ActionScript 3.0
经典教程

A闪
芒果 搜集
A闪工作室 隆重推出
欢迎访问我们的博客
http://hi.baidu.com/暗黑侧卫

前言

大家好！欢迎大家来看我们的教程。笔者经常在网上查看一些资料，其中发现一些比较不错的 FLASH 网站和博客。有些教程写的确实很精彩，觉得爱不释手。一直很想把这些优秀的教程总结一下，但由于时间的关系这个愿望一直没有实现。现在机会来了，觉得将这些经典的教程收录到一本电子书当中，一来是做一个学习总结，二来是想发给大家，为大家学习 ActionScript3.0 提供一些帮助。这就是制作本书的一个初衷。希望大家喜欢这本电子书。

提到 FLASH，可以说在网络中占有的地位越来越重要，很多网站都离不开 FLASH。从最早的 FLASH 广告，到后来的 FLASH 网站，再到后来的 FLASH 交互式网站，FLASH 在网络中的作用越来越强大。随着 FLASH 版本的不断更新，功能也越来越强大，运行效率也越来越高。这些发展不禁另我们为之震惊。现在很多人都开始学习 ActionScript3.0，制作这样的书籍也是迫在眉睫，所以我们将在这本书中引入大量的经典教程供大家学习！

好了！多余的话就不多说了！我们下面来介绍一些学习 ActionScript3.0 的一些网站吧！

网站

闪吧：http://www.flash8.net/

天地会：http://www.9ria.com/

A 客网：http://www.51as.com/

FLASH 3D 研究所：http://www.flab3d.com/

淘沙网：http://www.taoshaw.com/taoshaw/default.asp

博客

A 闪工作室：http://hi.baidu.com/暗黑侧卫

大頭的博客：http://www.cnblogs.com/cwin5/

小 S 吧：http://www.xiaos8.com/default.asp

粉色男孩：http://hi.baidu.com/fsnhf

ND_WEB：http://hi.baidu.com/dn%5Fweb

蜡笔的博客：http://hi.baidu.com/蜡笔工作室

PO PO BLOG：http://hi.baidu.com/zoujun1314

Lite3’ blog：http://www.lite3.cn/

BO BO’ bolg：http://www.flex2007.cn/

达达’ blog：http://www.asflex.cn/

Cbm’ land：http://www.cbmland.com/

黑羽翔天：http://www.kingda.org/

苹果树下：http://azure.cn/

更多博客可以查看以上博客中的友情链接！教程中出现是实例可到指定的博客中去下载！欢迎大家访问这些网站！

特别推荐一套AS3 的视频教程：http://hi.baidu.com/%B0%B5%BA%DA%B2%E0%CE%C0/blog/category/actionscript3%2E0%CA%D3%C6%B5%BD%CC%B3%CC

AS3 响应右键事件

作者：DN_Web

文章来源：http://hi.baidu.com/dn%5Fweb/blog/item/72dc9f25c2172b3b8644f924.html

转载自:http://www.duzengqiang.com/blog/post/355.html

From: http://www.roading.net/blog/post_207.html
flash对右键的按下没有对应的响应事件,但是有的时候需要用到这个事件.
在以前可以根据Key的侦听事件中Key.isDown(2)来判断按下了右键.
但是在as3 中Key.isDown函数因为安全原因被删掉了.那么在as3 中怎么获取右键的按下事件呢?

as3 的Mouse和MouseEvent也没有右键的事件.同样在Keyboard和KeyboardEvent中没有右键相关的事件.
但是,如果想想ContextMenu,就有办法了,在flash中,右键响应只关联着右键菜单.所以在ContextMenu里面有右键的响应事件ContextMenuEvent.MENU_Select(在as2 里面是ContextMenu.onSelect).

as3 中文帮助里面对menuSelect的解释:在用户首次生成上下文菜单但尚未显示上下文菜单内容时调度。 这将允许您的程序在显示菜单之前修改上下文菜单项集。 用户通过右键单击指针设备来生成上下文菜单。

所以可以用menuSelect来作为右键的响应事件.在这个事件执行后就会显示右键菜单.在这之前,可以执行响应函数,修改对应的菜单内容.

使用方法:
as3:
```
myMenu = new ContextMenu();
myMenu.hideBuiltInItems();
myMenu.addEventListener(ContextMenuEvent.MENU_Select,menuSelect);
this.contextMenu = myMenu;

function menuSelect(e:ContextMenuEvent)
{
trace("menuSelect");
}
```

as2:
```
var my_cm:ContextMenu = new ContextMenu();
function menuHandler(obj:Object, menu:ContextMenu) {
trace('menuSelect');
}
my_cm.onSelect = menuHandler;
this.menu = my_cm;
```

AS3 操作 XML

作者：A 客网

文章来源：http://www.51as.com/as3/AS3caozuoXML_32.html

简单说说 AS3.0 中对于 XML 支持的不同吧：

.AS2.0 对 XML 的支持勉勉强强，将就着可以用。而 AS3.0 中对 XML 的支持是全方位的，极其强大和灵活的。

AS2.0 对 XML 的支持不是内建的(build-in)，也并非基于 ECMAScript for XML(E4X)标准。而 AS3.0 中对 XML 的支持符合 E4X 标准，它的设计有三个优点：

1. 简易。包括操作和可读性。你会发现 AS3.0 中对于 XML 的操作犹如对一个普通 Object 对象一样浅显易懂。语句非常浅白流畅。
2. 连续性。其各个功能的设计和 AS3.0 其余的部分思想一致，易于理解。
3. 熟悉。操作符和操作逻辑对我们来说都相当熟悉易用。

在 AS2.0 时代，为了解决这部分的问题

效率。

效率包括两方面，开发效率，和代码执行效率。开发效率的论述见上。AS3.0 对于 XML 的执行效率远远高过没有内建 XML 支持的 AS2.0。

XML 的输入

在 AS2.0 时代，在代码行中输入 XML 代码是一种痛苦。如果不是从文件中读取，那么我们就要忍受一长串挤在一块儿的字符串。

而在 AS3.0 中，太简单了。直接按照 XML 的内容输即可，想换行就换行，想 Tab 就 Tab，就一个字，爽。

新建一个 fla,选中第一帧，F9 打开动作面板，输入如下代码：

//例 1

```
var kingdaXML:XML =
<tutorial>
<item id='1'>
<level>2</level>
<title> First touch of Flash 9</title>
</item>
<item id='2'>
<level>3</level>
<title> Binding Classes</title>
</item>
<item id='3'>
<level>4</level>
<title>Document Class</title>
</item>
</tutorial>
trace (kingdaXML.item[1].level); //output:3
```

//例 2

```
var kS:String = "<root><txt>this is a test</txt></root>";
var kXML:XML = new XML(kS);
trace (kXML.txt); //output:this is a test;
```

例 1 中注意到没，直接写 XML 内容在后面，想换行就换行，想 tab 就 tab，多爽。不想 AS2.0 中写 string 时，换个行就不行了。

写完这一句后，我们所写出的类似于 string 的形式立刻就被 Flash 理解成了 XML 对象了，所以我们马上就可以用"."操作符来访问相应的属性。本例中访问了第 2 个 item 节点的 level 值。

这么简便直观的访问方式是不是比 AS2.0 中那千遍一律的 childNodes 要好得多？

不过要注意，最后可以加";"结束。但我为了 XML 的视觉美观没有加。这个没有关系，编译时不会考虑这一点。

事实上只要你喜欢，AS1.0, 2.0, 3.0 中语句结束都可以不加";"号。但是这并不是一个好的编程习惯，更不符合严谨的自我语法要求。因此我建议，除了 XML 可以不加外，其余的都应该加，呵呵。

例 2 展示了如何将一个包含了 XML 内容的字符串转换成 XML 对象。用的是 XML 的构造函数转换的。

AS3 更有趣的是，可以使用已有的变量来直接构造 XML，带来方便的编程特性。如下例。

```
var rootNodeName :String = "site";
var subNodeName :String = "orgin";
var subNodeContent :String = "Kingda's Blog";
var attributeName :String = "url"
var attributeValue :String = "http://www.kingda.org";
var extXML:XML =
<{rootNodeName} {attributeName}={attributeValue}>
<{subNodeName}>{subNodeContent}</{subNodeName}>
</{rootNodeName}>;
trace (extXML.toString());
/*output:
<site url="http://www.kingda.org">
<orgin>Kingda's Blog</orgin>
</site>
*/
```

要点就是要把变量用"{}"括起来，并且设置属性时不要再加引号了。

XML 的外部读取

包括读取外部 xml 文件，和通过 URL 读取 xml。AS3.0 中不像 2.0 那样集成了一个 load()。

AS3.0 在架构上就设计了所有与外部打交道的都由 URLrequest 对象来进行，数据都由 URLloader 对象来接受。这个我们会在下一部分教程详细讲解。这一次只要知道这样的架构设计是深思熟虑，且简洁优美的即可。

```
var myXML:XML = new XML();
//初始化 XML 地址，可以是本地的"xxx.xml"，也可以是如下的 URL 地址。
var XML_URL:String = "http://www.kingda.org/blog/index.xml"; //我的 Blog RSS Feed
var myXMLURL:URLRequest = new URLRequest(XML_URL);
var myLoader:URLLoader = new URLLoader(myXMLURL);
//添加装载完成侦听器,
//Event.COMPLETE 的值是"complete",直接用此字符串也可以。
myLoader.addEventListener(Event.COMPLETE, xmlLoaded);
function xmlLoaded(evtObj:Event) {
myXML = XML(myLoader.data);
trace("数据装载完成.");
trace (myXML);
}
```

XML 的操作

1.查询

```
//显示 level 为 4 的节点的 title 值

trace (kingdaXML.item.(level == 4).title);

//output：Document Class

//显示 level>2 的节点的 title 值，本处结果大于 1，所以是一个 XML Array。

trace (kingdaXML.item.(level > 2).title);

/*output:

<title>Binding Classes</title>

<title>Document Class</title>

*/

//使用属性用@开头即可。真方便。

trace (kingdaXML.item.(level > 2).@id);

//output:23

//这儿要注意，实际上是 2,3。一个 Array.

//也可以用属性来做判断
```

```
trace (kingdaXML.item.(@id > 1).title);
```

2.添加或者修改属性
方便的不能再方便，直接写即可。爽翻天啊。

```
//把 id == 1 的节点 level 值改为 2
kingdaXML.item.(@id==1).level = 2;
//把 id==1 的节点添加一个属性  page
kingdaXML.item.(@id==1).page = 100;
trace (kingdaXML.item.(@id==1));
```

3.按某条件插入节点

```
var newNode1:XML = <item id='2.5'><level>0</level><title>None</title></item>
var newNode2:XML = <item id='1.5'><level>0</level><title>None</title></item>
//把 newNode1 插入到 id==2 的节点后面
kingdaXML = kingdaXML.insertChildAfter(kingdaXML.item.(@id==2), newNode1);
//把 newNode1 插入到 id==2 的节点前面
kingdaXML = kingdaXML.insertChildBefore(kingdaXML.item.(@id==2), newNode2);
trace (kingdaXML);
```

XML 的高级操作

常用的操作上面已经介绍的很清楚了。高级操作则是留给对 XML 应用更深的兄弟们。

几点注意：
1.在 AS3.0 中， XML 类的 ignoreWhitespace 默认为 true。
2.AS3.0 支持对 comments 的直接操作。但默认:

```
XML.ignoreComments = false;
var kingdaXML:XML =
<item>
<!-- comment 1-->
<!-- comment 2-->
</item>;
trace(kingdaXML.toXMLString());

//默认为 true 时，不会显示 comment 的
```

访问 comment 用

```
trace(kingdaXML.comments()[1].toXMLString());
```

3.XML 支持克隆。

使用 copy()可以得到一份现有 XML 的值拷贝。

var kingdaCopy:XML = kingdaXML.copy();

对 kingdaCopy 操作就不会影响 kingdaXML 对象了。

4.极有用的 descendants 函数返回一个 XMLList 对象，包括所有的子节点。
设 ignoreComments = false;和 ignoreProcessingInstructions ＝ false 后，连 comments 和 process instructions 也会包含在这个 XMLList 对象中。

运用示例如下：

```
XML.ignoreComments = false;
var xml:XML =
<body>
<!-- comment -->
text1
<a>
<b>text2</b>
</a>
</body>;
trace(xml.descendants("*").length()); // 5
trace(xml.descendants("*")[0]); // // <!-- comment -->
trace(xml.descendants("*")[1].toXMLString()); // text1
trace(xml.descendants("a").toXMLString()); // <a><b>text2</b></a>
trace(xml.descendants("b").toXMLString()); // <b>text2</b>
```

还有太多的 XML 有用操作功能了(如对 namespace 的操作)。用到时再去翻参考书吧。
以上的介绍可以满足绝大部分运用了。

对了 AS2.0 已有的 XML 类，在 3.0 中变成了 XMLDocument 类，使用方法不变。便于 AS2.0 程序移植。其余不推荐

AS3 工程中的 Loading 的应用

作者：A 客网

文章来源：http://www.51as.com/as3/AS3gongchengzhongdeLoadingdeyingyong_239.html

今天又来介绍 Loading...(-_-!!!...好像整天都介绍 Loading..希望没误导各位...)

首先..由于 AS 工程没有帧..所以不能用常用的方法来做 Loading..

这里介绍的方法使用元标签 Frame(应该是元标签吧?还是叫元数据标签)

网上对 Frame 的介绍是...使用指定的类替换文档类...

并把其它的东西都丢到了该类的第二帧...

概念就不多说了...英文好的可以看看这个文章...http://www.bit-101.com/blog/?p=946

Preloader 类

```
package {

 import flash.display.DisplayObject;

 import flash.display.MovieClip;

 import flash.display.StageScaleMode;

 import flash.events.Event;

 import flash.events.ProgressEvent;

 import flash.text.TextField;

 import flash.text.TextFormat;

 import flash.utils.getDefinitionByName;

 /**

 * 加载类,由于原内容会放到此类的第二帧~所以需要使用 MovieClip;

 * @author L4cd.Net

 */

 public class Preloader extends MovieClip {

  [Embed(source="l4cd_48_48.jpg")] ;
```

AS3 研究 TextField 心得

作者：A 客网

文章来源：http://www.51as.com/as3/AS3yanjiuTextFieldxinde_111.html

这两天自己做了textinput和textarea组件，众所周知，这两种组件的内部其实都是一个原始的textfield。这里把自己的一些研究心得记录如下

首先要明白几种长度单位：

1、磅：pt(point)，这是一种绝对长度单位，为 1/72 英寸,等于 0.3527mm

2、像素：px，这是一种相对长度单位，譬如，WONDOWS的用户所使用的分辨率一般是 96 像素/英寸。而MAC的用户所使用的分辨率一般是 72 像素/英寸。

以下讨论文本框TextField相关：

1、字号

也就是fontsize，flash中单位是磅（office中也时），可以使用textHeight和textWidth来获取文本的像素高度和宽度，由于各种字体自身的差别，不是相同字号的字体所需要的文本框高度都一致。譬如Arial字体就比宋体所需要的文本框高度高。

2、文本框高度

对于单行文本框，给定一个高度和一种字体，如何确定所需要的最大的字号呢？

有一个简便方法就是，新建一个autosize的文本框，设置其字号并判断其高度来得到最大字号。

```
function getFontSizeByTxtHeight(h:Number, font:String):Number
{
  //给定单行文本框高度 h，字体 font 获取最大字号
  var testtxt:TextField = new TextField();
  testtxt.type = TextFieldType.INPUT;
  testtxt.autoSize = TextFieldAutoSize.LEFT;
  for (var size:Number=Math.round(h);size>0; size-=0.5)
  {
    var tf:TextFormat = new TextFormat();
    tf.leading = 0;
    tf.font = font;
    tf.size = size;
    testtxt.defaultTextFormat = tf;
    testtxt.text = "test 单行文本";
    //trace(size, testtxt.height);
    if (testtxt.height <= h)
    {
      break;
    }
  }
  return size;
}
```

AS3 中 Matrix 类详解

作者：A 闪

文章来源：http://hi.baidu.com/%B0%B5%BA%DA%B2%E0%CE%C0/blog/item/18a1676427c654faf7365484.html

在 AS3 中 Matrix 可以说有着很重要的作用，为什么这么说呢，因为它可以控制我们的元件进行伸缩或倾斜，这是 AS2 所做不到的，或者说很难做到。这里提一句，在 AS2 中如果我们想倾斜一个对象就需要将这个对象切成三角形然后进行变换，比较麻烦，在早先的 PV3D 中我们可以看到这样的实例。这里我们不做过多介绍！既然 AS3 为我们准备了这么好的工具为什么我们不用呢？浪费了这么好的资源。打开 FLASH 的帮助面板查看一下吧！你会发现帮助不大，貌似里面说的都不能看懂，写的很复杂，实际上并没有这么复杂，而且使用起来非常的方便，经过笔者研究，发现 Matrix 类实际上只有 2 点要值得注意的地方，其他地方简单至极。好了下面我们就来看看这个令人摸不着头脑的 Matrix 类吧！

首先，我们要看看 Matrix 类的构造器函数，其实我是想让大家看看他的 6 个属性，，这 6 个属性对与我们非常的重要，他决定这我们的最终效果是什么样子的，会产生什么样的倾斜，都由这 6 个属性控制。看一下！

```
Matrix(a:Number = 1, b:Number = 0, c:Number = 0, d:Number = 1, tx:Number = 0, ty:Number = 0);
```

可以看到，6 个参数，或者说 6 个属性分别是 a,b,c,d,tx,ty。那么这 6 个属性分别是做什么的呢？这里我们不按照它的顺序来讲解，而是从最简单的开始。这样大家读起来就方便理解一些。在这 6 个属性中最简单的 2 个是最后 2 个。那么它们起到的作用是什么呢？答案是“位置坐标”。一看到这 4 个字大家应该这个时候都明白了，实际上这 2 个属性和我们平常用的 x,y 属性的作用是完全一样的，他们的作用就是设置对象的横纵坐标！我想到这里大家应该对这点都再清楚不过了！x,y 属性是我们最常见的属性了，也是使用率最多的属性，几乎每创建一个对象我们就要设置一下 x,y 属性。所以这里我们不在做过多的解释了！唯一要注意的一点是 tx,ty 这 2 个属性的默认值是 0,0。也就是说，当我们没有对这 2 个参数进行设置的时候，系统会默认的将对象的位置放置到原点坐标，也就是(0,0)点。这一点大家注意一下即可。

好了！最简单的 2 个属性我们说完了，还剩下 4 个，那么我们还是遵循由易到难的规则来讲解。这次我们要提到的属性是 a,d。有人会问为什么不是 a 和 b 呢？实际上，在这 6 个属性中 a 和 d 是一组。所以我们这里要将这 2 个属性放到一起去讲解。那么这 2 个属性是做什么用的呢？我再写另外 2 个和 Matrix 类无关的 2 个属性来比较一下！scaleX 和 scaleY。有人会说，你将这 2 个属性放到这里做什么？首先，你要知道这 2 个属性的功能—控制缩放比例。知道了它们的功能实际上就知道了 a 和 d 的功能。不错，a 和 d 的功能就是控制缩放比例。当他们的值为 1 的时候，就表示按照原有大小进行显示，如果为 2，就表示放大一倍。非常好理解，可以说简单至极，这 4 个属性我们可以用其他的 4 个属性来间接理解，功能上没有任何区别，只是放到的类中不同罢了。相信大家读到这里会感觉到，原来 Matrix 类没有什么神秘的，不过是一个普普通通的类而已。

最后，我们还剩下 2 个属性，也是最难理解，最难明白的两个属性。我将用很大的篇幅来介绍这 2 个属性，细致的来分析它们的用法及使用注意事项。

b,c 默认值为 0。看到这里我们应该知道，默认是没有倾斜的。说到这里，可能很多人多这两个属性的功能还不清楚，在此再强调一遍，这两个属性是控制显示对象倾斜的。所谓倾斜，举一个最简单的例子，就是把一个长方形编程一个平行四边形。这样解释就直观多了。好了！那么他们怎么样来控制倾斜的度数呢？这才是我们的重点，就是如何控制倾斜的角度。说到角度，先抛开 b,c 不说，在 FLASH 当中我们不可以直接使用度数，为什么呢？不知道，没有原因，ADOBE 公司就是这么

设计的，没办法。那么我们用什么来表示度数呢？我们可以使用弧度单位来表示度数。这里要提一句“360°=2π 弧度”。这个公式大家一定要记住，以后会经常用到，那么比如说我们要表示 30°，那应该怎么写呢？就是写成下面这样：

30*π/180

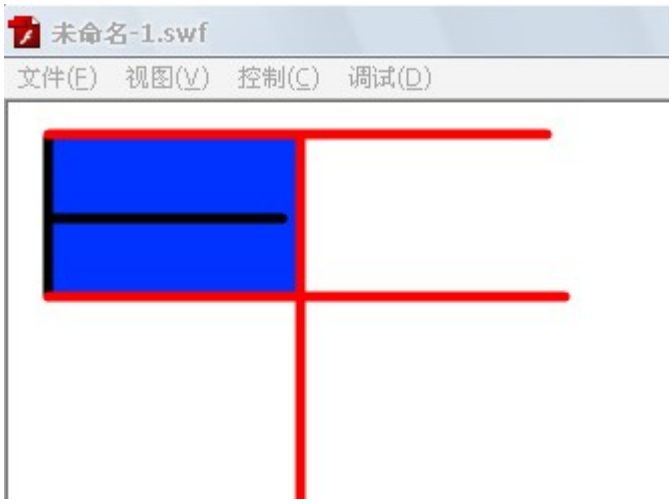
在 AS3 中的写法就是这样的：

30*Math.PI/180

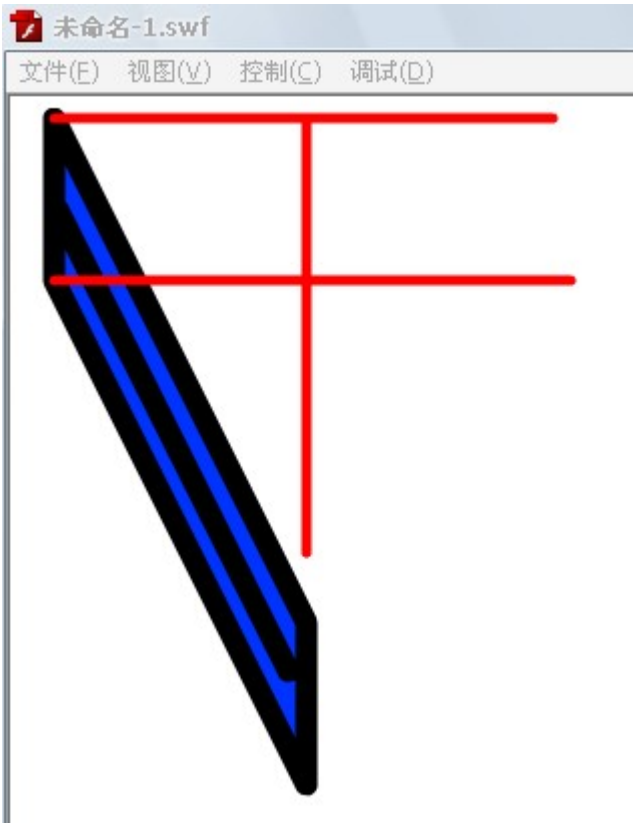
这样写，在 AS3 中就表示 30° 了。好了！我们先来看 b 吧。这个 b 表示显示对象沿 Y 轴倾斜。我们来看一段代码：

```
var ju:Matrix = new Matrix(1,2,0,1,0,0);
a_mc.transform.matrix = ju;
```

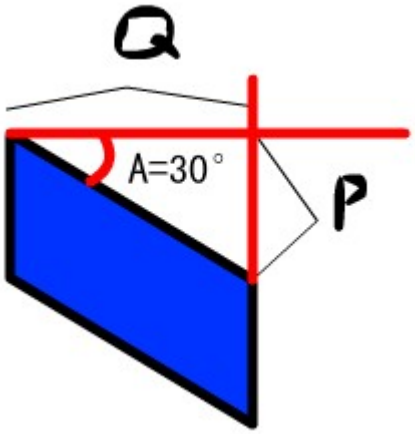
这段代码的 b 属性我设置成了 2，那么也就是说元件会沿 Y 轴向下倾斜。我们来看原图片，这是没有执行语句前的图像。



这里，我为了标识出蓝色方块原来的位置，我用红色的线来记录一下。注意：红色的线只起到一个标尺的作用。好了！下面我们来执行一下语句。



我们看到，原来的矩形已经严重倾斜了！这就是倾斜的效果。那么我们要注意的是什么呢？或者说我想给大家的只是点是什么呢？请大家注意，矩形虽然倾斜了，但是他的宽没有变化，同时，他的左右两条边的长度也没有发生变化。唯一变化的就是它的高和上下两条边。这一点大家要值得注意。有人会说了！这只是向右侧倾斜，那如果我想向左侧倾斜怎么办呢？简单，我们只需要将 b 的值改为负数即可！这里就不在做演示了！大家可以自行实验。那么下面要讲解的就是如何去控制倾斜的角度！这里我们先来看一个图！



我们来看一下这个图，现在假如我想要这个蓝色的矩形倾斜 30°，也就是说∠A 的度数为 30°。那么这个时候我们怎么表示属性 b 呢？我们要使用到三角函数中的正切。所为正切就是这样的。

