损失；

4.2.6．父对象立刻被删除了不代表子对象就会被删除或立刻被删除，可能会在后期被系统自动删除或第二次移除操作时被删除；

4.2.7．如果父对象remove了子对象后没有清除对子对象的引用，子对象一样是不能被删除的，父对象也不能被删除；

4.2.8．注册的事件如果没有被移除不影响自定义的强行回收机制，但有可能会影响正常的回收机制，所以最好是做到注册的事件监听器都要记得移除干净。

4.2.9．父对象被删除了不代表其余子对象都删除了，找到一种状态的泄露代码不等于其他状态就没有泄露了，要各模块各状态逐个进行测试分析，直到测试任何状态下都能删除整个对象为止。

内存泄露举例：

1. 引用泄露：对子对象的引用，外部对本对象或子对象的引用都需要置null；

2. 系统类泄露：使用了系统类而忘记做删除操作了，如BindingUtils.bindSetter()，ChangeWatcher.watch()函数时候完毕后需要调用ChangeWatcher.unwatch()函数来清除引用 ，否则使用此函数的对象将不会被删除；

类似的还有MUSIC，VIDEO，IMAGE，TIMER，EVENT，BINDING等。

3. 效果泄露：当对组件应用效果Effect的时候，当本对象本删除时需要把本对象和子对象上的Effect动画停止掉，然后把Effect的target对象置null; 如果不停止掉动画直接把 Effect置null将不能正常移除对象。

4. SWF泄露：要完全删除一个SWF要调用它的unload()方法并且把对象置null;

5. 图片泄露：当Image对象使用完毕后要把source置null;(为测试)；

6. 声音、视频泄露: 当不需要一个音乐或视频是需要停止音乐，删除对象，引用置null;

内存泄露解决方法：

1. 在组件的REMOVED\_FROM\_STAGE事件回掉中做垃圾处理操作（移除所有对外引用（不管是VO还是组件的都需要删除），删除监听器，调用系统类的清除方法）

先remove再置null, 确保被remove或者removeAll后的对象在外部的引用全部释放干净;

2. 利用Flex的性能优化工具Profile来对项目进程进行监控，可知道历史创建过哪些对象，目前有哪些对象没有被删除，创建的数量，占用的内存比例和用量，创建过程等信息；

总结：关键还是要做好清除工作，自己设置的引用自己要记得删除，自己用过的系统类要记得做好回收处理工作。 以上问题解决的好的话不需要自定义强制回收器也有可能被系统正常的自动回收掉。

**5．版本控制规范（version manage）**

5.1．SVN版本控制

5.2．线上产品每次修改重新上线需做修改说明

5.3．每次更新版本上线严格以时间戳年、月、日、时、分来标注

**6．UI规范**

6.1．.Fla源文件库里的每一个元件都需规范命名并建相应文件夹归类

6.2．元件尽量做到少嵌套层级、少循环动画

**7．安全机制**