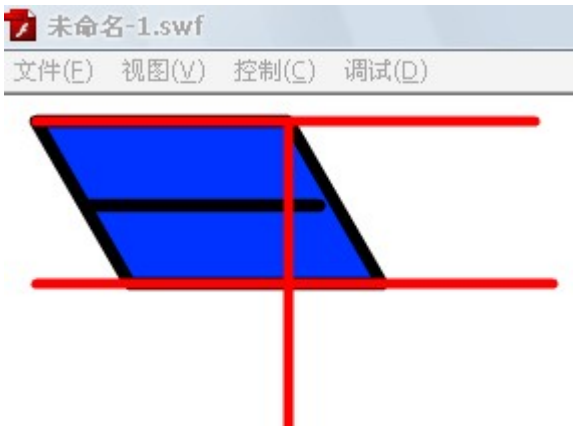
正切（tan）:角 α 的对边比上邻边

这是比较正确且规范的说法。那么好了！这个正切值就表示了属性 b。在这个实例中，正切值实际上就是 P/Q。得到的就是正切值。那么，我们不可能在没一个程序中自己手动的去计算我们这个正切值。所以这里又要用到 AS3 中的 Math 类。在这个工具类中有专门用于计算正切值的方法。我们来实际的写一个程序来看一下。代码如下：

```
var ju:Matrix = new Matrix(1,Math.tan(30*Math.PI/180),0,1,0,0);
a_mc.transform.matrix = ju;
```

这里我们就将倾斜角度设置为了 30° 。我们测试一下影片就可以看到效果了！这就是 b 属性的用法，那么 c 属性又说做什么的呢？和 b 属性的功能相同，只不过这次是沿 X 轴进行倾斜，我们同样倾斜 30° 来看一下。

```
var ju:Matrix = new Matrix(1,0,Math.tan(30*Math.PI/180),1,0,0);
a_mc.transform.matrix = ju;
```



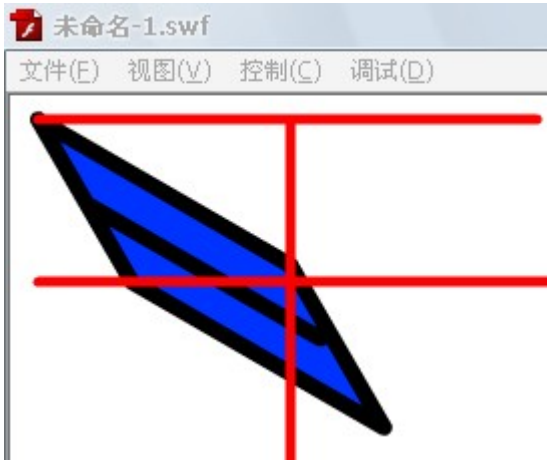
不过这次要注意的是，虽然倾斜了，但是矩形的上下两条边并没有改变长度，且矩形的高也没有改变。

这就是向右倾斜。如果想要想做倾斜也非常的简单，只要将 c 的属性改成负数就可以了。

最后，我们来看一种比较常用的情况，就是 X 轴 Y 周同时倾斜。代码如下，我们还是倾斜 30° 。

```
var ju:Matrix = new Matrix(1,Math.tan(30*Math.PI/180),Math.tan(30*Math.PI/180),1,0,0);
a_mc.transform.matrix = ju;
```

运行效果如下：



这次我们就不能保证宽高都没有变化了！ 应为 X 轴和 Y 轴都发生了倾斜，所以都改变了！

好了！ 关于 Matrix 类我们就说这么多，具体的使用技巧还需要大家实践得来！ 886 各位！ 今天就写到这里！ 以后有什么新的应用还会写出来给大家参考的！

as3 中的进制转换方法

作者：zoujun1314

文章来源：http://hi.baidu.com/zoujun1314/blog/item/733f04dd4c47b93e5982dd4f.html

http://www.laaan.cn/?p=137 一、toString(进制) 方法，如： 1. trace(new uint(51).toString(2)); // 输出二进制: 110011 trace(new uint(25).toString(16)); // 输出十六进制: 19 2. var quantity:Number = 164; trace(quantity.toString(16)); // 输出十六进制: a4 3. var pink:Color = new ColorTransform( ); pink.rgb = 0xF612AB; trace(pink.rgb.toString(16)); // 输出十六进制: f612ab 二、parseInt 函数 parseInt(expression:String, [radix:Number]) : Number 将字符串转换为整数。如果参数中指定的字符串不能转换为数字，则此函数返回 NaN。以 0x 开头的字符串被解释为十六进制数字。以 0 开头的整数或指定基数为 8 的整数被解释为八进制数字。有效整数前面的空白将被忽略，有效整数后面的非数字字符也将被忽略。返回 Number – 一个数字或 NaN（非数字）。 trace(parseInt(”110011″, 2)); // Displays: 51 trace(parseInt(”19″, 16)); // Displays: 25 trace(parseInt(”17″, 10)); // Displays: 17 trace(parseInt(”A9FC9C”)); // NaN trace(parseInt(”017″, 10)); // Displays: 17 (not 15) //虽然”017″是以 0 开头（八进制），但是它后面有指定一个基数 10，所以把”017″ 当成是十进制

as3 中的数字类型的转换

作者：A 客网

文章来源：http://www.51as.com/as3/as3zhongdeshuzileixingdezhuanhuan_35.html

as3.0 中的不同数字类型转换

类型：二进制、八进制、十进制、十六进制

不管如何,你在 AS 中设置一个数值，最后的结果都是返回一个十进制,例如:

```
//建立一个 Color 对像
var pink:ColorTransform = new ColorTransform( );
// 设置 RGB 值为一个十六进制值
pink.rgb = 0xF612AB;
// 输出的值是一个十进制: 16126635
trace(pink.rgb);
```

如果想输出一个不同类型的值，可以用以下几种方法：

一、toString(进制) 方法，例：

```
1.
trace(new uint(51).toString(2)); // 输出二进制: 110011
trace(new uint(25).toString(16)); // 输出十六进制: 19
2.
var quantity:Number = 164;
trace(quantity.toString(16)); // 输出十六进制: a4
3.
var pink:Color = new ColorTransform( );
pink.rgb = 0xF612AB;
trace(pink.rgb.toString(16)); // 输出十六进制: f612ab
```

二、parseInt 函数

parseInt(expression:String, [radix:Number]) : Number

将字符串转换为整数。如果参数中指定的字符串不能转换为数字，则此函数返回 NaN。以 0x 开头的字符串被解释为十六进制数字。以 0 开头的整数或指定基数为 8 的整数被解释为八进制数字。有效整数前面的空白将被忽略，有效整数后面的非数字字符也将被忽略。

返回

Number - 一个数字或 NaN（非数字）。

```
trace(parseInt("110011", 2)); // Displays: 51
```

```
trace(parseInt("19", 16)); // Displays: 25
trace(parseInt("17", 10)); // Displays: 17
trace(parseInt("A9FC9C")); // NaN

trace(parseInt("017", 10)); // Displays: 17 (not 15)
//虽然"017"是以 0 开头（八进制），但是它后面有指定一个基数 10，所以把"017" 当成是十进制
```

AS3 中的位操作

作者：zoujun1314

文章来源：http://hi.baidu.com/zoujun1314/blog/item/a1b511265680e21e8a82a1c0.html

介绍 AS3 中常见的位运算技巧。

在 AS3 中位操作是非常快的，这里列出一些可以加快某些计算速度的代码片段集合。我不会解释什么是位运算符，也不会解释怎么使用他们，只能告诉大家如果想清楚其中的原理先认真学一下 2 进制.

左位移几就相当于乘以 2 的几次方（ Left bit shifting to multiply by any power of two ）

大约快了 300%

```
x = x * 2;
x = x * 64;
//相当于：
x = x << 1;
x = x << 6;
```

右位移几就相当于除以 2 的几次方（Right bit shifting to divide by any power of two）

大约快了 350%

```
x = x / 2;
x = x / 64;
//相当于：
x = x >> 1;
x = x >> 6;
```

Number 到 integer(整数)转换

在 AS3 中使用 int(x)快了 10% 。尽管如此位操作版本在 AS2 中工作的更好

```
x = int(1.232)
//相当于：
x = 1.232 >> 0;
```

提取颜色组成成分

不完全是个技巧，是正常的方法 (Not really a trick, but the regular way of extracting values using bit masking and shifting.)

```
//24bit
var color:uint = 0x336699;
var r:uint = color >> 16;
var g:uint = color >> 8 & 0xFF;
var b:uint = color & 0xFF;
//32bit
var color:uint = 0xff336699;
var a:uint = color >>> 24;
var r:uint = color >>> 16 & 0xFF;
```

```
var g:uint = color >>> 8 & 0xFF;
var b:uint = color & 0xFF;
```

合并颜色组成成分

替换值到正确位置并组合他们（‘Shift up’ the values into the correct position and combine them.）

```
//24bit
var r:uint = 0x33;
var g:uint = 0x66;
var b:uint = 0x99;
var color:uint = r << 16 | g << 8 | b;
//32bit
var a:uint = 0xff;
var r:uint = 0x33;
var g:uint = 0x66;
var b:uint = 0x99;
var color:uint = a << 24 | r << 16 | g << 8 | b;
```

使用异或运算交换整数而不需要用临时变量

这里快了 20%

```
var t:int = a;
a = b;
b = t;
//相当于:
a ^= b;
b ^= a;
a ^= b;
```

自增/自减(Increment/decrement)

这个比以前的慢不少，但却是个模糊你代码的好方法；-）

```
i = -~i; // i++
i = ~-i; // i--
```

取反（Sign flipping using NOT or XOR）

快了 300%!

```
i = -i;
//相当于：
i = ~i + 1;
//或者
i = (i ^ -1) + 1;
```

使用 bitwise AND 快速取模 （Fast modulo operation using bitwise AND）

如果除数是 2 的次方，取模操作可以这样做：

模数= 分子 & (除数 - 1);

这里大约快了 600%
```
x = 131 % 4;
//相当于：
x = 131 & (4 - 1);
```

检查是否为偶数（Check if an integer is even/uneven using bitwise AND）

这里快了 600%
```
isEven = (i % 2) == 0;
//相当于：
isEven = (i & 1) == 0;
```

绝对值

忘记 Math.abs()吧 (Forget Math.abs() for time critical code.)
version 1 比 Math.abs() 快了 2500% ，version 2 居然比 version 1 又快了 20% ！
```
//version 1
i = x < 0 ? -x : x;
//version 2
i = (x ^ (x >> 31)) - (x >> 31);
```

Comparing two integers for equal sign
This is 35% faster.
```
eqSign = a * b > 0;
//equals:
eqSign = a ^ b >= 0;
```

快速颜色转换从 R5G5B5 到 R8G8B8 象素格式用移位
```
R8 = (R5 << 3) | (R5 >> 2)
G8 = (R5 << 3) | (R5 >> 2)
B8 = (R5 << 3) | (R5 >> 2)
```

AS3 中鼠标事件与鼠标坐标

作者：A 客网

文章来源：http://www.51as.com/as3/AS3zhongshubiaoshijianyushubiaozuobiao_107.html

鼠标事件（MouseEvent）和鼠标位置（AS3 鼠标坐标总结）是 RIA 中最重要的人机交互途径。最近在做一个动态产品展示的系统 ProdutShow 的时候才发现自己对鼠标事件的了解有多么肤浅。现在 ProductShow 已经做完了，这里把在使用鼠标事件时要注意的问题总结一下：

1 鼠标事件分为 **MOUSE_OVER, MOUSE_MOVE, MOUSE_DOWN, MOUSE_UP, MOUSE_OUT, MOUSE_WHEEL 和 MOUSE_LEAVE**。其中前六个事件都来自 flash.events.MouseEvent 类，最后一个 MOUSE_LEAVE 却是来自 flash.events.Event，在导入类包的时候一定要注意这个问题，因为我在这点上就花了很长时间调试，才得发现问题所在。

MOUSE_OVER - 鼠标移动到目标对象之上时触发，可以用于模拟按钮的 mouse over 效果；

MOUSE_MOVE - 鼠标在目标对象之上移动时触发，主要用于判断。比如判断在拖拽实例时，实例是否在允许的范围之内，如果超出，立刻停止拖拽或者重新设定实例的坐标；

MOUSE_DOWN - 鼠标在目标对象之上按下时触发。注意，只有按下鼠标左键时才会触发，右键和滚轮都不会触发。在目标对象之外按下鼠标左键，再移动到目标对象之上时，也不会触发；

MOUSE_UP - 鼠标在目标对象之上松开时触发。注意，只有松开鼠标左键时才会触发，右键和滚轮都不会触发。在目标对象之上按下鼠标左键，再移动到目标对象之外松开时，不会触发。但在目标对象之外按下鼠标左键，再移动到目标对象之上松开时，就会触发。

MOUSE_OUT- 鼠标移动到目标对象之外时触发。

MOUSE_WHEEL - 鼠标在目标对象之上转动滚轮时触发。

MOUSE_LEAVE - 当光标离开舞台时触发（stage.addEventListener(Event.MOUSE_LEAVE,leaveHandler);）。在使用自定鼠标后，在鼠标离开舞台时，触发 MOUSE_LEAVE 事件，然后可以把自定义的鼠标隐藏掉，避免还停留在舞台上。

2 mouseChildren。目标对象中含有子实例时，感应鼠标行为的是子时列，而非目标对象。如果使用 cursor.mouseEnabled=false; 就可以由目标对象来更应鼠标行为。

3 mouseEnabled。当实例重叠时，出于显示列表上方的实例总比下方实例更有优先权感应鼠标行为。当想让下方实例感应鼠标行为时使用 cursor.mouseEnabled=false; 即可。这常用于自定义鼠标后，去除自定义鼠标对鼠标行为的干涉，因为自定义鼠标往往一直处于鼠标下方，其他实例无法再感应到鼠标的变化。

另外，也许 DOUBLE_CLICK 也应该算做鼠标事件，但要使用它，必须先让 doubleClickEnabled=true:

```
var bg:Sprite=new Sprite();
bg.doubleClickEnabled=true;
bg.addEventListener(MouseEvent.DOUBLE_CLICK,clickHandler);
```

鼠标是 Flash 里追主要的互动因素，经常需要侦测鼠标事件(AS3 中鼠标事件小结)和得到鼠标的坐标。鼠标坐标的获取可以分为在文档类和在子类中，两种不同的情况。

1）如果是在时间线轴上，或者文档类上使用：

```
stage.mouseX 和 stage.mouseY
```

2）在子类（如_sprite:Sprite）上使用:

```
_sprite.mouseX 和 _sprite.mouseY
```

这里得到的是鼠标相对于_sprite 的坐标。如果需要的是相对于舞台的坐标，则应该使用 localToGlobal,如：

```
var mousePoint:Point=new Point(_sprite.mouseX, _sprite.mouseY);
mousePoint=_sprite.localToGlobal(mousePoint);
trace("Stage coordinates:"+mousePoint);
```

注：要使用以上代码别忘了 import flash.geom.Point;

mask属性遮罩层的缩放，拖拽，删除

作者：小 S

文章来源：http://www.xiaos8.com/default.asp?cat=1&page=13

mask 这个属性相信很多人看过帮助文档之后，不会仔细去看，特别是熟悉 AS2 的 setMask 的人当然包括我自己也放过这样的错误，因此在此提起，希望各位新手在学习的时候，把帮助文档看清楚

AS3 帮助文档中 DisplayObject 类的 mask 属性

mask 属性

mask:DisplayObject [read-write]

语言版本 : ActionScript 3.0
Player 版本 : Flash Player 9

调用显示对象被指定的 mask 对象遮罩。 要确保当舞台缩放时蒙版仍然有效，mask 显示对象必须处于显示列表的活动部分。 但不绘制 mask 对象本身。 将 mask 设置为 null 可删除蒙版。

要能够缩放遮罩对象，它必须在显示列表中。 要能够拖动蒙版 Sprite 对象（通过调用其 startDrag() 方法），它必须在显示列表中。 要为基于 sprite 正在调度的 mouseDown 事件调用 startDrag() 方法，请将 sprite 的 buttonMode 属性设置为 true。

根据帮助文档的说法，如果要缩放遮罩层，就必须把遮罩层放在相应的显示列表中，也就是 addChild(mask)

比如：我现在使用的是

```
stage.scaleMode = StageScaleMode.SHOW_ALL;
```

如果说你的遮罩层没有 addChild 那么，在缩放窗口的时候，mask 是不会改变宽高；反之加入了显示列表，就会跟着窗口一起缩放

mask 的拖拽和点击等事件也是如此！

综上所述：设置遮罩层最佳做法是放到显示列表，当然不排除有特别作用的

Math类的误区

作者：小 S

文章来源：http://www.xiaos8.com/article.asp?id=478

引言

无论刚入门的还是有一定 AS3 编程基础的，对 Math 类应该都不陌生了，但 Math 类的性能又知多少呢？请看下文吧。

1、Math.floor()

通俗的讲这是一个取整函数。

其实官方解释是

返回由参数 val 指定的数字或表达式的下限值。 下限值是小于等于指定数字或表达式的最接近的整数。

性能测试：

```
var num:Number = Math.PI;
var length:int = 10000000;
var time:int = getTimer();
for(var i:int = 0; i < length; i ++){
  Math.floor(num);
}
trace(getTimer() - time);
// 结果：1865
var num:Number = Math.PI;
var length:int = 10000000;
var time:int = getTimer();
for(var i:int = 0; i < length; i ++){
  int(num);
}
trace(getTimer() - time);
// 结果：69
```

结果很明显，int 比 floor 快，可能大家就要说了，那 Adobe 傻 X 写个这样接口干什么？其实这就是我想说的对于 floor 的使用误区。

仔细看官方解释，其实 floor 不是一个真正我们所理解的取整函数，他是去找最接近自己，且比自己小或者等于的整数，那这是什么意思呢？

```
var num:Number = - Math.PI;
trace(int(num));
trace(Math.floor(num));
// 结果1：-3
// 结果2：-4
```

这个例子很明确表示，当目标数字是负数时，int 和 floor 所得出的结果不一样。

int()的官方解释

将给定数字值转换成整数值。 从小数点处截断十进制值。

相信看到这，你已经很明白了。

但是实际上，如果你这样去使用 int()，跟 floor 输出的结果相同：

```
var num:Number = - Math.PI;
var length:int = 10000000;
var time:int = getTimer();
for(var i:int = 0; i < length; i ++){
```

```
  if(num < 0){
    int(num) - 1;
  }else{
    int(num);
  }
}
trace(getTimer() - time);
// 结果：132
```

2、Math.pow()

对于这个方法，我也不知道说什么好了，先看性能测试吧：

```
var a:int = 3;
var b:int = 4;
var c:int = 5;
var length:int = 10000000;
var time:int = getTimer();
for(var i:int = 0; i < length; i ++){
  c * c == a * a + b * b;
}
trace(getTimer() - time);
// 结果：95
var a:int = 3;
var b:int = 4;
var c:int = 5;
var length:int = 10000000;
var time:int = getTimer();
for(var i:int = 0; i < length; i ++){
  Math.pow(c,2) == Math.pow(a,2) + Math.pow(b,2);
}
trace(getTimer() - time);
// 结果：7999
```

勾三股四弦五大家应该比较多，但是用"*"乘法运算和 pow 的性能比较那是非常明显啊。
难道大家又想说 Adobe 傻逼了？这儿我为它平反吧。

```
var num:Number = Math.PI;
var length:int = 10000000;
var time:int = getTimer();
for(var i:int = 0; i < length; i ++){
  Math.pow(num,10000);
}
trace(getTimer() - time);
// 结果：6682
```

先不说"*"乘法运算比 pow 快，就上面这段你能把它换算成使用"*"乘法运算吗？
而且 pow 其实是可以这样用的：（数学学得好都知道开方其实是可以转换成乘方来算的）

```
trace(Math.pow(27,1/3));
trace(Math.pow(256,1/4));
trace(Math.pow(3125,1/5));
```

而开方函数 Adobe 只提供了 sqrt 一个开平方根的接口（经测试 Math.sqrt(9)比 Math.pow(9,1/2)快，但开立方等就得靠 pow 了）。

3、Math.round()

这个方法跟 floor 一样的，先看性能测试：

```
var num:int = Math.PI;
var length:int = 10000000;
var time:int = getTimer();
for(var i:int = 0; i < length; i ++){
  Math.round(num);
}
trace(getTimer() - time);
// 结果：1931
var num:int = Math.PI;
var length:int = 10000000;
var time:int = getTimer();
for(var i:int = 0; i < length; i ++){
  int(num + 0.5)
}
trace(getTimer() - time);
// 结果：68
```

四舍五入其实加个 0.5 在取整，这样也是可以的，只不过用这个算法，存在跟第一个同样的问题，当目标数值为负时，两种方式结果不一样，需要加个判断，目标数值为负就把结果-1。

暂时讲到这，下次讲讲 Math 的一些妙用，如果你有什么疑问或补充，欢迎跟帖~~

Tween类，flash缓冲效果讲解

作者：小 S

文章来源：http://www.xiaos8.com/default.asp?cat=1&page=13

最近做的东西，对于效果编程比较多，特别是缓冲，加速度等效果，特意研究了一下 Tween 类，过年没什么礼物给大家，写篇教程分享给大家，算是过年送给大家的新年礼物！

下面是官方写的 fl.transitions.Tween 类：
可以先尝试看一下这段代码的写法，用这个方法去看：
从构造函数看起，假设我使用了该类

new Tween(mc,"x",Regular.easeOut,0,200,4);

结合 adobe 的帮助文档，当初始化实例的时候，便对对象 mc 的 x 进行 Regular.easeOut 操作，具体是从 0 值到 200 值，历时 4 帧
然后问一个为什么，他为什么会对对象 mc 的 x 进行 Regular.easeOut 操作，具体是从 0 值到 200 值，历时 4 帧？
带着这个问题，现在开始看这个类吧！从构造函数看

```
package fl.transitions
{
import flash.events.*;
import flash.display.*;
import flash.utils.*;
[Event(name="motionChange", type="fl.transitions.TweenEvent")]
[Event(name="motionFinish", type="fl.transitions.TweenEvent")]
[Event(name="motionLoop", type="fl.transitions.TweenEvent")]
[Event(name="motionResume", type="fl.transitions.TweenEvent")]
[Event(name="motionStart", type="fl.transitions.TweenEvent")]
[Event(name="motionStop", type="fl.transitions.TweenEvent")]
public class Tween extends EventDispatcher
{
   protected static var _mc:MovieClip = new MovieClip();

  public var isPlaying:Boolean = false;

   public var obj:Object = null;

  public var prop:String = "";

  public var func:Function = function (t:Number, b:Number, c:Number, d:Number):Number { return c*t
/d + b; }

  public var begin:Number = NaN;

  public var change:Number = NaN;

  public var useSeconds:Boolean = false;

  public var prevTime:Number = NaN;
```

```
 public var prevPos:Number = NaN;

 public var looping:Boolean = false;

 private var _duration:Number = NaN;

 private var _time:Number = NaN;

 private var _fps:Number = NaN;

 private var _position:Number = NaN;

 private var _startTime:Number = NaN;

 private var _intervalID:uint = 0;

 private var _finish:Number = NaN;

 private var _timer:Timer = null;

public function get time():Number
{
return this._time;
}

 public function set time(t:Number):void
 {
   this.prevTime = this._time;
   if (t > this.duration) {
     if (this.looping) {
       this.rewind (t - this._duration);
       this.update();
       this.dispatchEvent(new TweenEvent(TweenEvent.MOTION_LOOP, this._time, this._position));
     } else {
       if (this.useSeconds) {
         this._time = this._duration;
         this.update();
       }
       this.stop();
       this.dispatchEvent(new TweenEvent(TweenEvent.MOTION_FINISH, this._time, this._position));
     }
   } else if (t < 0) {
     this.rewind();
     this.update();
   } else {
     this._time = t;
     this.update();
   }
```

```
 }

public function get duration():Number
{
return this._duration;
}

 public function set duration(d:Number):void
 {
   this._duration = (d <= 0) ? Infinity : d;
 }

public function get FPS():Number
{
return this._fps;
}

 public function set FPS(fps:Number):void
 {
   var oldIsPlaying:Boolean = this.isPlaying;
   this.stopEnterFrame();
   this._fps = fps;
   if (oldIsPlaying)
   {
     this.startEnterFrame();
   }
 }

public function get position():Number
{
return this.getPosition(this._time);
}

 public function set position(p:Number):void
 {
   this.setPosition (p);
 }

public function getPosition(t:Number=NaN):Number
{
if (isNaN(t)) t = this._time;
return this.func (t, this.begin, this.change, this._duration);
}

 public function setPosition(p:Number):void
 {
   this.prevPos = this._position;
   if (this.prop.length)
     this.obj[this.prop] = this._position = p;
```

```
    this.dispatchEvent(new TweenEvent(TweenEvent.MOTION_CHANGE, this._time, this._position));
  }

 public function get finish():Number
 {
 return this.begin + this.change;
 }

  public function set finish(value:Number):void
  {
    this.change = value - this.begin;
  }

  function Tween(obj:Object, prop:String, func:Function, begin:Number, finish:Number, duration:Nu
mber, useSeconds:Boolean=false)
  {
    if (!arguments.length) return;
    this.obj = obj;
    this.prop = prop;
    this.begin = begin;
    this.position = begin;
    this.duration = duration;
    this.useSeconds = useSeconds;
    if (func is Function) this.func = func;
    this.finish = finish;
    this._timer = new Timer(100);
    this.start();
  }

  public function continueTo(finish:Number, duration:Number):void {
    this.begin = this.position;
    this.finish = finish;
    if (!isNaN(duration))
      this.duration = duration;
    this.start();
  }

  public function yoyo():void
  {
    this.continueTo(this.begin, this.time);
  }

  protected function startEnterFrame():void
  {
    if (isNaN(this._fps))
    {
      _mc.addEventListener(Event.ENTER_FRAME, this.onEnterFrame, false, 0, true);
    }
    else
```

```
    {
      var milliseconds:Number = 1000 / this._fps;
      this._timer.delay = milliseconds;
this._timer.addEventListener(TimerEvent.TIMER, this.timerHandler, false, 0, true);
this._timer.start();
    }
    this.isPlaying = true;
  }

  protected function stopEnterFrame():void
  {
    if (isNaN(this._fps))
    {
      _mc.removeEventListener(Event.ENTER_FRAME, this.onEnterFrame);
    }
    else
    {
      this._timer.stop();
    }
    this.isPlaying = false;
  }

  public function start():void
  {
    this.rewind();
    this.startEnterFrame();
    this.dispatchEvent(new TweenEvent(TweenEvent.MOTION_START, this._time, this._position));
  }

  public function stop():void
  {
    this.stopEnterFrame();
    this.dispatchEvent(new TweenEvent(TweenEvent.MOTION_STOP, this._time, this._position));
  }

  public function resume():void
  {
    this.fixTime();
    this.startEnterFrame();
    this.dispatchEvent(new TweenEvent(TweenEvent.MOTION_RESUME, this._time, this._position));
  }

  public function rewind(t:Number=0):void
  {
    this._time = t;
    this.fixTime();
    this.update();
  }
```

```
  public function fforward():void
  {
    this.time = this._duration;
    this.fixTime();
  }

  public function nextFrame():void
  {
    if (this.useSeconds)
      this.time = (getTimer() - this._startTime) / 1000;
    else
      this.time = this._time + 1;
  }

  protected function onEnterFrame(event:Event):void
  {
    this.nextFrame();
  }

  protected function timerHandler(timerEvent:TimerEvent):void
  {
    this.nextFrame();
    timerEvent.updateAfterEvent();
  }

  public function prevFrame():void
  {
    if (!this.useSeconds) this.time = this._time - 1;
  }

  private function fixTime():void
  {
    if (this.useSeconds)
      this._startTime = getTimer() - this._time*1000;
  }

  private function update():void
  {
    this.setPosition(this.getPosition(this._time));
  }

}
}
```

不知大家看的如何，反正我现在来告诉你结果，你会发现，实际这个类很简单，他就做了一件事，就是每个时间变化每个不同的状态，而具体怎么变的，谁看到了？比如加速度，弹簧效果等都应该有算法，但是这里面并没有看到算法啊，只看到他每次更改时间 time 不管如何，总会 update()一下，算法就在这，它会调用 getPosition 方法，而该方法返回的值是 this.func (t, this.begin, this.chang

e，this._duration)（插播参数讲解，func 的 4 个参数分别是，当前运动到的 time 值，初始值即构造函数中第 4 个参数，要改变的值即结束值减初始值，总时间即构造函数中第 6 个参数）func 方法得到的就是变化一下 time 之后的物体值，而这个 func 就是构造函数中第三个参数 Regular.easeOut，这个参数是 fl.transitions.easing 包中的某个类的某个方法，如果使用方法，返回的值就是 return -c * (t /= d) * (t - 2) + b;这个就是缓冲算法，fl.transitions.easing 包中还有很多不同移动效果的算法，比如弹簧，加速度等。竟然看到这里，大家都知道了 Tween 只不过是调用这些算法的一个东西罢了，那么我们根据自我需求写一些简单的缓冲类，然后调用 adobe 的这些精髓算法呢？

先看看官方的算法源代码
adobe 官方 fl.transitions.easing.Regular 类，源代码：

```
package fl.transitions.easing
{

public class Regular
{

  public static function easeIn(t:Number, b:Number,
                c:Number, d:Number):Number
  {
    return c * (t /= d) * t + b;
  }

  public static function easeOut(t:Number, b:Number,
                c:Number, d:Number):Number
  {
    return -c * (t /= d) * (t - 2) + b;
  }

  public static function easeInOut(t:Number, b:Number,
                  c:Number, d:Number):Number
  {
    if ((t /= d / 2) < 1)
      return c / 2 * t * t + b;

    return -c / 2 * ((--t) * (t - 2) - 1) + b;
  }
}

}
```

接下来看看，我为了某个项目，赶时间写出来针对显示对象一个非常简单的缓冲类，然后调用了官方的 fl.transitions.easing 包中的算法

```
package index.item.pairBumping{

  import flash.display.DisplayObject;
  import flash.events.Event;
  import flash.events.EventDispatcher;
```

```
  public class Motion extends EventDispatcher{

    private var _target:DisplayObject;
    private var proNum:Number;
    private var startNum:Number;
    private var endNum:Number;
    private var actionStr:String;
    private var num1:uint;
    private var num2:uint;
    private var func:Function;

    //构造函数与 Tween 基本上一样，只不过没有按时间计算，只有按帧运动
    //每个参数所表示的值也是一样的，fun:Function 传入的参数和 Tween 一样，使用官方的 fl.transitions.easing 包
    public function Motion(target:DisplayObject,str:*,fun:Function,_start:Number = 0,_end:Number
= 1,_pro:Number = 4){
      actionStr = str;
      _target = target;
      proNum = _pro;
      startNum = _start;
      endNum = _end;
      func = fun;
    }

    //创建 Motion 实例化后，并没有立即播放得执行 play 才播放，当然播放完一次，也能使用 play 继续播放
    public function play(){
      stop();
      num1 = 0;
      _target[actionStr] = startNum;
      _target.addEventListener(Event.ENTER_FRAME,fun1);
    }

    //顺走所执行事件
    private function fun1(e:Event){
      var t = num1 ++;//当前运行时间
      var d = proNum;//总时间
      var b = startNum;//开始值
      var c = endNum - startNum;//要改变的值
      _target[actionStr] = func(t,b,c,d);//调用官方算法，并且传入 4 个参数，进行计算!
      if(t > d){//判断是否时间到了
        stop();//ok，播放完毕，那么我们停止吧
        _target[actionStr] = endNum;//并且把参数强制性变成最终值
      }
    }

    //反过来播放一次
    public function back(){
      stop();
      num2 = 0;
      _target[actionStr] = endNum;
```

```
      _target.addEventListener(Event.ENTER_FRAME,fun2);
    }

    //反过来播放的执行事件
    private function fun2(e:Event){
      var t = num2 ++;
      var d = proNum;
      var b = endNum;
      var c = startNum - endNum;
      _target[actionStr] = func(t,b,c,d);
      if(t > d){
        stop();
        _target[actionStr] = startNum;
      }
    }

    //停止播放
    public function stop(){
      _target.removeEventListener(Event.ENTER_FRAME,fun1);
      _target.removeEventListener(Event.ENTER_FRAME,fun2);
      dispatchEvent(new Event("stop"));
    }
  }
}
```

关于try..catch

作者：大头

文章来源：http://www.cnblogs.com/cwin5/archive/2009/02/27/1400011.html

try..catch 主要是用来抛出异常的。防止程序中断

一般 try..catch 用在一些容易出错的地方，例如二进制的转换，或者网络异步方面。这次地方出错率比较高一些

假如数据是恒定的，当然这样就可以，但是当数据是从外部载入的或者从外部定义的，就不能保证是正确的。一旦出现问题，程序就会中断

有些错误是不用 catch 的，像除数是 0 等等常见的小错误，但是运行时有可能产生的错误需要 catch 的，这方面编译器一般是检测不出来的，需要显式处理，不然的话程序的健壮性就值得怀疑了！

例如在 socket 连接里面的数据转换我用的比较多，因为有时候万一发生 socket 数据出错那么很容易程序抛出异常，这个时候用 try catch 比较适合，不会打断程序

一般在和外部交换数据的时候用的比较多，应为我么根本就不知道会从外据接受到什么样的数据，很可能是错误的，所以一个原则就是不能相信从外部得到的数据，使用前一定要审核，那么这里一般就会用到 trycatch

"异常处理"的运行成本是比较高的,能避免使用的地方就避免使用.

即触发Event.CANCEL事件—OutDisplay类

作者：小 S

文章来源：http://www.xiaos8.com/article.asp?id=36

首先是代码

```
package index.base.program{

  import flash.display.DisplayObject;
  import flash.display.DisplayObjectContainer;
  import flash.events.Event;
  import flash.events.EventDispatcher;

  public class OutDisplay extends EventDispatcher{

    //保存所有对象
    private var objAr:Array;
    //保存侦听类型
    private var type:String;

    //构造函数
    public function OutDisplay(_type:String,dis:DisplayObject,...objs){
      type = _type;
      objAr = objs;
      dis.addEventListener(_type,fun);
    }

    //事件处理
    private function fun(e:Event):void{
      for(var i:uint = 0;i<objAr.length;i++){
        if(e.target == objAr[i]) return;
        if(objAr[i] is DisplayObjectContainer){
          if(objAr[i].contains(e.target)) return;
        }
      }
      e.currentTarget.removeEventListener(type,fun);
      dispatchEvent(new Event(Event.CANCEL));
    }

  }
}
```

类讲解：

通过构造函数，创建你所需侦听的事件类型，显示对象作用区域，以及可活动区域

public function OutDisplay(_type:String,dis:DisplayObject,...objs)

在 dis 区域侦听 MouseEvent.CLICK 事件，则在触发这个事件的时候，判断目标对象是否是 objs 里面的对象或者子对象

如果是的，则表示用户对该事件的活动区域尚且在范围内，不发布 Event.CANCEL 事件

如果触发的事件目标不是 objs 中的对象或者子对象，则发布 Event.CANCEL，外部侦听到事件，即可对显示对象进行一些操作，比如消失等

用法举例：

```
import index.base.program.OutDisplay;

new OutDisplay(MouseEvent.CLICK,stage,mc).addEventListener(Event.CANCEL,fun);

function fun(e:Event){
  e.currentTarget.removeEventListener(Event.CANCEL,fun);
  removeChild(mc);
}
```

在舞台 stage 进行鼠标点击事件侦听，如果目标对象不是 mc 或者 mc 的子对象，则触发 fun 侦听器，即去除 mc 显示对象

也可以这样

```
import index.base.program.OutDisplay;

new OutDisplay(MouseEvent.MOUSE_OVER,stage,mc).addEventListener(Event.CANCEL,fun);

function fun(e:Event){
  e.currentTarget.removeEventListener(Event.CANCEL,fun);
  removeChild(mc);
}
```

当鼠标离开 mc，则触发去除 mc

备注：构造函数中的，作用区域显示对象可以含有多个！

声音的全局控制类SoundMixer

作者：小 S

文章来源：http://www.xiaos8.com/article.asp?id=34

在群里面经常碰到朋友问怎么控制所有的声音，正好我自己经历过这么一次，找到了 SoundMixer 类

SoundMixer 类
flash.media 包中的

SoundMixer 是一个静态类，其中含有控制全局属性的静态属性 soundTransform
通过修改 SoundMixer 中的静态属性 soundTransform 可以改变 flash 中所有声音属性，比如让他的音量调节到最小，即关闭声音，静音模式
另外这个类还有个 computeSpectrum 的静态方法，可以获取当前 flash 的声音播放的波形，在制作 mp3 播放器的时候，用起来不错！

另外这个类当中还有一个 stopAll 的静态方法，他的作用等同于 AS2 的 stopAllSound()

事件层的技巧

作者：大头

文章来源：http://www.cnblogs.com/cwin5/archive/2009/03/10/1408199.html

类 A:

```
package
{
    import flash.events.Event;

    /**
     * ...
     * @author cwin5
     */
    public class A
    {
        public function A() {
            EventLayer.event.dispatchEvent(new Event("test"));
        }
    }

}
```

在类 A 产生测试事件.

类 B:

```
package
{
    import flash.events.Event;

    /**
     * ...
     * @author cwin5
     */
    public class B
    {
```

```
        public function B() {
            EventLayer.event.addEventListener("test", Test);
        }

        private function Test(e:Event):void
        {
            trace("test");
        }
    }

}
```

类 B 侦听测试事件

事件层 EventLayer:

```
package
{
    import flash.events.EventDispatcher;

    /**
     * ...
     * @author cwin5
     */
    public class EventLayer
    {
        public static const event:EventDispatcher = new EventDispatcher();
    }

}
```

事件层声明一个静态常量

这样写事件的好处在于不管显示结构如何变化..都不用修改事件的结构代码

AS3 与 JS 进行交互

作者：A 闪

文章来源：http://hi.baidu.com/%B0%B5%BA%DA%B2%E0%CE%C0/blog/item/324c76a1e97d8e82471064a7.html

这个话题在网上应该说很好找，有很多这方面的资料。比者也看过一些，不过大部分都是 AS2 与 JS 进行交互。很少提到 AS3，即使有，也让人感觉浅尝辄止。很多东西并没有介绍的太清楚。所有笔者萌生了写这样一篇教程的想法！

提及 AS3 与外部脚本的交互，笔者认为可以总结成两种。一是 AS3 调用外部函数，二是外部脚本调用 AS3 函数。无外乎就这两种。在调用函数的同时，我们还可以向函数传递一些参数。这就达到了传递数据的目的。举一个最简单的例子。我们平时在网络上看视频的时候，这些用 AS3 制作的播放器就是由外部脚本（可能是 JS,也可能是 ASP 或其他脚本）传递给它的视频地址。从而就简化我们的后台程序。有的人会问！你用播放器去读取外部的 XML 数据不是也可以吗？确实，当时，当你有成千上万个视频的时候，你不可能输入这么多数据。一是比较麻烦，二来是影响了网页运行的速度。所以，AS3 与外部脚本进行交互是非常重要的。

其实,我们做的绝大多数带有交互性的 FLASH 程序都要多多少少与后台的其他语言进行交互。这里，我们就以 JS 脚本来举例说明交互的过程。当然，AS3 与 JS 脚本可以说有血缘关系，具体的大家可以去查看一下 FLASH 的发展历史。好了！废话不多说，我们这就开始讲如何让 JS 与 AS3 进行一些信息的交换。本节中，我们只讲解 JS 调用 AS3 中的函数。反向的调用我们将在下一节中进行讲解！

首先，我们新建一个 FLASH 文档，然后在舞台上绘制一个动态文本，该文本的实例名称为 wen_txt。好了！美工的部分就算完成了（我们这里是一个非常简单的实例，不要求太复杂。力求突出重点）！

现在，我们要在 FLASH 中建立一个允许被外部调用的函数。那么如何去建立声明这样一个函数呢？我们要使用到一个类,这个类的名字就是 ExternalInterface。很多人都没有太注意过这个类。这个类实际上是在 flash.external 包中。这个包是专门用来和外部容器进行通信的。那么对这个 ExternalInterface 类，它的官方解释是“ExternalInterface 类是外部 API，在 ActionScript 和 Flash Player 的容器之间实现直接通讯的应用程序编程接口，例如，含有 JavaScript 的 HTML 页。”很显然，这里已经提到了 JS。好了！下面我们就来写一下脚本吧！如下：

```
ExternalInterface.addCallback("abcd",yun);
function yun(zi:String):void{
wen_txt.text = zi;
}
```

我们看，建立一个可以被外部调用的函数实际上就是使用了 ExternalInterface 类的 addCallback 方法。我们可以从宏观上这样理解。但实际上它的真正作用是将一个函数注册为可从容器调用。实际上我们的 FLASH 端就这么简单。好了！保存文件，发布。注意！这里，我发布出来的 SWF 文件的文件名是 ab.swf。

上面我们已经将 FLASH 端的脚本写好了，下面我们来编写 JS 代码，其实也是一个 HTML 网页代码，这个 HTML 代码中包含 JS 代码！如下：

```
<!-- saved from url=(0014)about:internet -->
<html lang="en">
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>FLASH 与 JS 交互</title>
<script language="JavaScript">
     function pageInit() {
        sendToActionScript("你好");
     }
     function sendToActionScript(value) {
         window.ExternalInterfaceExample.abcd(value);
     }
</script>
</head>
<body onload="pageInit();">
     <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
             id="ExternalInterfaceExample" width="500" height="500"
             >
         <param name="movie" value="ab.swf" />
     </object>
</body>
</html>
```

好了！现在将这个网页保存到和 ab.swf 同一文件夹下，然后运行这个网页。当程序运行的时候，我们就可以看到 FLASH 中的动态文本会出现“你好”字样！这个过程就是 JS 调用 AS3 函数，同时向 FLASH 传递了一个参数，或者说传递了一组数据。

关于这段 JS 代码，我们就不多说了。因为涉及到很多 JS 脚本的只是。大家可以在网上查找相关的资料，就可以明白这段 JS 脚本的含义了！

本节我们就讲解到这里，下一节我们讲解，FLASH 如何向 JS 传递参数！

书接上文，上次我们讲到 JS 向 FLASH 传递参数。本节我们来讲解 FLASH 向 JS 传递参数。

先说说原理吧。实际上我们所浏览的每一个网页都可以看作是一个容器。那何为容器呢？说白了就是一个盒子。在这个盒子里面放着许多东西，包括网页中的文字，图片，FLASH，脚本，按钮，文本框等。所以，我们的 FLASH 如果想访问网页中的 JS 函数，实际上就是访问上一级中的函数。这种操作相对就要简单一些。因为我们的 JS 函数对网页中的成员都是公开的，这个网页中的任何元素都能够调用这个 JS 函数。同样，FLASH 也拥有这样的权限。那么好了！我们来看看在网页中具体是怎样做的。

我们这次先来写网页代码，因为网页这边是接受端。代码如下：

```
<!-- saved from url=(0014)about:internet -->
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>FLASH 与 JS 交互</title>
<script language="JavaScript">
     function sendToJavaScript(value) {
         document.forms["form1"].output.value += "\n" + "ActionScript says: " + value;
     }
```

```
</script>
</head>
<body>
    <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
            id="ExternalInterfaceExample" width="500" height="500"
            >
        <param name="movie" value="ab.swf" />
    </object>
    <form name="form1" onsubmit="return false;">
        <textarea cols="60" rows="20" name="output"
readonly="true">Initializing...</textarea>
    </form>
</body>
</html>
```

具体的什么意思笔者在这里就不多说了，属于 JS 方面的知识，大家可以去查看相关资料。好了下面就是 FLASH 端的代码了！我们先来做美工界面。画一个输入文本，实例名为 wen_txt，再做一个影片剪辑元件，实例名为 an_mc。好了！美工部分做完了！我们来写脚本。如下：

```
an_mc.addEventListener(MouseEvent.CLICK,chuan);
function chuan(evt:Event):void {
ExternalInterface.call("sendToJavaScript", wen_txt.text);
}
```

我们看，其实在 FLASH 中调用外部的 JS 函数还是用到了 ExternalInterface 类。而这次我们是使用的 call 方法。

好了！运行网页，我们在 FLASH 的输入文本中输入一串文字，然后按一下按钮。之后这段文字就会出现在网页的文本框里面！

大家可以自己动手做一次，体会一下 FLASH 向 JS 传递参数的过程！

本地通讯 LocalConnection 的基础应用

作者：DN_Web

文章来源：http://hi.baidu.com/dn%5Fweb/blog/item/9c8a8cdca6f940e677c638d6.html

本地通讯,swf<->swf 之间互相通讯

我们建立一个客户 swf

```
package
{
import flash.display.Sprite;
import flash.net.LocalConnection;
import flash.text.TextField;
import flash.net.registerClassAlias;
/**
* ...
* @author DN
*/
public class Client extends Sprite
{
   private var _localConnection:LocalConnection;
   public function Client()
   {
    registerClassAlias("com.Person", Person);
    _localConnection = new LocalConnection();
    try {
     _localConnection.connect("_channel_one");
     _localConnection.client = this;
    }catch (e:Error) {
     throw new Error("不能建立连接,通道已经被占用");
    }
   }
   public function mess(person:Person) {
    var personMess:String = person.toString();
    TextField(messText).appendText(personMess+"\n");
   }

}

}
```

一个服务 swf

```
package
{
import flash.display.Sprite;
import flash.net.LocalConnection;
```

```
import flash.events.MouseEvent;
import flash.events.StatusEvent;
import flash.net.registerClassAlias;
/**
* ...
* @author DN
*/
public class Server extends Sprite
{
   private var _connection:LocalConnection;
   private var _person:Person;
   public function Server()
   {
    registerClassAlias("com.Person", Person);
    _person = new Person();
    _person.name = "DNight";
    _person.age = 24;
    _connection = new LocalConnection();
    _connection.addEventListener(StatusEvent.STATUS, onStatus);
    sendBtn.addEventListener(MouseEvent.CLICK,onSendDataHandle);
   }
   private function onSendDataHandle(event:MouseEvent) {
    _connection.send("_channel_one","mess",_person);
   }
   private function onStatus(event:StatusEvent) {
    switch (event.level) {
                case "status":
                    trace("LocalConnection.send() succeeded");
                    break;
                case "error":
                    trace("LocalConnection.send() failed");
                    break;
            }
   }

}

}
```

要传递的对象

```
package
{

/**
* ...
* @author DN
*/
public class Person
```

```
{
   private var _personName:String;
   private var _personAge:int;
   public var name:String;
   public var age:int;
   public function Person()
   {

   }
   public function toString():String {
    return "姓名:" + name + "\n" + "年龄:" + age;
   }

}

}
```

笔记:LocalConnection采用AMF来进行数据传递,因此如果我们想在接收后反序列话能继续使用Person类的方法,或者属性， 请使用registerClassAlias()进行类注册,注意构造函数如果带参数不能反序列话

LocalConnection 的 channel 是单通讯的,不能双向发送数据,因此如果 server 想要知道接收端是否获取了数据,需要客户端利用 send 在新建一个新通道,通过方法,发送参数告诉 server 端已经获取到了数据

关于 flash 域 与 策略文件问题

作者：粉色男孩

文章来源：http://hi.baidu.com/fsnhf/blog/item/80629dfb3c676e116d22ebd0.html

很高兴今天 jaja 给我讲解了关于 flash 域与策略文件的问题。下面我也就他的讲解说明这两个东东到底是什么意思

Flash Player 安全模型的问题:

比如当我们使用 Loader 类去加载外部的图片或者 swf 文件时候，就需要考虑到安全模型了,在通常情况下访问同样域内是没有问题的，但是很多时候我们的 swf 文件需要去访问服务器上的文件，或者说有些时候服务器的 swf 需要访问我们本地机器上的某些文件.那么这样一来就涉及到了跨域的问题

首先我来说下什么叫域:

"域"->顾名思义就是不同地址，打个比方就是说两个房子，服务器叫"大房子"本地叫"小房子",在打个比方，www.baidu.com.与www.sina.com就是两个域,当我们两个房子或者两个服务器之间想要进行通信那么怎么办呢？这时候需要有个"商定"，这个"商定"也就是我们所说的策略文件

"策略文件"->是一个 XML 文件,名字为 crossdomain.xml,它的格式为:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy
SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" />
</cross-domain-policy>
```

这个文件允许任何域的 flash 客户端访问服务器，可以在<allow-access-from domain="*" />这一句话中设置指定的域的 flash 客户端访问。

前面所说的都是把策略文件放在服务器的跟目录下,在某些时候如果策略文件没有放在根目录下，那该怎么办呢？

正好,jaja 又解说了一下:如果不放在网站根目录的话,要用 flash 客户端首先加载安全策略文件并用 Security.loadPolicyFile(filepath);方法加载安全策略文件
好了，今天我了解了域与策略文件问题,您呢？

解决 AS3 Socket 编程中最令人头疼的问题

作者：粉色男孩

文章来源：http://hi.baidu.com/fsnhf/blog/item/410c5a456e914f3787947356.html

——解决 AS3 Socket 安全问题

什么是最令人头疼的问题?也许大家会异口同声的说：“安全问题”，不错，不仅仅是 AS3 的 Socket，整个 AS3 语言中最令人头疼的问题也无非就是安全问题了。

很 多同行的兄弟在郁闷的时候就会骂 Adobe。但是，骂归骂，问题终归是要解决的，Adobe 做这样的限制肯定是有他的用意的，大家都知道，swf 文件是很 容易被反编的，那么也就是说你的 swf 文件内部与服务器通信的方式及路径是很容易被别人发现的，如果你的服务器中没有任何访问限制，那么你的服务器很容易 被一些人攻破，这并不需要很高的水平，只要一直刷，你的服务器就完了。

兄弟，您骂够了吗？我们来解决问题吧？

用 Java 写完 Socket 服务器后，运行，一切正常，用 Flex 写全 Socket 客户端后，运行，一切正常，可是当把生成的 swf 文件拷到其它地方来运行就出错了，总是无法连接服务器，然后就开始抛 securityError，下面我们看一下输出信息。

打 开 Flash CS3，打开远程调试器，选择菜单如 Debug->Begin Remote Debug Session->ActionScript3.0（中文版选择菜单如：调试－＞开始 远程调试会话－＞ActionScript3.0），下面我们打开客户端 swf 文件（记得此文件一定是调试版 swf），则它会自动连接 Flash CS3 的调试器，在 Flash CS3 中输出相关的调试信息。

可以看到输出信息如下：

－－

[SWF] C:\Users\Administrator\Desktop\MyClientFlash.swf - 1112717 bytes after decompression

警告: [strict] 将忽略 xmlsocket://localhost:9999 处的策略文件，因为出现语法错误。请访问 http://www.adobe.com/go/strict_policy_files_cn 以解决此问题。

*** 安全沙箱冲突 ***

到 localhost:9999 的连接已停止 - 不允许从 file:///C|/Users/Administrator/Desktop/MyClientFlash.swf 进行连接

错误: 拒绝请求位于 xmlsocket://localhost:9999 的资源（请求者从 file:///C|/Users/Administrator/Desktop/MyClientFlash.swf 发出请求），原因是缺乏策略文件权限。

－－

原因是没有在服务器的 9999 端口放置安全策略文件（我写的服务器用的是 9999 端口），那么好吧，我就在此处给客户端返回了一个安全策略文件信息，此文件格式如下：

```
<?xml version="1.0"?>
<cross-domain-policy>
    <site-control permitted-cross-domain-policies="all"/>
    <allow-access-from domain="localhost" to-ports="9999,300-400" />
</cross-domain-policy>
```

上述示例中是允许来自 localhost 域的 swf 文件访问 9999 端口和 300 至 400 端口，你也可以用*来代替 localhost 以允许来自任何域的 swf 文件访问。

此时我将客户端文件请求的信息在 Java 中打印出来，看到的是一段包含<policy-file-request/>的字符串，当 Java 服务器接收到这个字符串时，立即返回安全策略文件字符串。

我想这样应该没什么问题了吧，可是当我再连接服务器时仍然无法连接，输出信息成了这样：

－－

[SWF] C:\Users\Administrator\Desktop\MyClientFlash.swf - 1112717 bytes after decompression

警告: 等待 socket 策略文件时在 xmlsocket://localhost:9999 上超时（3 秒钟）。这不会造成任何问题，但可访问 http://www.adobe.com/go/strict_policy_files_cn 以获得说明。

*** 安全沙箱冲突 ***

到 localhost:9999 的连接已停止 - 不允许从 file:///C|/Users/Administrator/Desktop/MyClientFlash.swf 进行连接

错误: 拒绝请求位于 xmlsocket://localhost:9999 的资源（请求者从 file:///C|/Users/Administrator/Desktop/MyClientFlash.swf 发出请求），原因是缺乏策略文件权限。

－－

这时我并不灰心，就按照它的说明去了 http://www.adobe.com/go/strict_policy_files_cn 这里查看说明了。

等我看到这个页面时，我真的郁闷了，上面全是乱说一通，根本不着边，全是一些没用的信息，亏 Adobe 想得出来提示我到这里看。

就 在我将要放弃解决这个问题的时候，我发现了另外一个现象，那就是当我刚开始连接服务器时，服务器端并没有打印出来客户端的请求信息 (<policy-file-request/>),而是在 Flash CS3 的调试器输出了超时错误之后，服务器端才打印出来这个请求信 息。

这时我看到了一线希望，那就是服务器端确实出现了等待，这肯定是服务器端的程序问题。服务器端接收请求的处理代码片断如下：

```
BufferedReader br=new BufferedReader(new InputStreamReader(s.getInputStream()));
if(br.readLine().indexOf("<policy-file-request/>")!=-1){
//开始返回授权文件信息
...
}
```

其中变量 s 是 ServerSocket 实例通过 accept 方法获得的 Socket 实例。

此时我开始怀疑是 readLine 方法的问题了，因为 readLine 方法是当程序读到\n 或者\r 时才会返回信息，所以肯定是此方法中出现了等待，因为起初客户端并没有传来换行符或者回车符。

于是我改变了读取字符串的方法，不再用 readLine 了，而取流中前 22 个字符，代码片断如下所示：

```
BufferedReader br=new BufferedReader(new InputStreamReader(s.getInputStream()));
char[] ch=new char[22];
br.read(ch, 0, ch.length);
StringBuffer sb=new StringBuffer();
for(int i=0;i< ch.length;i++){
sb.append(ch[i]);
}
String st=sb.toString();
if(st.indexOf("<policy-file-request/>")!=-1){
//开始返回授权文件信息
...
}
```

当我再连接服务器时，果然不出我所料，成功连接服务器。

但是连接过程有点慢，貌似此过程也正好是 3 秒钟，莫非在连接此服务器之前已经进行了另外一次连接，而且是失败的。

查看 Adobe 官网的资料才明白，flashplayer 会在连接指定的端口之前连接目标主机的 843 端口，如果 3 秒后得不到授权文件才再向指定的端口去请求授权文件，如果再经过 3 秒还得不到授权文件的话，则断开连接，抛出 securityError。

那 意思就是说在连接我的服务器的 9999 端口之前还连接了我的服务器的 843 端口，并且在 843 端口等待了 3 秒，没有得到授权文件，之后开始向我指定的端口 请求此授权文件。那好吧，既然你要了，我就给你吧，不给你的话你还再磨矶磨矶，于是我又在 843 端口开了一个 ServerSocket，此处专门处理授权 文件的请求。

这时我再连接服务器，呵呵，特快专列(T843)，立即就连接上了。

说明：貌似有很多客户机上的 843 端被禁用了，所以为了保险，需要在指定端口和 843 端都要能够处理授权文件的请求。

示例程序及源文件下载

解决 flash 跨域读取数据问题

作者：粉色男孩

文章来源：http://hi.baidu.com/fsnhf/blog/item/dfd63ad3920df6d8a9ec9a5e.html

——让你的 flash 可以从任何域读取数据

在做 flash 开发时经常会遇到安全问题，adobe 所做的各种安全限制真的让人很头疼，那么我们能不能有一种方法让 flash 蔽开这些安全问题去读取其它域的数据呢？

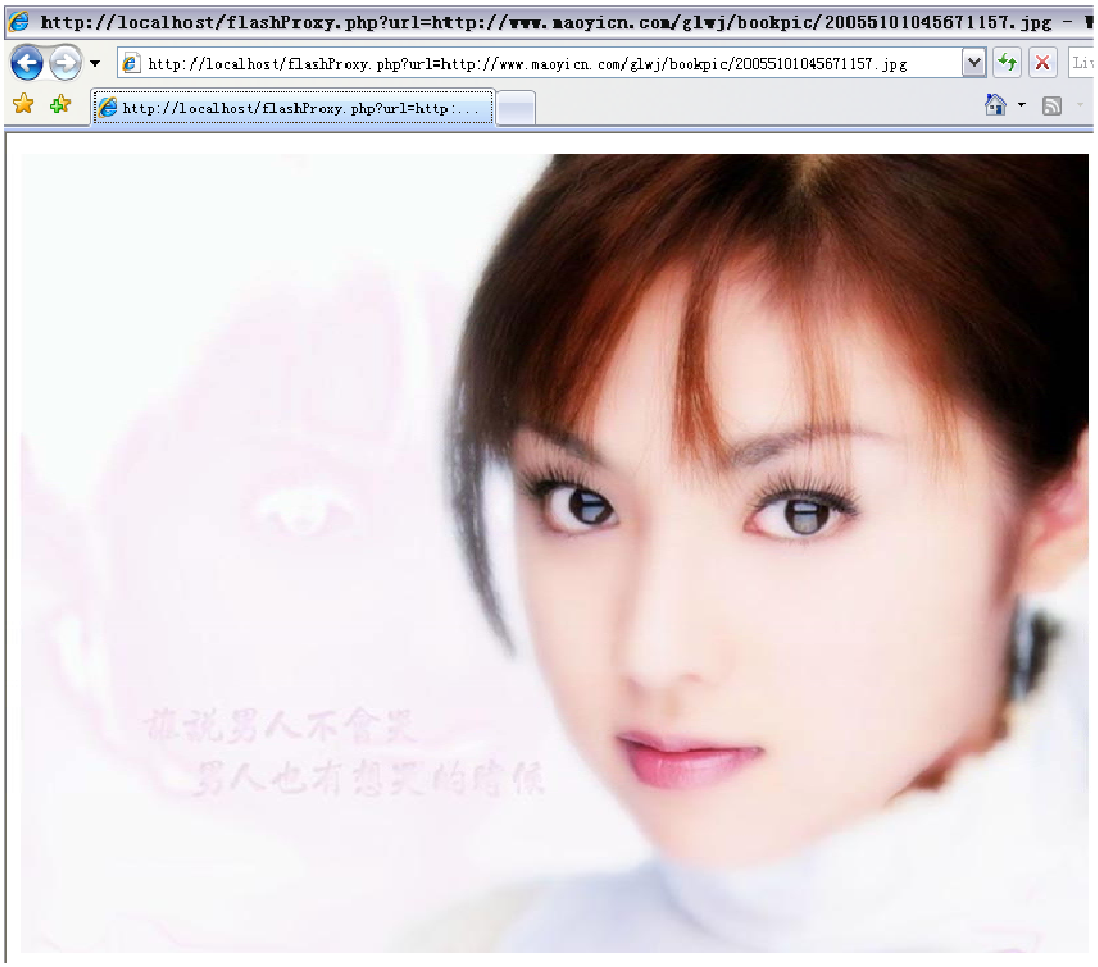
有。

当然，任何问题都是有解决方案的。

既然 flash 本身有很多安全限制，那么我们能不能借助其它的语言来实现呢？跟着这种想法，我想到了用 php 写个代理，用 php 来读取数据返回给 flash。这个 php 文件名为 flashProxy.php（当然名字随意），下面就是 php 代码：

```
<?php
    $file=file($_REQUEST["url"]);
    for($i=0;$i<count($file);$i++){
        echo $file[$i];
    }
?>
```

如何使用，就不用说了吧，下面这是我在我本机的 php 服务器测试效果，请注意地址栏中的参数传递。



图片流式加载

作者：DN_Web

文章来源：http://hi.baidu.com/dn%5Fweb/blog/item/fc628e10f6dfdacfa7ef3f9e.html

以往我们用 loader 去加载图片或者 swf 文件我们是需要在 complete 后才能使用,而且不能停止下载过程,现在我们可以用 URLStream 来达到流下载的目的.

我们用IE:打上http://www.as-max.cn/DN_Tree/Main.swf看一下

具体的代码如下.

```
package com.DNight.ByteLoader
{
import flash.events.EventDispatcher;
import flash.net.URLRequest;
import flash.net.URLStream;
import flash.utils.ByteArray;
import flash.events.Event
import flash.events.ProgressEvent

/**
* ...
* @author DN
*/
public class ByteLoader extends EventDispatcher
{
   private var _url:String;

   private var _data:ByteArray

   private var _stream:URLStream;

   public function get data() {

    return _data;
   }

   public function ByteLoader()
   {

   }

   public function load(url:String) {

    if (url!="") {

     _url = url
```

```
     }
     var request:URLRequest = new URLRequest(_url);

     _data = new ByteArray();

     _stream = new URLStream();

     _stream.addEventListener(Event.COMPLETE,onLoadCompleteHandle);
     _stream.addEventListener(ProgressEvent.PROGRESS,onProgressHandle);

     try {
      _stream.load(request);
     }catch (event:Error) {
      trace("Unable to load requested URL.");
     }

    }

    private function onLoadCompleteHandle(event:Event) {

     _stream.removeEventListener(Event.COMPLETE, onLoadCompleteHandle);

     _stream.removeEventListener(ProgressEvent.PROGRESS, onProgressHandle);

     _stream.close();

     dispatchEvent(event);

    }

    private function onProgressHandle(event:ProgressEvent) {
     trace(event.bytesLoaded);
     if (_stream.bytesAvailable == 0) {

      return
     }
     upData();
     dispatchEvent(event);
    }

    private function upData() {

     _stream.readBytes(_data,_data.length);
    }

}

}
```

主类

```
package
{
import flash.display.Loader;
import flash.display.Sprite;
import com.DNight.ByteLoader.ByteLoader
import flash.events.Event
import flash.events.ProgressEvent
/*** @author DN
*/
public class Main extends Sprite
{
   private var _path:String = "img/one.jpg";
   private var _byteLoader:ByteLoader

   public function Main()
   {
    initApp();
   }
   private function initApp() {

    _byteLoader = new ByteLoader();

    _byteLoader.load(_path);

    _byteLoader.addEventListener(Event.COMPLETE, onLoadedCompleteHandle);

    _byteLoader.addEventListener(ProgressEvent.PROGRESS,onProgressHandle);
   }

   private function onLoadedCompleteHandle(event:Event) {

    _byteLoader.removeEventListener(Event.COMPLETE, onLoadedCompleteHandle);
    _byteLoader.removeEventListener(ProgressEvent.PROGRESS, onProgressHandle);

   }

   private function onProgressHandle(event:ProgressEvent) {
    var loader:Loader = new Loader();
    loader.loadBytes(_byteLoader.data);
    this.addChild(loader);
   }
  }
}
```

值得注意的是我们采用 Loader 在 Progress 处理函数中依靠 loadBytes 来实现流加载过程.

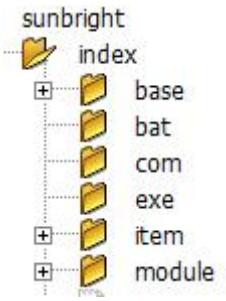
声明:ByteLoader 参考了网上兄弟的写法,并非原创

ActionScript 3.0 自写类整理笔记（一）——类的分包处理

作者：小 S

文章来源：http://www.xiaos8.com/default.asp?cat=1&page=12

分包情况：



图片地址：**uploads/200803/31_174221_2.jpg**

base 包：基础包，用于存放初级应用类
bat 包：应用包，用于存放高级应用类
com 包：系统化包，用于存放系统化的高级应用模块类
exe 包：框架包，用于存放框架方面的类
item 包：项目包，用于项目上靠经验积累下的类
module 包：组件包，用于存放组件的类

这段时间，我个人的情绪非常低落，为了摆脱这段时间的消极状态
因此开始整理在做项目中，碰到的种种问题，来整理出一套可用性高的类库
希望自己会坚持下去！

ActionScript 3.0 自写类整理笔记（二）——Dot类

作者：小 S

文章来源：http://www.xiaos8.com/default.asp?cat=1&page=12

今天刚刚写好，没做太多调试，望高手多多指正

flash 效果：

点击播放/隐藏媒体

/uploads/pro/Dot.swf

拖拽物体 1 和物体 2，就可以看到效果了

index.base.geom.Dot类讲解

基本功能：记录xy两点

构造函数

```
public function Dot(x_:Number = 0,y_:Number = 0,_isListen:Boolean = false)
```

前两个参数表示Dot的初始位置，第三个参数表示是否是一个功能性强的点

如果为false，当xy发生改变的时候，就不会发布事件，为true则会发布事件

bind 绑定显示对象方法

```
public function bind(_dis:DisplayObject,isInTime:Boolean = false):void
```

当Dot绑定到DisplayObject上之后，Dot的xy属性会随着_dis改变而改变

第一个参数为绑定的对象，第二个参数表示是否即时绑定

如果为false，Dot的xy属性不会随着_dis的改变而即时改变，但在获取Dot的xy属性，或者调用Dot的方法时，将会立即改变并且以_dis的xy属性为基准

如果为true，Dot的xy属性会随着_dis的改变而立即改变，如果isListen为true，还会即时发布xy改变的事件，那么等于可以侦听显示对象的xy属性，并且在改变后做出即时的反应

updata 刷新方法

```
public function updata():void
```

更新显示对象与Dot的xy属性

from 返回两点之间距离

```
public function from(_dot:Dot,isQuadrant:Boolean = false):Number
```

第一个参数表示，结束点，第二个参数表示，是否为真实距离

如果为false，返回的两点距离绝对是正数，指的是两点之间的绝对距离

如果为true，则返回相对坐标，那么是有可能的为负的！

angle 返回两点所形成的夹角

```
public function angle(_dot:Dot,isRadian:Boolean = false):Number
```

第一个参数表示，另外一个点，第二个参数表示，是否为弧度值

返回的角度是相对顺时间的真实角度值，具体的数值变化可以从上面的例子看出

quadrant 返回相对点所在的象限

public function quadrant(_dot:Dot,isMaster:Boolean = true):int
第一个参数表示另外一个点，第二个参数表示是否以该点为标准，具体请看示例
返回 0，表示两点在同一条横着或者竖着的直线上
返回 1，表示在第一象限，返回 2 表示第二象限…… 最高是第四象限。。
不知道象限是什么意思的，请看这 **点击跳转"象限"的百度百科页面**

clear 方法

public function clear():void
清空显示对象

length 属性（只读）

public function get length():Number
获取该点距 0,0 点的距离

x 属性

public function set x(num:Number):void
public function get x():Number
设置x属性，如果isListen为true，则会发布x改变的事件

y 属性

public function set y(num:Number):void
public function get y():Number
设置y属性，如果isListen为true，则会发布y改变的事件

isListener 属性

public var isListen:Boolean
指定设置isListen的值，是否为可侦听xy

举例：
上面的flash展示源代码

```
import index.base.geom.Dot;
import index.base.events.DotEvent;

var po1:Dot = new Dot(0,0,true);
var po2:Dot = new Dot(0,0,true);
po1.bind(p1,true);
po2.bind(p2,true);

po1.addEventListener(DotEvent.DOT_CHANGE,dotChangeFun);
po2.addEventListener(DotEvent.DOT_CHANGE,dotChangeFun);

function dotChangeFun(e:DotEvent):void{
  te.text = "物体 1 坐标：" + po1.x + "," + po1.y;
  te.appendText("\n 物体 2 坐标：" + po2.x + "," + po2.y);
  te.appendText("\n 两点之间距离：" + po1.from(po2));
  te.appendText("\n 所形成的角度：" + po1.angle(po2));
  te.appendText("\n 物体 1 所在象限：" + po1.quadrant(new Dot,false));
```

```
  te.appendText("\n物体2所在象限：" + po2.quadrant(new Dot,false));
  te.appendText("\n物体1对于物体2在象限：" + po2.quadrant(po1));
}

p1.addEventListener(MouseEvent.MOUSE_DOWN,p1MouseDownFun);
p2.addEventListener(MouseEvent.MOUSE_DOWN,p2MouseDownFun);

function p1MouseDownFun(e:MouseEvent):void{
  p1.startDrag();
  stage.addEventListener(MouseEvent.MOUSE_UP,p1MouseUpFun);
}

function p1MouseUpFun(e:MouseEvent):void{
  p1.stopDrag();
  stage.removeEventListener(MouseEvent.MOUSE_UP,p1MouseUpFun);
}

function p2MouseDownFun(e:MouseEvent):void{
  p2.startDrag();
  stage.addEventListener(MouseEvent.MOUSE_UP,p2MouseUpFun);
}

function p2MouseUpFun(e:MouseEvent):void{
  p2.stopDrag();
  stage.removeEventListener(MouseEvent.MOUSE_UP,p2MouseUpFun);
}
```

下面是类的源代码

```
package index.base.geom{

  import flash.events.EventDispatcher;
  import flash.display.DisplayObject;

  import index.base.events.DotEvent;

  public class Dot extends EventDispatcher{

    private var _x:Number;
    private var _y:Number;
    private var dis:DisplayObject;

    public var isListen:Boolean;

    public function Dot(x_:Number = 0,y_:Number = 0,_isListen:Boolean = false){
      _x = x_;
      _y = y_;
      isListen = _isListen;
    }
```

```
//绑定 DisplayObject
public function bind(_dis:DisplayObject,isInTime:Boolean = false):void{
  dis = _dis;
  updata();
  if(isInTime) dis.addEventListener("enterFrame",enterFrameFun);
}

//帧频繁事件
private function enterFrameFun(e:Object):void{
  if(_x != dis.x) x = dis.x;
  if(_y != dis.y) y = dis.y;
}

//更新 xy 数据
public function updata():void{
  if(dis != null){
    _x = dis.x;
    _y = dis.y;
  }
}

//计算该点与另外一点的距离
public function from(_dot:Dot,isQuadrant:Boolean = false):Number{
  updata();
  var num:Number = Math.sqrt(Math.pow(_dot.x - _x,2) + Math.pow(_dot.y - _y,2));
  if(!isQuadrant) num = Math.abs(num);
  return num;
}

//计算该点与另外一点所形成的线段与水平线的夹角,按顺时间计算
public function angle(_dot:Dot,isRadian:Boolean = false):Number{
  updata();
  var numx:Number = _dot.x - _x;
  var numy:Number = _dot.y - _y;
  var num:Number = Math.atan(numy/numx);
  if(!isRadian) num = num * 180 / Math.PI;
  return num;
}

//返回当前点处在另外一点的哪个象限中 或 返回另外一点处在当前点的哪个象限中
public function quadrant(_dot:Dot,isMaster:Boolean = true):int{
  updata();
  if(_x == _dot.x || _y == _dot.y){
    return 0;
  }

  var num:int;
  var p1:Boolean = (_x - _dot.x) > 0;
  var p2:Boolean = (_y - _dot.y) > 0;
```

```
      num = isMaster ? (p1 ? (p2 ? 2 : 3) : (p2 ? 1 : 4)) : (p1 ? (p2 ? 4 : 1) : (p2 ? 3 : 2));

      return num;
    }

    //返回该点距 0 点的距离
    public function get length():Number{
      updata();
      var num:Number = Math.sqrt(Math.pow(_x,2) + Math.pow(_y,2));
      return num;
    }

    //清除显示对象
    public function clear():void{
      dis = null;
    }

    //改变 X 坐标
    public function set x(num:Number):void{
      _x = num;
      if(dis != null) dis.x = num;
      if(isListen) dispatchEvent(new DotEvent(DotEvent.DOT_CHANGE,true));
    }

    //设置 X 坐标
    public function get x():Number{
      updata();
      return _x;
    }

    //改变 Y 坐标
    public function set y(num:Number):void{
      _y = num;
      if(dis != null) dis.y = num;
      if(isListen) dispatchEvent(new DotEvent(DotEvent.DOT_CHANGE,true));
    }

    //设置 Y 坐标
    public function get y():Number{
      updata();
      return _y;
    }
  }
}
```

事件类的代码：

```
package index.base.events{

  import flash.events.Event;
```

```
  public class DotEvent extends Event{

    public static const DOT_CHANGE:String = "dotChange";

    public function DotEvent(type:String, bubbles:Boolean = false, cancelable:Boolean = false){
      super(type,bubbles,cancelable);
    }
  }
}
```

ActionScript 3.0 自写类整理笔记（三）——ByteLoader类

作者：小 S

文章来源：http://www.xiaos8.com/default.asp?cat=1&page=12

该类的主要功能是把 swf，jpg，png，gif 等文件以字节的形式加载进来
以便于使用 Loader.loadBytes 方法，重复加载使用素材
如果图片格式为 jpg，并且是渐进式格式 jpeg，那么该类还可以帮助你边加载边显示

index.base.net.byteLoader 类讲解：
基本功能按字节加载图片，swf 等

构造函数

```
public function ByteLoader(url:String = "")
```

如果传入了参数 url，则立即执行加载！

load 加载方法

```
public function load(_url:String):void
```

开始加载，_url 是加载的地址

updata 更新数据方法

```
public function updata():void
```

更新缓冲区的可读字节

close 关闭方法

```
public function close():void
```

类使用完毕，清除所有无用的数据，也可以用来强行关闭数据流，停止下载

data 属性

```
public var data:ByteArray
```

返回加载的字节

url 属性

```
public var url:String
```

返回加载的 url

isLoad 属性（只读）

```
public function get isLoad():Boolean
```

返回是否有数据在加载

ProgressEvent.PROGRESS 事件

加载的过程中调度，并附带加载情况

Event.COMPLETE 事件

加载完毕调度

例子：

```
import index.base.net.ByteLoader;

var bl:ByteLoader = new ByteLoader;
bl.load("http://www.xiaos8.com/uploads/pro/50preso3a2.swf");
bl.addEventListener(Event.COMPLETE,completeFun);
bl.addEventListener(ProgressEvent.PROGRESS,progressFun);

function completeFun(e:Event):void{
  var loader:Loader = new Loader;
  loader.loadBytes(bl.data);
  addChild(loader);
  bl.removeEventListener(Event.COMPLETE,completeFun);
  bl.removeEventListener(ProgressEvent.PROGRESS,progressFun);
  bl.close();
  bl = null;
}

function progressFun(e:ProgressEvent):void{
  trace(e.bytesLoaded);
  //如果是渐进式格式的 jpeg 图片，那么在发布这个事件的时候读取字节，用 Loader.loadBytes 加载，就可以形成边加载边显
示
}
```

源代码：

```
package index.base.net{

  import flash.events.EventDispatcher;
  import flash.events.ProgressEvent;
  import flash.events.Event;
  import flash.utils.ByteArray;
  import flash.net.URLStream;
  import flash.net.URLRequest;

  public class ByteLoader extends EventDispatcher{

    public var url:String;
    public var data:ByteArray;
    private var stream:URLStream;

    public function ByteLoader(url:String = ""){
      if(url != ""){
        load(url);
      }
    }

    //加载
    public function load(_url:String):void{
```

```
      url = _url;
      data = new ByteArray;
      stream = new URLStream;
      stream.load(new URLRequest(url));
      stream.addEventListener(Event.COMPLETE,completeFun);
      stream.addEventListener(ProgressEvent.PROGRESS,progressFun);
    }

    //加载中
    private function progressFun(e:ProgressEvent):void{
      if(stream.bytesAvailable == 0) return;
      updata();
      dispatchEvent(e);
    }

    //加载完成
    private function completeFun(e:Event):void{
      stream.removeEventListener(Event.COMPLETE,completeFun);
      stream.removeEventListener(ProgressEvent.PROGRESS,progressFun);
      updata();
      if(isLoad) stream.close();
      dispatchEvent(e);
    }

    //更新数据
    public function updata():void{
      if(isLoad) stream.readBytes(data,data.length);
    }

    //清除数据
    public function close():void{
      if(isLoad) stream.close();
      stream = null;
      data = null;
    }

    //获取是否有数据在加载
    public function get isLoad():Boolean{
      if(stream == null) return false;
      return stream.connected;
    }
  }
}
```

ActionScript 3.0 自写类整理笔记（四）——ClassLoader类

作者：小 S
文章来源：http://www.xiaos8.com/default.asp?cat=1&page=12

主要用途：
1、在用 flash 做项目的时候，把一些元件，通过设置链接类，然后使用这个类，通过 getClass 方法即可把这个素材拿下来

图片名称：**图 1（详见例 1）**

图片地址：**uploads/200804/02_230327_1.jpg**

2、将许多许多的类分库到不同的 swf 中，然后通过调用 swf，达到调用类库的功能，然后通过 getClass 来获取类（详见例 2）

index.base.net.ClassLoader 类讲解：
加载 swf 文件，并且通过调用 getClass 获取类，也可以获取命名空间或者函数的定义

构造函数
public function ClassLoader(obj:Object = null,lc:LoaderContext = null)
参数 1 可以是字符串，可以是 ByteArray，如果为前者则采用 load 方法，后者采用 loadBytes 方法
参数 2 是创建带有指定 LoaderContext 对象的 ClassLoader 实例，LoaderContext 可以设置是否加载跨域文件，应用程序域等，详见官方 LoaderContext 类讲解！

load 方法
public function load(_url:String,lc:LoaderContext = null):void
加载文件
参数 1 是加载地址，参数 2 见构造函数

loadBytes 方法

public function loadBytes(bytes:ByteArray,lc:LoaderContext = null):void

加载字节文件

参数 1 是字节数据，参数 2 见构造函数

getClass 方法

public function getClass(className:String):Object

获取一个公共定义，可以是类，也可以是命名空间或者函数定义

参数 1 是获取 class 的完整包加类名，比如我们的这个类完整定义名称是 index.base.net.ClassLoader

详见例子

hasClass 方法

public function hasClass(className:String):Boolean

返回是否含有该公共定义

参数 1 见 getClass 方法

clear 方法

public function clear():void

清空

url 属性

public var url:String

获取 url 属性

loader 属性

public var loader:Loader

获取 loader 属性

/***/

例子 1：

图片名称：**图 2**

名称	链接
元件 8	导出: vary
元件 7	导出: rightwalk
元件 5	导出: loading
元件 4	导出: leftwalk
元件 3	导出: eat
元件 2	导出: drag
元件 1	导出: chat
元件 6	导出: normal

图片地址：**uploads/200804/02_232220_w.jpg**

这是一个虚拟人物形象的动作包，其中包含了 8 种不同的动作
在使用 ClassLoader 加载这个 swf 的动作包后，即可使用 getClass 来调用这些素材，而且可以重复的 new 这些元件，来达到多次重复使用

```
import index.base.net.ClassLoader;

var cl:ClassLoader = new ClassLoader;
cl.load("main.swf");

cl.addEventListener(Event.COMPLETE,fun);

function fun(e:Event){
  var tmp = cl.getClass("drag");
  addChild(new tmp);
}
/*******************************************/
```

例子 2：
将设我有一个类库，有这么三个类

```
import index.base.net.ByteLoader;
import index.base.net.ClassLoader;
import index.base.geom.Dot;

var bl:ByteLoader;
var cl:ClassLoader;
var dot:Dot;
```

然后把它编译成 swf

我们另外建一个文件，来加载这个所谓的类库

```
import index.base.net.ClassLoader;

var cl:ClassLoader = new ClassLoader;
cl.load("main.swf");

cl.addEventListener(Event.COMPLETE,fun);

function fun(e:Event){
  var tmp1 = cl.getClass("index.base.net.ByteLoader");
  trace(tmp1)

  var tmp2 = cl.getClass("index.base.net.ClassLoader");
  trace(tmp2)

  var tmp3 = cl.getClass("index.base.geom.Dot");
  trace(tmp3)
```

```
}

/**
 * trace 的结果：
 * [class ByteLoader]
 * [class ClassLoader]
 * [class Dot]
 */
```

我们的目的就达到了！

```
/*********************************************/
```

接下来是源代码！

```
package index.base.net{

  import flash.display.Loader;
  import flash.net.URLRequest;
  import flash.utils.ByteArray;
  import flash.events.Event;
  import flash.events.ProgressEvent;
  import flash.events.EventDispatcher;
  import flash.system.LoaderContext;

  public class ClassLoader extends EventDispatcher{

    public var url:String;
    public var loader:Loader;

    //构造函数
    public function ClassLoader(obj:Object = null,lc:LoaderContext = null) {
      if(obj != null){
        if(obj is ByteArray){
          loadBytes(obj as ByteArray,lc);
        }else if(obj is String){
          load(obj as String,lc);
        }else{
          throw new Error("参数错误，构造函数第一参数只接受 ByteArray 或 String");
        }
      }
    }

    //加载
    public function load(_url:String,lc:LoaderContext = null):void{
      url = _url;
      loader = new Loader;
      loader.load(new URLRequest(url),lc);
      addEvent();
    }

    //加载字节
```

```
    public function loadBytes(bytes:ByteArray,lc:LoaderContext = null):void{
      loader = new Loader;
      loader.loadBytes(bytes,lc);
      addEvent();
    }

    //开始侦听
    private function addEvent():void{
      loader.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS,progressFun);
      loader.contentLoaderInfo.addEventListener(Event.COMPLETE,completeFun);
    }

    //结束侦听
    private function delEvent():void{
      loader.contentLoaderInfo.removeEventListener(ProgressEvent.PROGRESS,progressFun);
      loader.contentLoaderInfo.removeEventListener(Event.COMPLETE,completeFun);
    }

    //加载成功，发布成功事件
    private function completeFun(e:Event):void {
      delEvent();
      dispatchEvent(e);
    }

    //加载过程
    private function progressFun(e:ProgressEvent):void{
      dispatchEvent(e);
    }

    //获取定义
    public function getClass(className:String):Object {
      return loader.contentLoaderInfo.applicationDomain.getDefinition(className);
    }

    //是否含有该定义
    public function hasClass(className:String):Boolean {
      return loader.contentLoaderInfo.applicationDomain.hasDefinition(className);
    }

    //清除
    public function clear():void{
      loader.unload();
      loader = null;
    }
  }
}
```

ActionScript 3.0 自写类整理笔记（五）——ImageLoader类（附加篇）

作者：小 S

文章来源：http://www.xiaos8.com/default.asp?cat=1&page=11

跟 ClassLoader 差不多，但是不同的是，他是读取图片的 BitmapData，然后可以多次
new Bitamap(ImagesLoader.data)
进行图片调用

多次使用图片，直接用 ByteLoader 也可以，但是他加载进来的是字节，还要通过 loadBytes
但是加载进来的，却是一张图片，无法重复使用
虽说要用就 loadBytes 一下，就是一张图片，但是实际起来还是比较麻烦
那么这个类，就是帮助你把这些步骤全部省下，直接把 BitmapData 拿出来
你只需要，用一个引用值接住他
var bd:BitmapData = ImageLoader.data;
然后每次使用这张图片的时候
new Bitamap(ImagesLoader.data)
就行了

该类的具体用法就不详讲了，前面加过的 ClassLoader 类，在这个类里面都有，而且两个类连代码都没改什么。。
就是少了 getClass,hasClass，而多了 data 属性。相信应该不是很难看懂

示例：

```
import index.base.net.ImageLoader;

var il:ImageLoader = new ImageLoader;

il.load("http://www.xiaos8.com/uploads/200804/02_230327_1.jpg");

il.addEventListener(Event.COMPLETE,fun);

function fun(e:Event){
  addChild(new Bitmap(il.data));
  trace(il.loader)
  trace(il.url)
}
```

源代码：

```
package index.base.net{

  import flash.display.Loader;
  import flash.display.BitmapData;
  import flash.net.URLRequest;
  import flash.utils.ByteArray;
  import flash.events.Event;
  import flash.events.ProgressEvent;
```

```
import flash.events.EventDispatcher;
import flash.system.LoaderContext;

public class ImageLoader extends EventDispatcher{

  public var url:String;
  public var loader:Loader;
  public var data:BitmapData;

  //构造函数
  public function ImageLoader(obj:Object = null,lc:LoaderContext = null) {
    if(obj != null){
      if(obj is ByteArray){
        loadBytes(obj as ByteArray,lc);
      }else if(obj is String){
        load(obj as String,lc);
      }else{
        throw new Error("参数错误，构造函数第一参数只接受 ByteArray 或 String");
      }
    }
  }

  //加载
  public function load(_url:String,lc:LoaderContext = null):void{
    url = _url;
    loader = new Loader;
    loader.load(new URLRequest(url),lc);
    addEvent();
  }

  //加载字节
  public function loadBytes(bytes:ByteArray,lc:LoaderContext = null):void{
    loader = new Loader;
    loader.loadBytes(bytes,lc);
    addEvent();
  }

  //开始侦听
  private function addEvent():void{
    loader.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS,progressFun);
    loader.contentLoaderInfo.addEventListener(Event.COMPLETE,completeFun);
  }

  //结束侦听
  private function delEvent():void{
    loader.contentLoaderInfo.removeEventListener(ProgressEvent.PROGRESS,progressFun);
    loader.contentLoaderInfo.removeEventListener(Event.COMPLETE,completeFun);
  }
```

```
    //加载成功，发布成功事件
    private function completeFun(e:Event):void {
      data = loader.content["bitmapData"];
      delEvent();
      dispatchEvent(e);
    }

    //加载过程
    private function progressFun(e:ProgressEvent):void{
      dispatchEvent(e);
    }

    //清除
    public function clear():void{
      loader.unload();
      loader = null;
      data = null;
    }
  }
}
```

ActionScript 3.0 自写类整理笔记（六）——疑难杂症汇总

作者：小 S

文章来源：http://www.xiaos8.com/default.asp?cat=1&page=11

在做项目的时候，总会碰到很多，奇奇怪怪的问题，看了这篇文章相信会有些帮助，虽然不是很全面，但只要是我碰到过的问题，就分享给大家！但是也怕有些想不起了。。尽力额、、

========== 气 死 你 的 分 割 线 ==========

1、数组的排序问题：Array.sort()方法

大家都经常用排序，一般都是字符串排序什么的，都不会发现什么的，现在来看一个有趣的现象

```
var a:Array = [7,3,32,64,96,13,42];
a.sort();
trace(a);
//输出：13,3,32,42,64,7,96
```

奇怪了，为什么排序出来的东西是个这样的。。

分析为什么：仔细看一下，会发现每个数字的第一个数字，的确是按排序放好的：1，3，3，4，6，7，9

为什么会这样呢？仔细来看看 Array 的官方文档

Array.sort()官方解释

默认情况下，Array。sort() 按以下方式进行排序：

1、排序区分大小写（Z 优先于 a）。

2、按升序排序（a 优先于 b）。

3、修改该数组以反映排序顺序；在排序后的数组中不按任何特定顺序连续放置具有相同排序字段的多个元素。

4、元素无论属于何种数据类型，都作为字符串进行排序，所以 100 在 99 之前，这是因为 "1" 的字符串值小于 "9" 的字符串值。

不难发现，第 4 条说明了，默认是以字符串进行排序，而不是以字符串排序，难怪会出现这样的结果额

那么应该如何排序才会出现我们要的答案呢？

```
var a:Array = [7,3,32,64,96,13,42];
a.sort(Array.NUMERIC);
trace(a);
//输出：3,7,13,32,42,64,96
```

加的这个参数是什么东西额。。仔细看看帮助文档吧。。

然后 Array 提供了几个常量分别是：

Array 官方文档

CASEINSENSITIVE : uint = 1 [static] 指定 Array 类排序方法为不区分大小写的排序。

DESCENDING : uint = 2 [static] 指定 Array 类排序方法为降序排序。

NUMERIC : uint = 16 [static] 指定 Array 类排序方法为数值（而不是字符串）排序。

RETURNINDEXEDARRAY : uint = 8 [static] 指定排序返回的数组包含数组索引。

UNIQUESORT : uint = 4 [static] 指定 Array 类排序方法的唯一排序要求。

========== 气 死 你 的 分 割 线 ==========

2、Loader.load 加载图片，显示不出来

这个问题可以一笔带过额。。很多人都碰到过，其实是因为你没有加 checkPolicyFile
如果确定图片跨域了，那么加载图片的时候，应该这样写

```
var loader:Loader = new Loader;
var request:URLRequest= new URLRequest(url);
var lc:LoaderContext = new LoaderContext(true);
loader.load(request, lc):
```

加的这个 new LoaderContext(true)，可以去参考官方帮助文档
这个构造函数可以传三个参数
第一参数：指定是否去加载跨域文件
第二参数：指定要使用的应用程序域
第三参数：指定要使用的安全沙箱

========== 气 死 你 的 分 割 线 ==========

3、flash 嵌入代码

经常出问题，比如打开窗口会被阻止，flash 游戏做方向键的，屏幕会跟着动等类似问题
我也经常碰到这样的问题，花时间研究了一下
注意第一个，嵌入代码中有一个使 flash 透明的参数，不要把它设置成透明，那么很多问题都可以解决，不信的话，大家可以试试，还有很多不常用的属性，有些可以阻止 flash 某些方法的使用，很烦人的，特别是某些博客，公开平台等，传 flash 之后，他在嵌入代码中会加一些这样的属性，那么 flash 有很多事情都不能做
另外还有一个就是，如果是从 flash 中弹出新窗口，设置成不透明也会被阻止，但是在当前窗口打开，就不会
这是我在做项目的时候，碰到的问题，以及我的解决方法，如果有高人有更好的办法，一定要告诉我额。。。

========== 气 死 你 的 分 割 线 ==========

4、mask 遮罩层

mask 这个属性相信很多人看过帮助文档之后，不会仔细去看，特别是熟悉 AS2 的 setMask 的人
当然包括我自己也放过这样的错误，因此在此提起，希望各位新手在学习的时候，把帮助文档看清楚

AS3 帮助文档中 DisplayObject 类的 mask 属性

mask 属性
mask:DisplayObject [read-write]
语言版本 : ActionScript 3.0
Player 版本 : Flash Player 9
调用显示对象被指定的 mask 对象遮罩。 要确保当舞台缩放时蒙版仍然有效，mask 显示对象必须处于显示列表的活动部分。 但不绘制 mask 对象本身。 将 mask 设置为 null 可删除蒙版。
要能够缩放遮罩对象，它必须在显示列表中。 要能够拖动蒙版 Sprite 对象（通过调用其 startDrag() 方法），它必须在显示列表中。 要为基于 sprite 正在调度的 mouseDown 事件调用 startDrag() 方法，请将 sprite 的 buttonMode 属性设置为 true。

根据帮助文档的说法，如果要缩放遮罩层，就必须把遮罩层放在相应的显示列表中，也就是 addChild(mask)
比如：我现在使用的是

```
stage.scaleMode = StageScaleMode.SHOW_ALL;
```

如果说你的遮罩层没有 addChild 那么，在缩放窗口的时候，mask 是不会改变宽高；反之加入了显示列表，就会跟着窗口一起缩放
mask 的拖拽和点击等事件也是如此！
综上所述：设置遮罩层最佳做法是放到显示列表，当然不排除有特别作用的

另外，你把 mask 的属性清空，本来被做为遮罩层的，就可以看的到了，否则是看不到遮罩层的

========== 气 死 你 的 分 割 线 ==========

5、莫名其妙的 mc
你使用 Loader 加载进来的 mc，如果里面是动画，或者带有声音
你加载进来，就算你不 addChild，他也在播放的，而且有时候你控制他 stop，还控制不了。。
解决方案是：加 stop()，做动画素材的时候，尽量把关键动画都做在主场景帧上，那么程序中就可以控制动画了
置于声音，还真是个头疼的东西，当然使用 mc 的同样管理方案也是可以的，但是我建议，把声音做到库中
然后使用 ClassLoader 把声音拿下来，然后用程序控制声音。

========== 气 死 你 的 分 割 线 ==========

6、幽灵般的容器
一个容器，当被 addChild 后，你就可以直观的看得到他，但是你又 removeChild，很多人都认为这样，容器就不存在了，实则不然，他还存在
比如你做一个飞机游戏，飞机爆炸后，你就会 removeChild 它，但是这样做是不够的，他依然还存在那个位置，而且 x，y 值是依然不变的
如果飞机上还写了碰撞检测，如果这时候有子弹飞过来，依然会算碰撞成功，还有很多类似的例子
个人建议：两个办法，简单的游戏嘛，可以设个全局逻辑值，当确定不需要在碰撞检测，把这个值变成 false，每次检测碰撞，也检查一下这个值，就行了
第二个办法是复杂一点的，扩展容器类，侦听他的 removedFromStage，removed 事件和 addedToStage，added 事件，如果加入了，则容器中某个值等于 true，反之是 false
这个属性就可以叫 返回是否被加入显示对象
当然还有其他的办法，比如检查容器的 contains 方法，可以检查出当前容器中是否含有容器，如果 removeChild，那么你检查结果就是不存在

========== 气 死 你 的 分 割 线 ==========

oh。。先写这么多，肯定不是很完全。。一下子我也没办法把我在项目中碰到的任何问题搬出来。。想起来，我在继续写。。

ActionScript 3.0 自写类整理笔记（七）——OutDisplay类

作者：小 S

文章来源：http://www.xiaos8.com/default.asp?cat=1&page=11

点击播放/隐藏媒体

/uploads/pro/OutDisplay.swf

index.base.func.OutDisplay 类讲解：

当对象再一次触发某事件的时候，判断是否在事件范围内，如果不在则调度 Event.CANCEL 事件

构造函数

public function OutDisplay(_type:String,_dis:DisplayObject,...objs)

第一个参数：侦听类型

第二个参数：作用范围，一般来说都是用 Stage 作为范围，当然也有用在其他地方的

第三，四……个参数：可以拥有_type 事件的对象

实例化后，侦听_dis 的_type 事件，每当触发_type 事件，就开始判断事件目标是否存在 objs 中，或者 objs 中某项的子集显示对象，如果有则没有反应，如果没有则发出 Event.CANCEL 事件

add 方法

public function add(...objs):Array

增加可以拥有_type 事件的对象

objects 属性（只读）

public function get objects():Array

返回可以拥有_type 事件的对象列表

clear 方法

public function clear(isDispatch:Boolean = false):void

清除类里面的侦听，以及引用

第一个参数：如果为 true，那么调用 clear 之后会发出 Event.CANCEL，反之没反应！默认为 false

例子：

下面是展示 flash 的源代码，这只是一个简单应用，如果写复杂一点，也可以写出真正的快捷菜单

```
import index.base.func.OutDisplay;

var menu:Menu = new Menu;
menu.mc1.addEventListener(MouseEvent.MOUSE_OVER,mouseOverFun);
menu.mc2.addEventListener(MouseEvent.MOUSE_OVER,mouseOverFun);
menu.mc3.addEventListener(MouseEvent.MOUSE_OVER,mouseOverFun);
menu.mc4.addEventListener(MouseEvent.MOUSE_OVER,mouseOverFun);
addChild(menu);

function mouseOverFun(e:MouseEvent){
  var m:Menu = new Menu;
  m.x = e.currentTarget.x + e.currentTarget.parent.x;
```

```
  m.y = e.currentTarget.y + e.currentTarget.parent.y;
  addChild(m);

  var out:OutDisplay = new OutDisplay(MouseEvent.MOUSE_OVER,stage,e.currentTarget);
  out.addEventListener(Event.CANCEL,cancelFun);
  out.add(m);

  m = null;
  out = null;
}

function cancelFun(e:Event):void{
  var out:OutDisplay = e.currentTarget as OutDisplay;
  var tmpAr:Array = out.objects;
  removeChild(tmpAr[1]);
  out.removeEventListener(Event.CANCEL,cancelFun);

  tmpAr = null;
  out = null;
}
```

类的源代码：

```
package index.base.func{

  import flash.display.DisplayObject;
  import flash.display.DisplayObjectContainer;
  import flash.events.Event;
  import flash.events.EventDispatcher;

  public class OutDisplay extends EventDispatcher{

    //保存所有对象
    private var objAr:Array;
    //保存侦听类型
    private var type:String;
    //保存侦听范围
    private var dis:DisplayObject;

    //构造函数
    public function OutDisplay(_type:String,_dis:DisplayObject,...objs){
      type = _type;
      objAr = objs;
      dis = _dis;
      dis.addEventListener(_type,fun);
    }

    //事件处理
    private function fun(e:Event):void{
      for(var i:uint = 0;i<objAr.length;i++){
```

```
        if(e.target == objAr[i]) return;
        if(objAr[i] is DisplayObjectContainer){
          if(objAr[i].contains(e.target)) return;
        }
      }
      dis.removeEventListener(type,fun);
      dispatchEvent(new Event(Event.CANCEL));
    }

    //添加对象
    public function add(...objs):Array{
      for(var i:int = 0; i < objs.length; i ++){
        objAr.push(objs[i]);
      }
      return objects;
    }

    //获取对象列表
    public function get objects():Array{
      return objAr;
    }

    //卸载
    public function clear(isDispatch:Boolean = false):void{
      if(isDispatch) dispatchEvent(new Event(Event.CANCEL));
      dis.removeEventListener(type,fun);
      objAr = null;
    }

  }
}
```

ActionScript 3.0 自写类整理笔记（八）——Direction类和Dot类

作者：小 S

文章来源：http://www.xiaos8.com/default.asp?cat=1&page=11

点击播放/隐藏媒体
/uploads/pro/dir.swf

关于该例子的教程请关注第九篇笔记！
因为即将出 Direction 类与 Dot 类的实战使用教程，因此本篇文章只对 Direction 类的方法属性讲解和 Dot 的更新部分讲解

首先是
index.base.game.Direction 类
作用：控制飞机游戏，坦克游戏，或者一些和方向有关的方向按键操作

构造函数：
```
public function Direction(_area:InteractiveObject,isSole:Boolean = false,_up:uint = 38,_down:uint = 40,_left:uint = 37,_right:uint = 39)
```
参数一：方向键的作用区域，如果_area 当前不是焦点，那么是侦听不到键盘事件的，一般这儿都是使用 Stage 做为作用区域
参数二：是否为单向事件触发，如果为 false，那么按了什么键就是什么，可以同时触发上和左等两个或者两个以上的事件，反之以最后按的那个键为准
参数三，四，五，六：按键的键值，默认为 38，40，37，39，分别是方向键的上下左右！

start 方法：
```
public function start():void
```
开始捕获事件，当触发构造函数，将自动执行 start 方法

stop 方法：
```
public function stop():void
```
停止捕获事件

setKey 方法：
```
public function setKey(num:uint,vars:uint):void
```
设置按键键值
参数一：方向键标识，请参考该类的常量属性
参数二：按键键值

常量属性：
```
public static const UP:uint = 0;
public static const DOWN:uint = 1;
public static const LEFT:uint = 2;
public static const RIGHT:uint = 3;
```
分别代表：上下左右的方向键标识

clear 方法：

public function clear():void

清除所有方向记录

area 属性：

public var area:InteractiveObject

返回作用区域

sole 属性：

public var sole:Boolean

返回是否单向操作

DirectionEvent.DO 事件：

当有方向键是按下去的时候，则会发布事件，事件中含有 up,down,left,right，4 个属性，分别表示哪几个键是按下去的！

========== 气 死 你 的 分 割 线 ==========

Dot 类在前面的整理笔记中，曾经说过，这次是更新类的方法和属性

增加了旋转属性，并且可以计算当前方向的某距离后的点

以下只对更新的方法和属性进行讲解：其他的请看老的整理笔记：

go 方法：

public function go(num:Number,isChange:Boolean = false):Dot

参数一，表示面向旋转方向前进多少的距离

参数二，表示是否也跟新该点基于 num 变化之后的点坐标

clear 方法：

public function clear():void

清空绑定对象的引用

r 属性：

public function set r(num:Number):void

public function get r():Number

旋转属性的设置，如果 isListener 值为真，则改变旋转值会触发 R_CHANGE 的事件

Direction 类源代码：

```
package index.base.game{

  import flash.events.EventDispatcher;
  import flash.events.KeyboardEvent;
  import flash.events.Event;
  import flash.display.InteractiveObject;

  import index.base.events.DirectionEvent;
```

```
 public class Direction extends EventDispatcher{

   //方向表示
   public static const UP:uint = 0;
   public static const DOWN:uint = 1;
   public static const LEFT:uint = 2;
   public static const RIGHT:uint = 3;

   //作用区域
   public var area:InteractiveObject;
   //是否单向
   public var sole:Boolean;

   //上下左右键值
   private const directionAr:Array = new Array(4);

   //是否上下左右
   private var _up:Boolean = false;
   private var _down:Boolean = false;
   private var _left:Boolean = false;
   private var _right:Boolean = false;

   public function Direction(_area:InteractiveObject,isSole:Boolean = false,_up:uint = 38,_down:
uint = 40,_left:uint = 37,_right:uint = 39){
     area = _area;
     sole = isSole;
     directionAr[UP] = _up;
     directionAr[DOWN] = _down;
     directionAr[LEFT] = _left;
     directionAr[RIGHT] = _right;
     start();
   }

   //开始获取事件
   public function start():void{
     area.addEventListener(KeyboardEvent.KEY_DOWN,onKeyDown);
     area.addEventListener(KeyboardEvent.KEY_UP,onKeyUp);
     area.addEventListener(Event.ENTER_FRAME,onEnterFrame);
   }

   //事件帧频繁触发
   private function onEnterFrame(e:Event):void{
     var num:uint = Number(_up) + Number(_down) + Number(_left) + Number(_right);
     if(num == 0){
       return;
     }

     var eve:DirectionEvent = new DirectionEvent(DirectionEvent.DO);
```

```
    eve.up = _up;
    eve.down = _down;
    eve.left = _left;
    eve.right = _right;
    dispatchEvent(eve);
  }

  //停止获取事件
  public function stop():void{
    area.removeEventListener(KeyboardEvent.KEY_DOWN,onKeyDown);
    area.removeEventListener(KeyboardEvent.KEY_UP,onKeyUp);
    area.removeEventListener(Event.ENTER_FRAME,onEnterFrame);
  }

  //鼠标按下去事件
  private function onKeyDown(e:KeyboardEvent):void{
    key(e.keyCode,true)
  }

  //鼠标弹上来事件
  private function onKeyUp(e:KeyboardEvent):void{
    key(e.keyCode,false)
  }

  //变化状态
  private function key(num:uint,isDown:Boolean):void{
    switch(num){
      case directionAr[UP]:
        if(sole) clear();
        _up = isDown;
      break;
      case directionAr[DOWN]:
        if(sole) clear();
        _down = isDown;
      break;
      case directionAr[LEFT]:
        if(sole) clear();
        _left = isDown;
      break;
      case directionAr[RIGHT]:
        if(sole) clear();
        _right = isDown;
      break;
    }
  }

  //设置按钮
  public function setKey(num:uint,vars:uint):void{
    directionAr[num] = vars;
```

```
    }

    //清空按键
    public function clear():void{
      _up = _down = _left = _right = false;
    }
  }
}
```

DirectionEvent 类源代码：

```
package index.base.events{

  import flash.events.Event;

  public class DirectionEvent extends Event{

    public var up:Boolean;
    public var down:Boolean;
    public var left:Boolean;
    public var right:Boolean;

    public static const DO:String = "do";

    public function DirectionEvent(type:String){
      super(type);
    }
  }
}
```

Dot 类源代码：

```
package index.base.geom{

  import flash.events.EventDispatcher;
  import flash.display.DisplayObject;

  import index.base.events.DotEvent;

  public class Dot extends EventDispatcher{

    private var _x:Number;
    private var _y:Number;
    private var _r:Number;
    private var dis:DisplayObject;

    public var isListen:Boolean;

    public function Dot(x_:Number = 0,y_:Number = 0,r_:Number = 0,_isListen:Boolean = false){
      _x = x_;
      _y = y_;
```

```
    _r = r_;
    isListen = _isListen;
  }

  //绑定 DisplayObject
  public function bind(_dis:DisplayObject,isInTime:Boolean = false):void{
    dis = _dis;
    updata();
    if(isInTime) dis.addEventListener("enterFrame",enterFrameFun);
  }

  //帧频繁事件
  private function enterFrameFun(e:Object):void{
    if(_x != dis.x) x = dis.x;
    if(_y != dis.y) y = dis.y;
    if(_r != dis.rotation) r = dis.rotation;
  }

  //更新 xy 数据
  public function updata():void{
    if(dis != null){
      _x = dis.x;
      _y = dis.y;
      _r = dis.rotation;
    }
  }

  //计算该点向 R 方向前进某距离后的点
  public function go(num:Number,isChange:Boolean = false):Dot{
    updata();
    var yx:Number = Math.tan(_r * Math.PI / 180);
    var tmpx:Number = num / Math.sqrt(Math.pow(yx,2) + 1);
    var tmpy:Number = tmpx * yx;
    var n:int = Number(Math.abs(_r) <= 90) * 2 - 1;
    var dot:Dot = new Dot(_x + tmpx * n,_y + tmpy * n,_r);
    if(isChange){
      x = dot.x;
      y = dot.y;
    }
    return dot;
  }

  //计算该点与另外一点的距离
  public function from(_dot:Dot,isQuadrant:Boolean = false):Number{
    updata();
    var num:Number = Math.sqrt(Math.pow(_dot.x - _x,2) + Math.pow(_dot.y - _y,2));
    if(!isQuadrant) num = Math.abs(num);
    return num;
  }
```

```
//计算该点与另外一点所形成的线段与水平线的夹角,按顺时间计算
public function angle(_dot:Dot,isRadian:Boolean = false):Number{
  updata();
  var numx:Number = _dot.x - _x;
  var numy:Number = _dot.y - _y;
  var num:Number = Math.atan(numy/numx);
  if(!isRadian) num = num * 180 / Math.PI;
  return num;
}

//返回当前点处在另外一点的哪个象限中 或 返回另外一点处在当前点的哪个象限中
public function quadrant(_dot:Dot,isMaster:Boolean = true):int{
  updata();
  if(_x == _dot.x || _y == _dot.y){
    return 0;
  }

  var num:int;
  var p1:Boolean = (_x - _dot.x) > 0;
  var p2:Boolean = (_y - _dot.y) > 0;
  num = isMaster ? (p1 ? (p2 ? 2 : 3) : (p2 ? 1 : 4)) : (p1 ? (p2 ? 4 : 1) : (p2 ? 3 : 2));

  return num;
}

//返回该点距 0 点的距离
public function get length():Number{
  updata();
  var num:Number = Math.sqrt(Math.pow(_x,2) + Math.pow(_y,2));
  return num;
}

//清除显示对象
public function clear():void{
  dis = null;
}

//改变旋转值
public function set r(num:Number):void{
  _r = num;
  if(dis != null) dis.rotation = num;
  if(isListen) dispatchEvent(new DotEvent(DotEvent.R_CHANGE,true));
}

//改变旋转值
public function get r():Number{
  updata();
  return _r;
```

```
    }

    //改变X坐标
    public function set x(num:Number):void{
      _x = num;
      if(dis != null) dis.x = num;
      if(isListen) dispatchEvent(new DotEvent(DotEvent.X_CHANGE,true));
    }

    //设置X坐标
    public function get x():Number{
      updata();
      return _x;
    }

    //改变Y坐标
    public function set y(num:Number):void{
      _y = num;
      if(dis != null) dis.y = num;
      if(isListen) dispatchEvent(new DotEvent(DotEvent.Y_CHANGE,true));
    }

    //设置Y坐标
    public function get y():Number{
      updata();
      return _y;
    }
  }
}
```

DotEvent 类源代码：

```
package index.base.events{

  import flash.events.Event;

  public class DotEvent extends Event{

    public static const X_CHANGE:String = "xChange";
    public static const Y_CHANGE:String = "yChange";
    public static const R_CHANGE:String = "rChange";

    public function DotEvent(type:String, bubbles:Boolean = false, cancelable:Boolean = false){
      super(type,bubbles,cancelable);
    }
  }
}
```

ActionScript 3.0 自写类整理笔记（九）——小游戏实战尝试

作者：小 S

文章来源：http://www.xiaos8.com/default.asp?cat=1&page=11

点击播放/隐藏媒体

/uploads/pro/dir.swf

这几天也写了一些类了，是驴子还是骡子，拿出来遛一遛就知道了，先看这个上面这个 flash 动画！

一个类似坦克游戏的 demo 程序
使用 Direction 类来进行方向控制
使用 Dot 类来计算距离
用上 Direction 类和 Dot 类之后，这个 demo 程序变得异常简单额。。
也没什么好说，主要透过这个例子，让大家类熟悉 Direction 类和 Dot 类的使用方法
不懂的可以在后面跟帖提问，高手如果看到什么有错误的地方，请指正出来，多谢指教

下面是 fla 的源代码：

```
import index.base.game.Direction;
import index.base.events.DirectionEvent;
import index.base.geom.Dot;

//舞台属性设置
stage.showDefaultContextMenu = false;
stage.align = "TL";
stage.scaleMode = "noScale";

//创建坦克
var tank:Tank = new Tank;
tank.x = tank.y = 250;
addChild(tank);

//创建绑定坦克的点
var dot:Dot = new Dot;
dot.bind(tank);

//坦克移动
var dirTank:Direction = new Direction(stage);
//炮台转动
var dirTower:Direction = new Direction(stage,true,87,83,65,68);

//坦克炮台事件
dirTank.addEventListener(DirectionEvent.DO,doTankFun);
dirTower.addEventListener(DirectionEvent.DO,doTowerFun);

//坦克移动
function doTankFun(e:DirectionEvent):void{
  if(e.up){
    dot.go(2,true);
```

```
  }
  if(e.down){
    dot.go(-2,true);
  }
  if(e.left){
    tank.rotation -= 2;
  }
  if(e.right){
    tank.rotation += 2;
  }
  if(tank.x < 0) tank.x = 0;
  if(tank.y < 0) tank.y = 0;
  if(tank.x > stage.stageWidth) tank.x = stage.stageWidth;
  if(tank.y > stage.stageHeight) tank.y = stage.stageHeight;
}

//是否可以发射炮台，子弹
var isBullet:Boolean = true;
var isShell:Boolean = true;
//炮台发射转动
function doTowerFun(e:DirectionEvent):void{
  if(e.up && isBullet){
    var bullet:Bullet = new Bullet;
    bullet.x = tank.x;
    bullet.y = tank.y;
    bullet.rotation = tank.rotation + tank.tower.rotation;
    bullet.addEventListener(Event.ENTER_FRAME,bulletFun);
    addChild(bullet);

    isBullet = false;
    setTimeout(function(){isBullet = true},200);
  }
  if(e.down && isShell){
    var shell:Shell = new Shell;
    shell.x = tank.x;
    shell.y = tank.y;
    shell.rotation = tank.rotation;
    shell.addEventListener(Event.ENTER_FRAME,shellFun);
    addChild(shell);

    isShell = false;
    setTimeout(function(){isShell = true},500);
  }
  if(e.left){
    tank.tower.rotation -= 5;
  }
  if(e.right){
    tank.tower.rotation += 5;
  }
```

```
}

//炮台
function shellFun(e:Event):void{
  var tmp:Shell = e.currentTarget as Shell;
  var d:Dot = new Dot(tmp.x,tmp.y,tmp.rotation);
  d.bind(tmp);
  d.go(4,true);
  if(tmp.x < 0 || tmp.x > stage.stageWidth || tmp.y < 0 || tmp.y > stage.stageHeight){
    removeChild(tmp);
    tmp.removeEventListener(Event.ENTER_FRAME,shellFun);
  }

  tmp = null;
  d = null;
}

//子弹
function bulletFun(e:Event):void{
  var tmp:Bullet = e.currentTarget as Bullet;
  var d:Dot = new Dot(tmp.x,tmp.y,tmp.rotation);
  d.bind(tmp);
  d.go(5,true);
  if(tmp.x < 0 || tmp.x > stage.stageWidth || tmp.y < 0 || tmp.y > stage.stageHeight){
    removeChild(tmp);
    tmp.removeEventListener(Event.ENTER_FRAME,bulletFun);
  }

  tmp = null;
  d = null;
}
```

另外注意源代码，有个地方多次对 tank 的 tower 属性就行引用，并且返回他的 x，y 或者旋转值，有人就会问了，as3 不是不支持类似 mc 那样的直接访问显示对象，为什么我这儿却可以？
愿意是我把素材绑定在 Tank 类上，并且对 Tank 类做了以下编写：

```
package{

  import flash.display.Sprite;

  public class Tank extends Sprite{

    public function Tank(){

    }

    public function get tower():Sprite{
      return towerMc;
    }
```

```
  }
}
```

光看这个类，也许你还是不明白，是什么原因，为什么会多出来一个towerMc出来，详细的原因，请自己下载提供的源文件，下载下来看看吧。。不懂跟帖问！

点击下载源文件

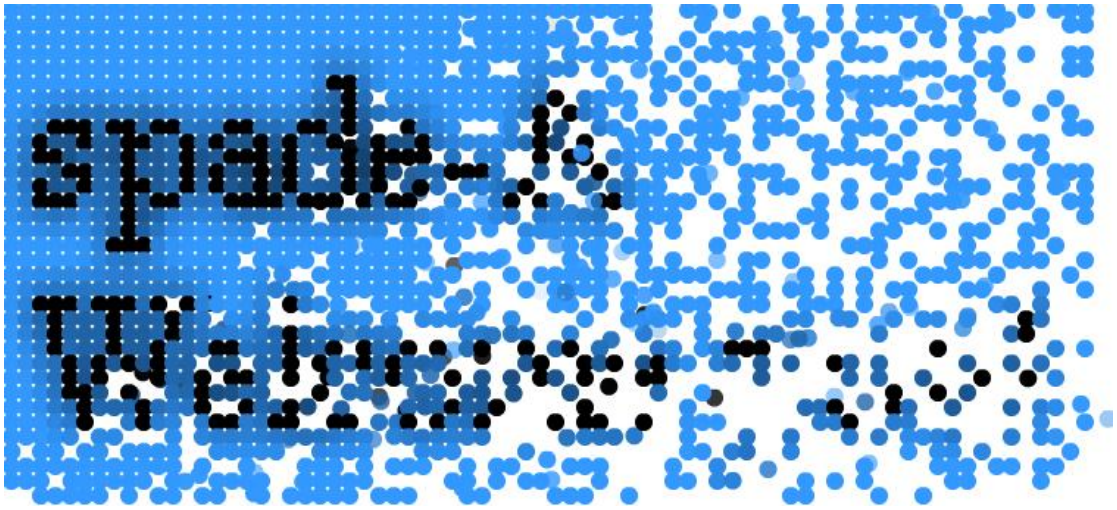
AS3 BitmapData 特效

作者：**DN_Web**

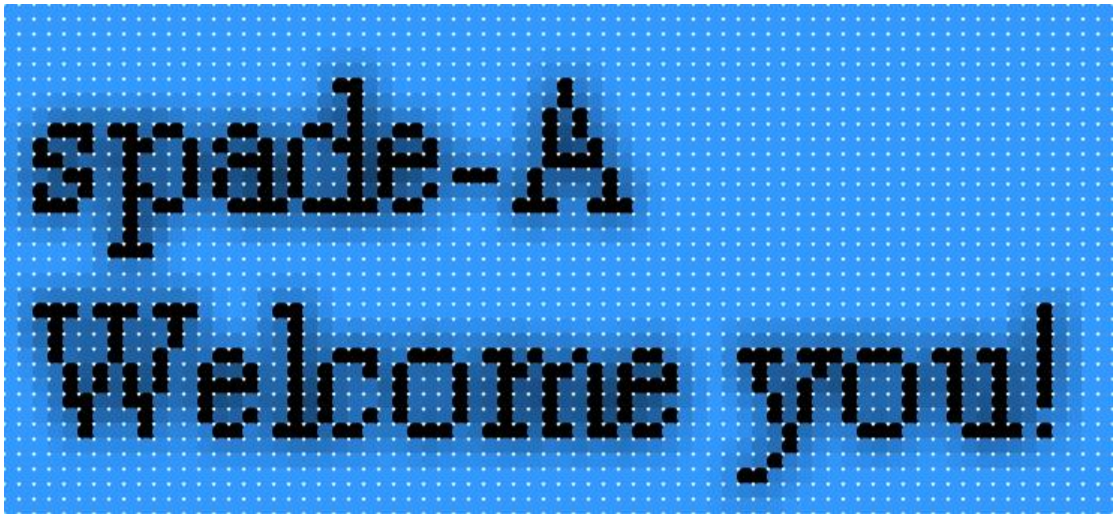
文章来源：http://hi.baidu.com/dn%5Fweb/blog/item/20adda1ac473e6138618bf89.html

BitmapData 是个很强大的东西,很多特效可以利用他来实现,当然了,高级的特效,数学算法要求很高,今天在网

上看到个效果不错,也能看懂,就拿来记录一下,效果如下:



运行完



没空间,放不上swf直观的效果,有兴趣的把下面的代码粘贴到.as文件.然后运行下.提示需要用到缓动类(Tweener),可以去code.google.com/p/tweener/downloads/list下载到

```
package{
    import flash.display.*;
    import flash.text.*;
    import flash.filters.*;
    import flash.geom.*;
    import caurina.transitions.Tweener;
```

```
    public class Foo extends Sprite{
        private var bd:BitmapData;
        public function Foo():void{
            var tf:TextField = new TextField();
            tf.textColor = 0x000000;
            tf.text = "spade-A\nWelcome you!";
            tf.autoSize = "left";
    //文本 draw 成 BitmapData
            bd = new BitmapData(tf.width, tf.height, false, 0x3399ff);
            bd.draw(tf);
    //添加滤镜
            bd.applyFilter(bd, bd.rect, new Point(), new BlurFilter());
            bd.draw(tf);

    //循环排列
            for(var i:int = 0; i < bd.width; i++){
                for(var j:int = 0; j < bd.height; j++){
                    Tweener.addTween(
                        randomize(addChild(new Circle(bd.getPixel(i, j)))), //getPixel
（）用来按照位置获取颜色
                        {
        //排列
                            x: (stage.stageWidth-tf.width*10)/2+(i * 10),
                            y: (stage.stageHeight-tf.height*10)/2+(j * 10),
                            alpha: 1,
                            delay: (i + j) * .2 * Math.random(),
                            time: 1
                        }
                    );
                }
            }
        }
   //仿粒子
        private function randomize(d:DisplayObject):DisplayObject{
           d.x = 1000 * Math.random();
            d.y = 400 * Math.random();
            d.alpha = 0;
            return d;
        }
    }
}


import flash.display.Sprite;
//元素小球
class Circle extends Sprite{
    public function Circle(color:uint):void{
        graphics.beginFill(color);
        graphics.drawCircle(0, 0, 6);
```

```
            graphics.endFill();
        }
    }
```

AS3 读取音频频谱

作者：A 闪

文章来源：http://hi.baidu.com/%B0%B5%BA%DA%B2%E0%CE%C0/blog/item/c3e88a2381b39f589822edf8.html

在做 FLASH 音乐播放器的时候，大家都希望在自己的音乐播放器中添加音频频谱的功能，来为自己的播放器平添色彩。本节我们就来学习如何制作一个音频频谱的功能。这个教程在网上非常多，笔者找到了一个写的比较精炼的，介绍给大家。

首先，我们要知道，频谱的显示形式有很多。最常见的就是柱状频谱图。除此之外还有很多形式，比如波浪，星空，线条等等。这里我们就挑选最常见最简单的“柱状频谱图”。

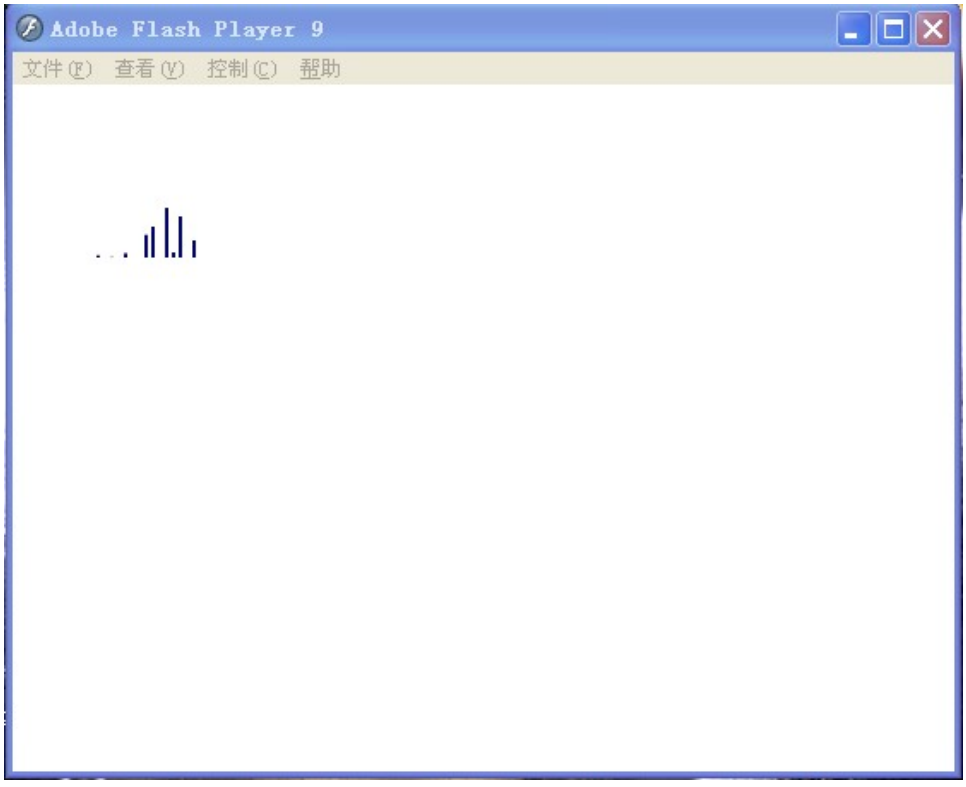
好了！新建一个空白的 FLASH 文档，然后保存，在同目录下复制一段音频（MP3 音频，你喜欢的即可），这段音频的名称我们暂时命名为“asd.mp3”。下面就是编写代码的阶段。如下：

```
package {
//导入类
//Sprite是一个只有一帧的MovieClip,相对于MovieClip少了帧和场景...
import flash.display.Sprite;
//事件类
import flash.events.Event;
//声音类
import flash.media.Sound;
//混音器类
import flash.media.SoundMixer;
//声音通道类(暂时这么翻吧)
import flash.media.SoundChannel;
//URLRequest类
import flash.net.URLRequest;
//二进制数组类
import flash.utils.ByteArray;
//定义类
public class SwfdongSound extends Sprite {
   //声明MP3 路径
   private var url:String = "asd.mp3";
   //声明一个Sound对象
   private var DongSound:Sound = new Sound();
   //声明一个SoundChannel对象
   private var sChannel:SoundChannel;
   //声明一个ByteArray对象
   private var bArray:ByteArray = new ByteArray();
   //声明一个数组对象
   private var Ary:Array;
   //声明一个数字对象
   private var n:Number = 0;
   //初始化函数
   public function SwfdongSound() {
    var req:URLRequest = new URLRequest(url);
```

```
    DongSound.load(req);
    //播放
    DongSound.play();
    //添加一个EnterFrame的侦听器,同以前的this.onEnterFrame=showBar();
    this.addEventListener(Event.ENTER_FRAME,showBar);
   }
   private function showBar(event:Event) {
    n = 0;
    //清除绘图
    this.graphics.clear();
    //将当前声音输出为ByteArray -1.0~1.0
    SoundMixer.computeSpectrum(bArray,true,0);

    for (var i=0; i < 256; i+=16) {
     //在ByteArray中读取一个 32 位的单精度浮点数(这个是livedoc上写的,实际就是把数据流读取成浮点数)
     n = bArray.readFloat();
     //这个实际作用是把n扩大一下
     var num:Number = n*100;
     //设置线条样式,颜色绿色,宽度 10,透明度 100
     this.graphics.lineStyle(2,0x04076A,100,false,"noSacle","none");
     //移动到x坐标 50+i,y坐标 100 的位置
     this.graphics.moveTo(50+i/4,100);
     //向上画图
     this.graphics.lineTo(50+i/4,100-num/2);
    }
   }
}
```

} 代码的解释都写道了脚本中了！将这个AS类文档保存，并将它设置为文档类。这样测试影片我们就可以看到效果了！

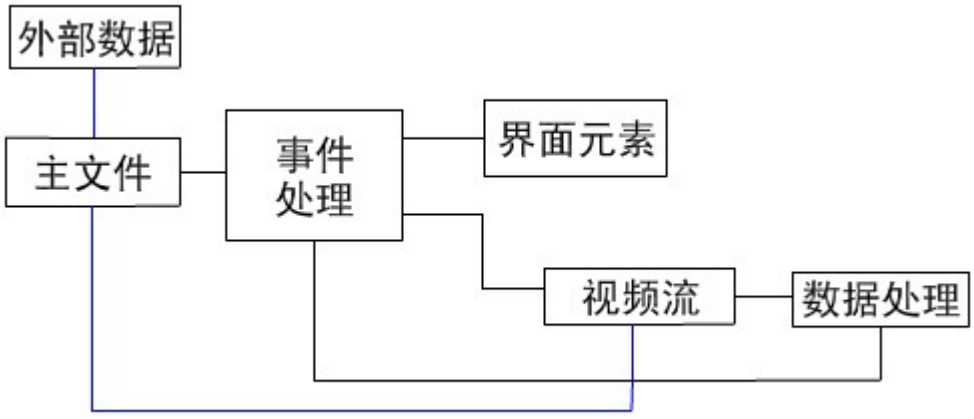


AS3 制作全功能的 FLV 网络视频播放器

作者：A 闪

文章来源：http://hi.baidu.com/%B0%B5%BA%DA%B2%E0%CE%C0/blog/item/2bd91d43e5b42e1a72f05d32.html

大家好！我们本节来讲解 AS3 制作 FLV 视频播放器的原理。在做播放器之前，我们先来设计一下程序的运行流程，并把这个流程制作成流程图。这样做起来，就显得方便快捷。我们可以尽量详细的描写我们程序的运行流程，不过要从宏观上去考虑。具体遇到的编程问题再慢慢去解决！



我们来看一下这个流程图（这个是偶自己取的名字）！首先是主文件接受外部的参数，包括视频地址，选项等等。然后在主文件中我们来定义一些事件，比如说按钮事件，有关视频的事件等等。透过这些事件我们又对视频流和界面元素进行了调用。最后是视频流中的一些数据进行数据处理，然后将这些数据在传递给主文件。好了！这样就完成了一个播放器的全部工作流程。

很多人都有一个不好的习惯，就是说到一个效果，拿来就做，没有一个计划。这样反而事倍功半。不仅写起来很费劲，而且返工的机率很大。所有我们在做一些 FLASH 交互式程序的时候一定要先构思，再动手去做。

好了！我们现在来仔细考虑，这个程序的一些细节。我们用顺序号来表示。

1、当网页被打开的时候，我们要 FLASH 接受几个参数。这些参数决定着播放器的一些选项。

2、网页被打开的时候，FLASH 要读取自身的宽高。这个要用来对视频的宽高进行限制。

3、播放器检测视频连接是否正常，检查跨域访问权限。

4、开始播放视频。

5、视频的各个调节功能（包括进度，声音，全屏，亮度等）。

6、视频播放结束后的工作。

7、断开所有数据流！

这样，我们的程序功能就清楚可见了！根据这个来制作程序效率是非常高的。

时间很仓促，为了完成这套教程，特意抽出了 3 天的业余时间来制作这个 FLV 播放器。其中难免有 BUG，如果大家发现了 BUG 欢迎留言给我。再不断的完善！

那么首先提一个问题，制作一个中型的 FLASH 交互式程序是先写文档类还是先写别的自定义类呢？我个人觉得应该先写自定义类，将我们所需要的功能都封装好后再写文档类。这样写起来比较方便。好了！废话不多说，我们本节先来写第一个类吧！

我们平常播放 FLV 视频要三个类，笔者感觉这样写比较麻烦。所以我的第一步就是从新封装 Video 类。让这个类和另外 2 个类封装到一起。好了这个封装的类放到了一个名为 playermedia 的文件夹内，代码如下：

```
/*A 闪工作室 编写
版权所有 本实例不可用于商业用途，违者必究
2009.9.11 作者：A 闪
*/
package playermedia{
import playermedia.CustomClient;
import flash.media.SoundTransform;
import flash.events.IOErrorEvent;
import flash.events.NetStatusEvent;
import flash.events.AsyncErrorEvent;
import flash.net.NetConnection;
import flash.net.NetStream;
import flash.media.Video;
public class Playervideo extends Video {
   private var netconnection:NetConnection;
   private var netstream:NetStream;
   private var video_url:String;
   private var videosound:SoundTransform;
   public static var video_time:Number=0;
   public function Playervideo():void {
    CustomClient.setdata(this);
    netconnection=new NetConnection ;
    netconnection.connect(null);
    netstream=new NetStream(netconnection);
    netstream.client=new CustomClient ;
    videosound = new SoundTransform();
    videosound.volume = 1;
    netstream.soundTransform = videosound;
    this.attachNetStream(netstream);
    netstream.checkPolicyFile = true;
    netconnection.addEventListener(NetStatusEvent.NET_STATUS,streamnotfound);
    netconnection.addEventListener(AsyncErrorEvent.ASYNC_ERROR,netmistake);
    netconnection.addEventListener(IOErrorEvent.IO_ERROR,mistake);
    netstream.addEventListener(IOErrorEvent.IO_ERROR,mistake);
    netstream.addEventListener(AsyncErrorEvent.ASYNC_ERROR,netmistake);
    netstream.addEventListener(NetStatusEvent.NET_STATUS,streamnotfound);

   }
   private function netmistake(evt:AsyncErrorEvent):void {
    trace("异步错误");
   }
```

```
private function streamnotfound(evt:NetStatusEvent):void {
 switch (evt.info.code) {
  case "NetStream.Play.Stop" :
   CustomClient.stopvideo();
   break;
  case "NetStream.Play.Start" :
   trace("视频开始播放");

   break;
  case "NetStream.Buffer.Empty" :
   trace("数据的接收速度不足以填充缓冲区。");
   break;
  case "NetStream.Buffer.Full" :
   trace("缓冲区已满并且流将开始播放。");
   CustomClient.secplay();
   break;
  case "NetStream.Buffer.Flush" :
   trace("数据已完成流式处理，剩余的缓冲区将被清空。");
   break;
  case "NetStream.Seek.Notify" :
   trace("搜寻操作完成。");

   break;
 }
}
private function mistake(evt:IOErrorEvent):void {
 //错误处理代码
 trace("错了!");
}
//播放视频
public function playvideo(url:String):void {
 netstream.play(url);
 video_url = url;
}
//加载条进度百分比
public function loadprogress():Number {
 return netstream.bytesLoaded / netstream.bytesTotal;
}
//视频的播放时间计算
public function playtime():String {
 var streamminute:Number = Math.floor(netstream.time / 60);
 var smin_str:String;
 if (streamminute>=10) {
  smin_str = String(streamminute);
 } else {
  smin_str = "0" + streamminute;
 }
 var streamsecond:Number = Math.round(netstream.time % 60);
 var ssec_str:String;
```

```
     if (streamsecond>=10) {
      ssec_str = String(streamsecond);
     } else {
      ssec_str = "0" + streamsecond;
     }
     var newtime:String = smin_str + ":" + ssec_str;
     return newtime;
    }
    //设置视频音量
    public function setsound(sound_data:Number):void {
     videosound.volume = sound_data;
     netstream.soundTransform = videosound;
    }
    //播放或停止视频
    public function playstopvideo():void {
     netstream.togglePause();
    }
    //计算当前播放时间的百分比
    public function playvideonowtime():Number {
     return netstream.time / video_time;
    }
    //跳转视频
    public function playtimevideo(whereplaytime:Number):void {
     netstream.seek(whereplaytime);
     netstream.play(video_url);
     netstream.togglePause();
    }
    public function playtimemyvideo(timevideo:Number):void{
     netstream.seek(timevideo);
    }
}
}
```

这就是我封装的类!其中用到另外一个回调函数的类!我们下节中将回调函数的类告诉大家!

上一节中我们提到一个回调函数的类。这节中我们就来编写这个类。这个类放到一个名为 playermedia 的文件夹内，代码如下：

```
/*A 闪工作室 编写
版权所有 本实例不可用于商业用途，违者必究
2009.9.11 作者：A 闪
*/
package playermedia{
import mathpool.Videorange;
import mathpool.Videotime;
import playermedia.Playervideo;
import Main;
import flash.media.Video;
public class CustomClient {
```

```
   public static var stage_width:Number=320;
   public static var stage_height:Number=280;
   private static var stage_video:Video;
   public function onMetaData(info:Object):void {
    Videotime.caltime(info.duration);
    Playervideo.video_time = info.duration;
    stage_video.width = info.width;
    stage_video.height = info.height;
    Main._videowidth = info.width;
    Main._videoheight = info.height;
    Main.videowhboolean = true;
    Main.timeforvideo = info.duration;
    Videorange.videosize(stage_video,info.width,info.height,stage_width,stage_height);
   }
   //设置 Video 对象
   public static function setdata(obj:Video):void {
            stage_video = obj;
   }
   //视频停止
   public static function stopvideo():void{
    Main.video_durationboolean = false;
    Main.playboolean = false;
   }
   //视频开始播放
   public static function secplay():void{
    Main.proboolean = false;
   }
}
}
```

好了！这个类中我们又引出了两个工具类！分别是用来计算视频宽高和视频时间的类！我们将在后面两节中编写这两个工具类！

上一节中提到工具类，本节我们来编写这个负责计算视频时间的工具类！这个类放到一个名为 mathpool 的文件夹内，代码如下：

```
/*A 闪工作室 编写
版权所有 本实例不可用于商业用途，违者必究
2009.9.11 作者：A 闪
*/
package mathpool{
import Main;
public class Videotime {
   public static function caltime(video_time:Number):void {
    var minute:Number=Math.floor(video_time / 60);
    var min_str:String;
    if (minute >= 10) {
     min_str=String(minute);
    } else {
```

```
     min_str="0" + minute;
    }
    var second:Number=Math.round(video_time % 60);
    var sec_str:String;
    if (second >= 10) {
     sec_str=String(second);
    } else {
     sec_str="0" + second;
    }
    Main.videotime=min_str + ":" + sec_str;
   }
}
}
```

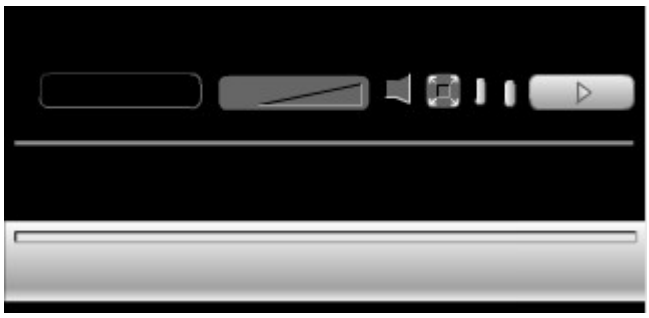
这个类是比较简单的！下节中我们将编写视频宽高的工具类！其中涉及到一个算法！大家要去自己琢磨了！

大家好！书接上文，我们说道这个类将涉及到一个算法！下面我们就来看看这个算法究竟是什么样子的！依然是放到 mathpool 文件夹下，代码如下：

```
/*A 闪工作室 编写
版权所有 本实例不可用于商业用途，违者必究
2009.9.11 作者：A 闪
*/
package mathpool{
import flash.media.Video;
public class Videorange {
   public static function
videosize(obj:Video,video_width:Number,video_height:Number,stage_width:Number,stage_he
ight:Number):void {
    if ((video_width/video_height)<(stage_width/(stage_height-40))) {
     obj.height = stage_height-40;
     obj.width = (stage_height-40)*video_width/video_height;
     obj.x = (320-stage_width)/2 + Math.abs((stage_width-obj.width)/2);
     obj.y = (280-stage_height)/2;
    } else if ((video_width/video_height)>(stage_width/(stage_height-40))) {
     obj.width = stage_width;
     obj.height = stage_width*video_height/video_width;
     obj.x = (320-stage_width)/2;
     obj.y = (280-stage_height)/2 + Math.abs((stage_height-obj.height-40)/2);
    } else if ((video_width/video_height)==(stage_width/(stage_height-40))) {
     obj.width = stage_width;
     obj.height = stage_height-40;
     obj.x = (320-stage_width)/2;
     obj.y = (280-stage_height)/2;
    }
   }
}
}
```

脚本不算多，但是其中的算法很重要！好了！关于视频类我们都写完了！后面我们要编写有关界面的类了！最多的还是涉及到位置的变换！

上一节中我们将视频类所需要的函数都写完了！本节开始我们来编写界面类！首先，我们要绘制我们所需要的一些界面元素，包括按钮等。我们就用截图的方式来向大家展示如何制作！



这就是笔者做的界面元素，分别为他们设置连接，具体连接的名称是什么大家可以在下面的脚本中查到！这个界面类是放到一个名为 playerinterface 文件夹内的，代码如下：

```
/*A 闪工作室 编写
版权所有 本实例不可用于商业用途，违者必究
2009.9.11 作者：A 闪
*/
package playerinterface{
//导入工具类
import faceplace.facepool;
//导入文本类
import flash.text.TextField;
import flash.text.TextFieldAutoSize;
//导入基类
import flash.display.Sprite;
public class Drawinterface extends Sprite {
   //创建对象
   private var progressbarBackground:ProgressbarBackground;
   public var progressbutton:ProgressButton;
   public var playandstopbutton:PlayAndStopButton;
   private var timeframe:TimeFrame;
   private var soundbackground:SoundBackground;
   public var soundmute:SoundMute;
   public var soundregulate:SoundRegulate;
   public var fullscreen:FullScreen;
   private var loadprogress:LoadProgress;
   private var timetext:TextField;
   private var loadprogress_width:Number;
   public function Drawinterface():void {
    //设置文本
    timetext = new TextField();
    timetext.width = 80;
    timetext.height = 17;
    timetext.autoSize = TextFieldAutoSize.CENTER;
    timetext.selectable = false;
```

```
     timetext.textColor = 0x000000;
     timetext.text = "00:00/00:00";
     //新建对象
     progressbarBackground = new ProgressbarBackground();
     progressbutton = new ProgressButton();
     playandstopbutton = new PlayAndStopButton();
     timeframe = new TimeFrame();
     soundbackground = new SoundBackground();
     soundmute = new SoundMute();
     soundregulate = new SoundRegulate();
     fullscreen = new FullScreen();
     loadprogress = new LoadProgress();
     //将对象添加到显示列表
     addChild(progressbarBackground);
     addChild(loadprogress);
     addChild(progressbutton);
     addChild(playandstopbutton);
     playandstopbutton.gotoAndStop(2);
     addChild(timeframe);
     timeframe.addChild(timetext);
     addChild(soundbackground);
     addChild(soundmute);
     soundmute.stop();
     soundbackground.addChild(soundregulate);
     addChild(fullscreen);
     fullscreen.stop();
    }
    public function
moveinterface(rootwidth:Number=320,rootheight:Number=280,setsound:Number=71):void{
     //对显示元素进行重新的位置排列
     facepool.getdata(progressbarBackground,rootwidth,rootheight,setsound);
     loadprogress_width = facepool.getdata(loadprogress,rootwidth,rootheight,setsound);
     facepool.getdata(progressbutton,rootwidth,rootheight,setsound);
     facepool.getdata(playandstopbutton,rootwidth,rootheight,setsound);
     facepool.getdata(timeframe,rootwidth,rootheight,setsound);
     facepool.getdata(soundbackground,rootwidth,rootheight,setsound);
     facepool.getdata(soundmute,rootwidth,rootheight,setsound);
     facepool.getdata(soundregulate,rootwidth,rootheight,setsound);
     facepool.getdata(fullscreen,rootwidth,rootheight,setsound);
    }
    //设置文本显示内容
    public function showtime(now:String):void{
     timetext.text = now;
    }
    //进度条长度
    public function loadprogresswidth(percentage:Number):void{
     loadprogress.width = loadprogress_width*percentage;
    }
```

```
}
}
```

好了！这里我们又引出一个工具类！用来计算界面元素位置的！我们下节中编写该类。

这节我们来编写工具类！这个类放在名称为 faceplace 的文件夹内，代码如下：

```
/*A 闪工作室 编写
版权所有 本实例不可用于商业用途，违者必究
2009.9.11 作者：A 闪
*/
package faceplace{
import flash.utils.getQualifiedClassName;
public class facepool {
   public static function
getdata(obj:*,thewidth:Number,theheight:Number,soundX:Number=71):* {
    switch (getQualifiedClassName(obj)) {
     case "ProgressbarBackground" :
      obj.x = (thewidth - 320) / 2 * -1;
      obj.y = (theheight-240)/2+220;
      obj.width = thewidth;
      break;
     case "LoadProgress" :
      obj.x = ((thewidth-320)/2-6)*-1;
      obj.y = (theheight-240)/2+225.5;
      obj.width = thewidth-12;
      return obj.width;
      break;
     case "ProgressButton":
        obj.x = (thewidth-320)/2*-1+5;
      obj.y = (theheight-240)/2+230;
      break;
     case "PlayAndStopButton":
        obj.x = (thewidth - 320) / 2 * -1+5;
      obj.y = (theheight-240)/2+239;
      break;
     case "TimeFrame":
        obj.x = (thewidth - 320) / 2 * -1+65;
      obj.y = (theheight-240)/2+239;
      break;
     case "SoundBackground":
        obj.x = (thewidth - 320) / 2+190;
      obj.y = (theheight-240)/2+239;
      break;
     case "SoundMute":
        obj.x = (thewidth - 320) / 2+193;
      obj.y = (theheight-240)/2+240;
      break;
     case "SoundRegulate":
```

```
        obj.x = soundX;
       obj.y = 16;
       break;
      case "FullScreen":
        obj.x = (thewidth - 320) / 2+285;
       obj.y = (theheight-240)/2+239;
       break;
     }
    }
}
}
```

好了！我们所有的自定义类都写完了！后面我们就来编写文档类！

终于到最后了！激动人心的时刻到来了！我们已经准备好了我们所需要的类，现在要做的就是写一个文档类，将这些零散的类综合起来！好了！看文档类的代码如下：

```
/*A 闪工作室 编写
版权所有 本实例不可用于商业用途，违者必究
2009.9.11 作者：A 闪
*/
package {
//导入矩阵类
import flash.geom.Rectangle;
//导入视频参数类
import playermedia.CustomClient;
//导入视频大小调整类
import mathpool.Videorange;
//导入按钮事件
import flash.events.MouseEvent;
//导入视频类
import playermedia.Playervideo;
//导入界面类
import playerinterface.Drawinterface;
//导入 JS 交互类
import flash.external.ExternalInterface;
//导入计时类
import flash.utils.setInterval;
import flash.utils.clearInterval;
//导入事件类
import flash.events.Event;
//导入舞台类
import flash.display.Stage;
//导入舞台缩放类型
import flash.display.StageScaleMode;
//导入基类
import flash.display.Sprite;
public class Main extends Sprite {
   //总体时间长度
```

```
public static var videotime:String = "00:00";
//界面对象
private var playerface:Drawinterface;
//视频的宽高
public static var _videowidth:Number;
public static var _videoheight:Number;
//计时器的次数
private var counter:uint = 0;
//计时对象
private var intervalId:uint;
//播放器的宽
private var stagewidth:Number = 320;
//视频地址
private var flvvideourl:String;
//视频宽高改变开关
public static var videowhboolean:Boolean = false;
//视频对象
private var flvvideo:Playervideo;
//视频的总时间长
public static var timeforvideo:Number;
//视频载入进度判断开关
private var loadvideoboolean:Boolean = false;
//舞台宽高改变开关
private var rootsizeboolean:Boolean = false;
//静音按钮开关
private var soundboolean:Boolean = false;
//音量调节按钮的 X 轴
private var soundbuttonX:Number = 71;
//视频播放的状态记录
public static var playboolean:Boolean = true;
//视频播放完毕判断
public static var video_durationboolean:Boolean = true;
//视频进度滑块的拖动范围
private var buttonre:Rectangle;
//时间拖动开关
public static var proboolean:Boolean = true;
//**************************************//
//构造器函数
public function Main():void {
 //设置播放器的缩放类型为无缩放
 stage.scaleMode = StageScaleMode.NO_SCALE;
 //设置播放器的宽高响应
 stage.addEventListener(Event.RESIZE,rangerevision);
 //对播放器的下载进度进行监视
 this.loaderInfo.addEventListener(Event.COMPLETE,loadthis);
}
//**************************************//
//播放器的宽高发生改变是进行响应的函数
private function rangerevision(evt:Event):void {
```

```
    //将舞台宽赋予一个变量
    stagewidth = stage.stageWidth;
    //设置 CustomClient 类中的静态变量，将他们的值赋予舞台坐标
    CustomClient.stage_width = stage.stageWidth;
    CustomClient.stage_height = stage.stageHeight;
    //界面元素进行位置调整
    playerface.moveinterface(stage.stageWidth,stage.stageHeight,soundbuttonX);
    //视频尺寸改变
    if (videowhboolean) {
     //调用视频类的一个函数，从而设置视频的宽高
     Videorange.videosize(flvvideo,_videowidth,_videoheight,stage.stageWidth,stage.sta
geHeight);
    }
    //定义舞台上进度按钮的拖动范围
    buttonre = new
Rectangle(playerface.progressbutton.x,playerface.progressbutton.y,(stagewidth-10),0);
   }
   //**************************************//
   //载入完成后执行
   private function loadthis(evt:Event):void {
    //运行初始化函数
    init();
    //删除播放器下载监视函数
    this.loaderInfo.removeEventListener(Event.COMPLETE,loadthis);
   }
   //**************************************//
   //初始化
   private function init():void {
    //生成视频
    loadvideo();
    //生成界面
    playerface = new Drawinterface();
    //将界面元素添加到显示列表
    addChild(playerface);
    //界面元素位置进行设置
    playerface.moveinterface();
    //对界面元素定义事件
    playerface.soundregulate.addEventListener(MouseEvent.MOUSE_DOWN,sounddown);
    playerface.soundregulate.addEventListener(MouseEvent.MOUSE_UP,soundup);
    addEventListener(MouseEvent.MOUSE_UP,soundup);
    stage.addEventListener(MouseEvent.MOUSE_UP,soundup);
    flvvideo.addEventListener(MouseEvent.MOUSE_UP,soundup);
    playerface.soundmute.addEventListener(MouseEvent.CLICK,soundup);
    playerface.fullscreen.addEventListener(MouseEvent.CLICK,fullroot);
    playerface.playandstopbutton.addEventListener(MouseEvent.CLICK,playvideo);
    playerface.progressbutton.addEventListener(MouseEvent.MOUSE_DOWN,proset);
            //重新定义进度滑块的拖动范围
    buttonre = new
Rectangle(playerface.progressbutton.x,playerface.progressbutton.y,310,0);
```

```
 //JS 调用
 readjavascript();
 //定义一个逐帧函数
 addEventListener(Event.ENTER_FRAME,work);
}
//**************************************//
//生成视频对象
private function loadvideo():void {
 //视频传递参数以及生成视频对象
 flvvideo = new Playervideo();
 //将视频添加到显示列表
 addChildAt(flvvideo,0);
 //定义变量的值
 loadvideoboolean = true;
}
//**************************************//
//调用外部 JS 函数
private function readjavascript():void {
 //判断该网页中的 FLASH 是否允许接受 JS 参数
 if (ExternalInterface.available) {
  //调试语句
  try {
   //对外部的 JS 开放一个接口
   ExternalInterface.addCallback("setvideodata",acceptjsurl);
   //当发生错误时进行响应
  } catch (e:SecurityError) {
   //跨域访问的权限错误
  } catch (e:SecurityError) {
   //无法回调
  } catch (e:Error) {
   //数据连接失败
  }
  //调用外部 JS 函数
  ExternalInterface.call("sendToActionScript");
 }
}
//响应 JS 的函数实体
/*videourl 控制视频路径，playboolean 控制是否开始播放*/
private function acceptjsurl(videourl:String = null):void {
 //讲传递过来的参数进行赋值
 flvvideourl = videourl;
 flvvideo.playvideo(flvvideourl);
 //判断时候接受到了视频参数
 if (videourl==null&&counter==0) {
  //调用调试函数
  calculagraphjs();
 }
}
//**************************************//
```

```
//计时，查看 JS 参数是否接收到
private function calculagraphjs():void {
 //定义一个时间函数
 intervalId = setInterval(detectvideourl,3000);
}
//对 JS 函数进行调用以获取视频的路径，时间函数的响应函数
private function detectvideourl():void {
 //查看是否接受到路径，
 if (flvvideourl==null) {
  //如果没有接收到参数则调用 JS 的函数，再次获取参数
  counter++;
  ExternalInterface.call("sendToActionScript");
  //如果获取函数超过 3 次，则不再尝试！直接报错！
  if (counter==4) {
   clearInterval(intervalId);
  }
 } else {
  //当获取到了参数时，清除这个时间函数
  clearInterval(intervalId);
 }
}
//*****************************************//
//逐帧函数
private function work(evt:Event):void {
 var video_time:String;
 var videopro:Number;
 if (loadvideoboolean) {
  //显示视频的当前播放时间和总体时间
  video_time = flvvideo.playtime();
  playerface.showtime(video_time+"/"+videotime);
  //设置进度条长度
  playerface.loadprogresswidth(flvvideo.loadprogress());
 }
 videopro = flvvideo.playvideonowtime();
 if (proboolean==false) {
  //设置进度按钮的位置
  playerface.progressbutton.x = (stagewidth-10)*videopro + (320-stagewidth)/2+5;
 }
 if (video_durationboolean==false) {
  //让播放按钮跳转到第一帧
  playerface.playandstopbutton.gotoAndStop(1);
 }
}
//*****************************************//
//声音按钮按下响应
private function sounddown(evt:MouseEvent):void {
 //声音按钮拖动
 var soundre:Rectangle = new Rectangle(22,16,49,0);
 evt.target.startDrag(false,soundre);
```

```
    }
    //*****************************************//
    //鼠标抬起响应函数，对声音调节和进度调节有效
    private function soundup(evt:MouseEvent):void {
     //停止拖动
     playerface.soundregulate.stopDrag();
     playerface.progressbutton.stopDrag();
     //调节视频的进度，通过进度按钮
     if (stagewidth>=320) {
      flvvideo.playtimemyvideo(Math.round((Math.abs((stagewidth-320)/2)+playerface.prog
ressbutton.x)/(stagewidth-10)*timeforvideo));
     } else {
      flvvideo.playtimemyvideo(Math.round((playerface.progressbutton.x-(320-stagewidth)
/2)/(stagewidth-10)*timeforvideo));
     }
     //对静音按钮和音量调节按钮作出响应
     if (evt.target.name=="instance14"&&soundboolean == false) {
      playerface.soundregulate.x = 22;
      flvvideo.setsound(0);
      playerface.soundmute.nextFrame();
      soundboolean = !soundboolean;
     } else if (evt.target.name=="instance14"&&soundboolean == true) {
      playerface.soundregulate.x = soundbuttonX;
      flvvideo.setsound((playerface.soundregulate.x-22)/49);
      soundboolean = !soundboolean;
      playerface.soundmute.prevFrame();
     } else {
      flvvideo.setsound((playerface.soundregulate.x-22)/49);
      soundbuttonX = playerface.soundregulate.x;
     }
    }
    //*****************************************//
    //全屏函数
    private function fullroot(evt:MouseEvent):void {
     switch (stage.displayState) {
      case "normal" :
       stage.displayState = "fullScreen";
       evt.target.nextFrame();
       break;
      case "fullScreen" :
      default :
       stage.displayState = "normal";
       evt.target.prevFrame();
       break;
     }
    }
    //*****************************************//
    //对播放按钮的点击事件进行响应的函数
    private function playvideo(evt:MouseEvent):void {
```

```
    if (playboolean==true) {
     evt.target.prevFrame();
    } else {
     evt.target.nextFrame();
     if (video_durationboolean==false) {
      flvvideo.playtimevideo(0);
      video_durationboolean = !video_durationboolean;
     }
    }
    flvvideo.playstopvideo();
    playboolean = !playboolean;
   }
   //**************************************//
   //进度调节按钮的拖动事件响应函数
   private function proset(evt:MouseEvent):void {
    proboolean = true;
    evt.target.startDrag(false,buttonre);
   }
}
}
```

在 FLA 文件中将文档类改为 Main，测试发布影片！~咦！怎么没有效果呢？别着急！我们这个 FLV 播放器是通过 JS 传递视频路径的，所以我们还需要一个 HTML 文件！下一节中我们再说！

前面我们已经完成了这个 FLV 播放器的代码编写阶段。接下来就是测试阶段了！我们写一个 HTML 文档！代码如下：

```
<!-- saved from url=(0014)about:internet -->
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>FLASH 与 JS 交互</title>
<script language="JavaScript">
     function sendToActionScript(value) {
         window.ExternalInterfaceExample.setvideodata("aa.flv");
     }
</script>
</head>
<body>
     <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
             id="ExternalInterfaceExample" width="500" height="500"
             >
         <param name="movie" value="flvplayerbeta1.swf" />
         <param name="allowFullScreen" value="true" />
     </object>
</body>
</html>
```

为了测试方便，我们可以自己找一段网络上的视频地址写到这个 HTML 文档中，也可以在同目录文件夹下放置一个 FLV 视频！运行网页，我们就可以看到效果了！这里笔者将播放器的运行画面截图。

这是播放器运行时候的画面！

结束语

我们的 FLV 播放器做完了!不知道大家有什么感想？其实做这样的一个播放器并非非常困难！只要我们用心去做。就一定能做好！

我这个播放器并不是非常的完美，其中可能还含有 BUG！只是想起到一个抛砖引玉的作用！并不是让大家全班照抄来制作这样的播放器！

由于时间比较仓促，感觉这个程序的耦合度还是比较高。以后我会继续完善，陆续发出更新版本的！希望大家留意我的博客！我会在第一时间将最新版本的源码奉献给大家！

最后让大家看一下这个工程文件夹（主要是了解一下文件的结构）

很多人觉得这样将脚本复制一边有时候会出现错误！确实，为了方便大家学习，特意将这个源代码放到网上供大家下载!

下载地址：http://hi.baidu.com/%B0%B5%BA%DA%B2%E0%CE%C0/blog/item/a9dc3a032affa17d3912bbae.html

Flash(FLV)视频播放器开源代码大集合

作者：A 客网

文章来源：http://www.51as.com/as3/Flash-FLV-shipinbofangqikaiyuandaimadajihe_238.html

下面的页面里收集了目前所有的 Flash(FLV)视频播放器和开源代码

http://www.flashstreamworks.com/tutorials/flvplayerlinklist.php

Flv Player List

Proxus FLV player

http://www.proxus.com/components/index.php

Check Flashstreamworks' review

FLV player by Martijn de Visser

http://www.martijndevisser.com/

FLV player by Peldi

http://www.peldi.com/blog/000103.html

FLV player by VideoSpark

http://www.videospark.com/prog_flv8/player.php

FLV player by Wimpy

http://www.gieson.com/Library/projects/wimpyAV.html

FLV player component by Meshybeats

http://meshybeats.com/components/

http://www.flashstreamworks.com/tutorials/flvplayersource.php

Flash Video Player sources

These are simple source examples for Flash Video applications. Feel free to modify and edit them.

Playing video without components

Download

Playing video with components

Download

Using FVSS for Flash Video with HTML authoring

Download

Live streaming without components

Download

Multiuser chat room with components

Download

[FLVParser].

用法:

命令行>flvparse 你的视频.flv cue.txt -s

它另一个功能是根据 cue-point 文件分割 FLV 文件.

点击下载

flash九宫对对碰 V1

作者：小 S

文章来源：http://www.xiaos8.com/article.asp?id=44

欢迎多多提程序方面的意见，先公布源代码，欢迎高手指导

点击播放/隐藏媒体

http://www.xiaos8.com/uploads/pro/lookAgain.swf

查看最新版对对碰，请点下面这个链接

http://www.kongregate.com/games/sunbright/bumping-friuts

首先是fla主文件

```
import index.item.lookAgain.Vessel;

var scene:Vessel = new Vessel;
scene.x = scene.y = 30;
addChild(scene);

var txt:TextField = new TextField;
txt.htmlText = "九宫对对碰 V1 版\n 作者：sunbright\n 玩法：将相邻的两个方块相互交换，如果促成一条线都是同样形状，则消除成功（45°斜线亦可）\n\n 胡乱瞎搞的练手之作，因为本身美工不好，所以在美工方面没下功夫，请见谅。\n\n 源代码已在<a href='http://www.xiaos8.com'target='_blank'><font color='#0000ff'><u>我的博客</u></font></a>公开\n\n 谢谢观看，欢迎来我博客交流技术问题\n\n\n 欢迎光临小 S 吧\n\n<a href='http://www.xiaos8.com' target='_blank'><font color='#0000ff'><u>sunbright 博客\n 点击进入</u></font></a>\n\n<a href='http://www.xiaos8.com/article.asp?id=44' target='_target'><font color='#0000ff'><u>直接进入源代码帖子浏览</u></font></a>";
txt.setTextFormat(new TextFormat("Arial"));
txt.multiline = txt.wordWrap = true;
txt.selectable = false;
txt.width = 120;
txt.height = 400;
txt.x = 420;
txt.y = 10;

addChild(txt);
```

然后是 index.item.lookAgain.Vessel.as 类文件：

```
package index.item.lookAgain{

  import flash.display.Sprite;
  import flash.events.TextEvent;
  import flash.events.MouseEvent;
  import flash.text.TextField;
  import flash.utils.setTimeout;

  import index.base.program.TwoArray;
```

```
  import index.item.lookAgain.Thing;
  import index.item.lookAgain.Painting;

  public class Vessel extends Sprite{

    private const listAr:Array = new Array("circle","square","rectangle","triangle","ellipse","he
xagon","trapezoid","parallelogram","rhombus","star");
    private const z:TwoArray = new TwoArray(10,10);
    private var thingAr:Array = new Array;
    private var tmpAr:Array;

    private const _scoreText:TextField = new TextField;
    private var _score:uint;

    private const _timesText:TextField = new TextField;
    private var _times:uint;

    public function Vessel(){
      mouseEnabled = false;

      //设置默认分数，重新排列次数
      score = 0;
      times = 0;

      //设置分数，重新排列次数的文本框
      _scoreText.mouseEnabled =
      _scoreText.selectable =
      _timesText.selectable = false;
      _scoreText.autoSize = "left"
      _timesText.autoSize = "right";
      _scoreText.y =
      _timesText.y = 380;
      _timesText.x = 200;

      addChild(_scoreText);
      addChild(_timesText);

      _timesText.addEventListener(TextEvent.LINK,reload);

      z.traversal(function(ar:int,ix:int,iy:int):void
      {
        createThing(ix,iy);
      });

      addEventListener(MouseEvent.MOUSE_OVER,funOver);
      addEventListener(MouseEvent.MOUSE_OUT,funOut);
      addEventListener(MouseEvent.CLICK,funClick);

      if(expSuccess()){
```

```
      setTimeout(clearThing,200);
    }
  }

  //重新排列
  private function reload(e:TextEvent):void{
    if(_times > 0){
      times --;
      z.traversal(function(ar:int,ix:int,iy:int):void
      {
        removeChild(getChildByName(ix + "_" + iy));
        createThing(ix,iy);
      });

      if(expSuccess()){
        setTimeout(clearThing,200);
      }
    }
  }

  //鼠标滑过事件
  private function funOver(e:MouseEvent):void{
    if(e.target is Thing) e.target.toBig();
  }

  //鼠标离开事件
  private function funOut(e:MouseEvent):void{
    if(e.target is Thing) e.target.toSmall();
  }

  //鼠标单击事件
  private function funClick(e:MouseEvent):void{
    if(e.target is Thing){
      if(e.target.isTo){
        e.target.isTo = false;
        thingAr.push(e.target as Thing);
        if(thingAr.length == 2){
          exchange(thingAr[0],thingAr[1]);
        }
      }
    }
  }

  //两个东西调换
  private function exchange(t1:Thing,t2:Thing):void{
    if(Math.abs(t1.ix - t2.ix) < 2 && Math.abs(t1.iy - t2.iy) < 2){
      exchangeData(t1,t2);

      if(expSuccess()){
```

```
      setTimeout(clearThing,200);
    }else{
      setTimeout(exchangeData,200,t2,t1);
    }
  }

  t1.isTo = true;
  t2.isTo = true;
  thingAr = new Array;
}

//数据交换
private function exchangeData(t1:Thing,t2:Thing):void{
  //交换 ix，iy 值
  var tmpx:uint = t1.ix;
  var tmpy:uint = t1.iy;
  t1.ix = t2.ix;
  t1.iy = t2.iy;
  t2.ix = tmpx;
  t2.iy = tmpy;

  //交换数字标识
  var tmpNum:int = z[t1.ix][t1.iy];
  z[t1.ix][t1.iy] = z[t2.ix][t2.iy];
  z[t2.ix][t2.iy] = tmpNum;

  //交换实例名
  var tmpStr:String = t1.name;
  t1.name = t2.name;
  t2.name = tmpStr;
}

//检测是否有成功项
private function expSuccess():Boolean{
  tmpAr = new Array;
  z.traversal(function(ar:int,ix:int,iy:int):void
  {
    expPoint(ix,iy);
  });
  return Boolean(tmpAr.length);
}

//检测相对某点的成功项
private function expPoint(ix:int,iy:int):void{
  expLine(ix,iy,ix+1,iy,ix+2,iy);
  expLine(ix,iy,ix+1,iy,ix-1,iy);
  expLine(ix,iy,ix-1,iy,ix-2,iy);
  expLine(ix,iy,ix,iy+1,ix,iy+2);
  expLine(ix,iy,ix,iy+1,ix,iy-1);
```

```
      expLine(ix,iy,ix,iy-1,ix,iy-2);

      expLine(ix,iy,ix+1,iy+1,ix+2,iy+2);
      expLine(ix,iy,ix+1,iy+1,ix-1,iy-1);
      expLine(ix,iy,ix-1,iy-1,ix-2,iy-2);
      expLine(ix,iy,ix+1,iy-1,ix+2,iy-2);
      expLine(ix,iy,ix+1,iy-1,ix-1,iy+1);
      expLine(ix,iy,ix-1,iy+1,ix-2,iy+2);
    }

    //是否 3 点一样
    private function expLine(ix1:int,iy1:int,ix2:int,iy2:int,ix3:int,iy3:int):void{
      if(ix1 < 0 || ix2 < 0 || ix3 < 0 || iy1 < 0 || iy2 < 0 || iy3 < 0 || ix1 > 9 || ix2 > 9 || ix3 >
 9 || iy1 > 9 || iy2 > 9 || iy3 > 9 || z[ix1][iy1] == undefined || z[ix2][iy2] == undefined || z[ix3]
[iy3] == undefined){
        return;
      }
      if(z[ix1][iy1] == z[ix2][iy2] && z[ix3][iy3] == z[ix2][iy2]){
        tmpAr.push(getChildByName(ix1 + "_" + iy1));
        tmpAr.push(getChildByName(ix2 + "_" + iy2));
        tmpAr.push(getChildByName(ix3 + "_" + iy3));
      }
    }

    //消除方块
    private function clearThing():void{
      //去掉同样的项目
      tmpAr = tmpAr.sort().filter(function(item:Thing, index:int, array:Array):Boolean
      {
        while(index > 0){
          if(item == array[-- index]){
            return false;
          }
        }
        return true;
      });

      //消除方块
      for(var i:int = 0; i < tmpAr.length; i ++){
        if(tmpAr[i] != null){
          if(contains(tmpAr[i])){
            removeChild(tmpAr[i]);
            score ++;

            setTimeout(createThing,200,tmpAr[i].ix,tmpAr[i].iy);
          }
        }
      }
```

```
      //再次检查是否有成功项
      if(expSuccess()){
        setTimeout(clearThing,200);
      }
    }

    //生成一个方块
    private function createThing(ix:int,iy:int):void{
      z[ix][iy] = Math.floor(Math.random() * 10);

      var tmpSpr:Thing = Painting.draw(listAr[z[ix][iy]]);
      tmpSpr.name = ix + "_" + iy;
      tmpSpr.ix = ix;
      tmpSpr.iy = iy;
      addChild(tmpSpr);
    }

    //设置分数
    private function set score(num:int):void{
      _score = num;
      _scoreText.text = "一共得 " + _score * 100 + " 分";

      //判断是否过了 100 次方块消除，如果是增加一次重新排列使用机会
      if(_score / 100 == Math.floor(_score / 100)){
        times ++;
      }
    }

    //获取真实分数
    private function get score():int{
      return _score;
    }

    //设置重新排列次数
    private function set times(num:int):void{
      _times = num;
      _timesText.htmlText = "您还有 " + num + " 次机会使用 <a href='event:'><u>重新排列</u></a>";
    }

    //获取重新排列次数
    private function get times():int{
      return _times;
    }
  }
}
```

再是 index.item.lookAgain.Thing.as 文件

```
package index.item.lookAgain{
```

```
import flash.display.Sprite;

public class Thing extends Sprite{

  private var _isTo:Boolean = true;
  private var _ix:uint;
  private var _iy:uint;
  private var pro:uint;

  public function Thing(num:uint = 40){
    pro = num;
    mouseChildren = false;
  }

  //设置是否响应变大，变小
  public function set isTo(p:Boolean):void{
    _isTo = true;
    if(p){
      toSmall();
    }else{
      toBig();
    }
    _isTo = p;
  }

  //设置是否响应变大，变小
  public function get isTo():Boolean{
    return _isTo;
  }

  //设置 x 系数
  public function set ix(num:uint):void{
    x = num * pro;
    _ix = num;
  }

  //获取 x 系数
  public function get ix():uint{
    return _ix;
  }

  //设置 y 系数
  public function set iy(num:uint):void{
    y = num * pro;
    _iy = num;
  }

  //获取 y 系数
  public function get iy():uint{
```

```
      return _iy;
    }

    //变大
    public function toBig():void{
      if(_isTo){
        getChildAt(0).scaleX =
        getChildAt(0).scaleY = 0.8;
        graphics.clear();
        graphics.lineStyle(2,0);
        graphics.beginFill(0xffdddd,1);
        graphics.drawRect(-20,-20,40,40);
      }
    }

    //变小
    public function toSmall():void{
      if(_isTo){
        getChildAt(0).scaleX =
        getChildAt(0).scaleY = 0.6;
        graphics.clear();
        graphics.lineStyle(1,0x336699);
        graphics.beginFill(0xffffff,0);
        graphics.drawRect(-20,-20,40,40);
      }
    }

    //改写 toString
    override public function toString():String{
      return name;
    }
  }

}
```

接下来是 index.item.lookAgain.Painting.as 文件：
这个类中讲了一个设计模式的定义，该设计模式定义，是我自己总结的一些编程经验，归纳写下来的，希望在设计模式上有一定造诣的高手，多给指点！

```
/**
 * 设计模式：外卖模式
 * 定义：提交类型，返回实例，并通过固定流程，最终返回真正实用的实例。
 * 比如：外卖餐厅，打电话告诉前台服务员，需要一碗鱼香肉丝饭，一碗大排饭，提交类型：鱼香肉丝饭，大排饭；最后送到我家两份盖浇饭。在盖浇饭的生成过程中，除了鱼香肉丝和大排不同，其它所用的饭，饭盒，配菜等等，都是一样的。
 */
package index.item.lookAgain{

  import flash.display.Shape;
  import flash.filters.DropShadowFilter;
  import flash.filters.BlurFilter;
```

```
import index.item.lookAgain.Thing;

public class Painting{

  //创建东西的方法
  public static function draw(type:String):Thing{
    var reSha:Shape = Painting[type]();
    reSha.scaleX = reSha.scaleY = 0.6;
    reSha.filters = new Array(new DropShadowFilter(2),new BlurFilter(2,2,1));

    var reThi:Thing = new Thing;
    reThi.graphics.lineStyle(1,0x336699);
    reThi.graphics.beginFill(0xffffff,0);
    reThi.graphics.drawRect(-20,-20,40,40);

    reThi.addChild(reSha);
    return reThi;
  }

  //创建圆形
  private static function circle():Shape{
    var reSha:Shape = new Shape;
    reSha.graphics.beginFill(0x336699);
    reSha.graphics.drawCircle(0,0,20);
    return reSha;
  }

  //创建正方形
  private static function square():Shape{
    var reSha:Shape = new Shape;
    reSha.graphics.beginFill(0x00ff00);
    reSha.graphics.drawRect(-20,-20,40,40);
    return reSha;
  }

  //创建矩形
  private static function rectangle():Shape{
    var reSha:Shape = new Shape;
    reSha.graphics.beginFill(0x0000ff);
    reSha.graphics.drawRect(-20,-10,40,20);
    return reSha;
  }

  //创建三角形
  private static function triangle():Shape{
    var reSha:Shape = new Shape;
    reSha.graphics.beginFill(0x00ffff);
    reSha.graphics.moveTo(0,-20);
```

```
    reSha.graphics.lineTo(20,20);
    reSha.graphics.lineTo(-20,20);
    reSha.graphics.endFill();
    return reSha;
  }

  //创建椭圆
  private static function ellipse():Shape{
    var reSha:Shape = new Shape;
    reSha.graphics.beginFill(0xffff00);
    reSha.graphics.drawEllipse(-20,-10,40,20);
    return reSha;
  }

  //创建六边形
  private static function hexagon():Shape{
    var reSha:Shape = new Shape;
    reSha.graphics.beginFill(0xff00ff);
    reSha.graphics.moveTo(-20,0);
    reSha.graphics.lineTo(-10,-20);
    reSha.graphics.lineTo(10,-20);
    reSha.graphics.lineTo(20,0);
    reSha.graphics.lineTo(10,20);
    reSha.graphics.lineTo(-10,20);
    reSha.graphics.endFill();
    return reSha;
  }

  //创建梯形
  private static function trapezoid():Shape{
    var reSha:Shape = new Shape;
    reSha.graphics.beginFill(0x999999);
    reSha.graphics.moveTo(-10,-20);
    reSha.graphics.lineTo(10,-20);
    reSha.graphics.lineTo(20,20);
    reSha.graphics.lineTo(-20,20);
    reSha.graphics.endFill();
    return reSha;
  }

  //创建平行四边形
  private static function parallelogram():Shape{
    var reSha:Shape = new Shape;
    reSha.graphics.beginFill(0x99ff99);
    reSha.graphics.moveTo(-10,-20);
    reSha.graphics.lineTo(20,-20);
    reSha.graphics.lineTo(10,20);
    reSha.graphics.lineTo(-20,20);
    reSha.graphics.endFill();
```

```
      return reSha;
    }

    //创建菱形
    private static function rhombus():Shape{
      var reSha:Shape = new Shape;
      reSha.graphics.beginFill(0x77ff33);
      reSha.graphics.moveTo(0,-20);
      reSha.graphics.lineTo(-15,0);
      reSha.graphics.lineTo(0,20);
      reSha.graphics.lineTo(15,0);
      reSha.graphics.endFill();
      return reSha;
    }

    //创建五角星
    private static function star():Shape{
      var reSha:Shape = new Shape;
      reSha.graphics.beginFill(0x770033);
      reSha.graphics.moveTo(0,-20);
      reSha.graphics.lineTo(15,20);
      reSha.graphics.lineTo(-20,-5);
      reSha.graphics.lineTo(20,-5);
      reSha.graphics.lineTo(-15,20);
      reSha.graphics.endFill();
      return reSha;
    }
  }
}
```

最后一个导入使用过的二维数组类，这个类只是随便写了一下，方便大家了解上面，依然把类贴出来

```
/**
 *二维数组类
 *第一个参数为 x，第二个参数为 y
 *使用的时候，可以完全按照数组使用
 */
package index.base.program{

  public dynamic final class TwoArray extends Array{

    private var itemX:uint;
    private var itemY:uint;

    public function TwoArray(_itemX:uint,_itemY:uint) {
      itemX = _itemX;
      itemY = _itemY;
      for (var i:int=0; i<itemX; i++) {
        var tmpY:Array=new Array(itemY);
```

```
        push(tmpY);
      }
    }

    //遍历所有组员
    public function traversal(fun:Function):void{
      forEach(function(element:*,itemX:int,arr:Array)
      {
        element.forEach(function(elements:*,itemY:int,arrs:Array)
        {
          fun(elements,itemX,itemY);
        });
      });
    }
  }
}
```

Flash 涂鸦回放功能的实现

作者：A 客网

文章来源：http://www.51as.com/as3/Flashtuyahuifanggongnendeshixian_154.html

网上虽然有不少涂鸦，但是好像关于一些回放过程的代码.....
实现的原理有很多，最简单的就是记录用户的操作，下面我的这个画线回放，就是记录用户鼠标移动过程。
存放进一个数组里，用","和";"来区分是 moveto 还是 lineto。这样就可以实现简单的回放了
当然，如果要实现复杂的，例如填充色等等，就需要更多的记录了...在这里只做个开始，呵呵
(可以继续添加许多功能的，例如画笔的不同等等)

```
/**

* @(#)DrawBoard.as

*

* @author soda.C E-mail:sujun10@21cn.com

* @version 1.0

* <br>Copyright (C), 2007 soda.C

* <br>This program is protected by copyright laws.

* <br>Program Name:Soda.C.Draw

* <br>Date:2008-3-18

*/

package org.sujun.drawboard

{

import flash.display.Sprite;



import fl.controls.Button;



import flash.events.MouseEvent;

import flash.events.Event;

public class DrawBoard extends Sprite

{
```

```
private var playBtn :Button; //回放按钮

private var drawSprite :Sprite; //画板 mc

private var count :int; //回放的进度

private var pointAry :Array; //存放用户操作数据



public function DrawBoard()

{

playBtn = new Button();

playBtn.label = "回放";

this.addChild(playBtn);

playBtn.addEventListener(MouseEvent.MOUSE_DOWN, playDraw);

stage.addEventListener(MouseEvent.MOUSE_DOWN, mouseDownEvent);

stage.addEventListener(MouseEvent.MOUSE_UP, mouseupEvent);



drawSprite = new Sprite();

this.addChild(drawSprite);

//画线风格

drawSprite.graphics.lineStyle(1,0x000000);



pointAry = new Array();

count = 0;





}

//回放事件

private function playDraw(event:MouseEvent):void
```

```
{

event.stopPropagation();

//准备重画

drawSprite.graphics.clear();

drawSprite.graphics.lineStyle(1,0x000000);

stage.addEventListener(Event.ENTER_FRAME, onEnterFrames);



}



private function mouseDownEvent(event:MouseEvent):void

{

stage.addEventListener(MouseEvent.MOUSE_MOVE, mouseMoveEvent);

drawSprite.graphics.moveTo(this.mouseX, this.mouseY);

pointAry.push(this.mouseX + "," + this.mouseY);



}

private function mouseupEvent(event:MouseEvent):void

{

stage.removeEventListener(MouseEvent.MOUSE_MOVE, mouseMoveEvent);

}



private function mouseMoveEvent(event:MouseEvent):void

{

drawSprite.graphics.lineTo(this.mouseX, this.mouseY);

pointAry.push(this.mouseX + ";" + this.mouseY);

}
```

```
private function onEnterFrames(event:Event):void
{

if(count < pointAry.length)
{
var str:String = pointAry[count];
//如果是 moveto 记号
if(str.indexOf(",") != -1)
{
var ary:Array = str.split(",");
drawSprite.graphics.moveTo(ary[0], ary[1]);
}
else
{
var ary2:Array = str.split(";");
drawSprite.graphics.lineTo(ary2[0], ary2[1]);
}
count++;
}
else
{
//结束后删除事件
stage.removeEventListener(Event.ENTER_FRAME, onEnterFrames);
}
} } }
```

在线直接打开本地 MP3 文件（播放二进制格式的 MP3 文件）

作者：粉色男孩

文章来源：http://hi.baidu.com/fsnhf/blog/item/be40b0f04d076dca7931aa28.html

在线直接打开本地 MP3 文件（播放二进制格式的 MP3 文件）

我们知道在 Sound 类中没有 loadBytes 方法，所以我们无法直接播放本地 MP3 文件，但是牛人总是有办法 的，http://www.flexiblefactory.co.uk/flexible/?p=46，根据他的类库，我写一个小例子做个测试，感觉虽 然可以，但还不是太完善，好像还有些 BUG。

我们还是先看一下效果吧：

http://remotedu.net/blog/article/52.htm

他的原理是：通过 FlashPlayer10 的 API 将文件以二进制形式载入，然后将此 MP3 文件转成一个 swf 文件，并根据 MP3 文件生成 SoundClass 类，然后播放此声音。

下面是我写的这个测试程序：

```
package
{
import flash.display.Sprite;
import flash.events.Event;
import flash.events.MouseEvent;
import flash.media.SoundChannel;
import flash.net.FileFilter;
import flash.net.FileReference;
import org.audiofx.mp3.MP3FileReferenceLoader;
import org.audiofx.mp3.MP3SoundEvent;

/**
* ...
* @author Jaja
*/
public class Main extends Sprite
{
private var mp3Type:FileFilter = new FileFilter("mp3 文件(*.mp3)", "*.mp3");
private var fr:FileReference;
private var mp3Loader:MP3FileReferenceLoader;
private var channel:SoundChannel;

public function Main():void
{
if (stage) init();
else addEventListener(Event.ADDED_TO_STAGE, init);
}

private function init(e:Event = null):void
```

```
{
removeEventListener(Event.ADDED_TO_STAGE, init);
// entry point

_btn.addEventListener(MouseEvent.CLICK, browse);
}

private function browse(event:MouseEvent):void {
fr = new FileReference;
fr.addEventListener(Event.Select, selectFile);
fr.browse([mp3Type]);
}

private function selectFile(event:Event):void {
fr.removeEventListener(Event.Select, selectFile);

_txt.text = fr.name;

mp3Loader = new MP3FileReferenceLoader;
mp3Loader.addEventListener(MP3SoundEvent.COMPLETE, mp3Loaded);
mp3Loader.getSound(fr);
}

private function mp3Loaded(event:MP3SoundEvent):void {
mp3Loader.removeEventListener(MP3SoundEvent.COMPLETE, mp3Loaded);

if (channel != null) {
channel.stop();
}
channel = event.sound.play();
}

}

}
```

AS3 制作五子棋游戏

作者：不详

文章来源：http://hi.baidu.com/%B0%B5%BA%DA%B2%E0%CE%C0/blog/item/726f5bdd1773bc3e5982ddeb.html

这节我们来写类文件！新建一个名为“Classes”的文件夹，在这个文件夹添加两个类文件！

Chessman 类文件代码：

```
package Classes{
import flash.display.MovieClip;
import flash.events.Event;
public class Chessman extends MovieClip{
   private var inc:uint = 0;
   public var bPlayer:Boolean = false;
   public function Chessman(){
    this.addEventListener(Event.ENTER_FRAME,twinkle);
   }
   public function twinkle(e:Event):void{
    if(!bPlayer){
     if(inc<15){
      this.alpha = ((inc%5)/5) + .2;
      inc ++;
     }else{
      this.removeEventListener(Event.ENTER_FRAME,twinkle);
     }
    }
   }
}
}
```

GobangDoc 类文件代码：

```
package Classes{
import flash.display.*;
import flash.events.MouseEvent;
import flash.geom.*;
import flash.text.TextField;
/*
* 五子棋初级教程
*
* @author Dingo
* @version 1.0
* @date 070829
*/

public class GobangDoc extends MovieClip {
   //棋盘格宽度
```

```
private const gridsize:Number = 20;
//棋盘格数
private const gridnum:Number = 15;
//没有棋子为0,黑子为1，白子为2
private const NOTHING:uint = 0;
private const BLACK:uint = 1;
private const WHITE:uint = 2;
//现在轮到哪一方出子
private var crtSide:uint = WHITE;
//玩家的棋子
private var mySide:uint = WHITE;
//对手的棋子
private var otherSide:uint;
//是否可以进行游戏
private var canPlay:Boolean = false;
//记录盘面状态的数组
private var aGridState:Array = [];
//记录盘面上的棋子的数组
private var aChessmen:Array = [];
//棋子的几种状态
public const STWO:int = 2;//眠二
public const FTWO:int = 4;//假活二
public const STHREE:int = 5;//眠三
public const TWO:int = 8;//活二
public const SFOUR:int = 12;//冲四
public const FTHREE:int = 15;//假活三
public const THREE:int = 40;//活三
public const FOUR:int = 90;//活四
public const FIVE:int = 200;//五连
//玩家的棋形表
private var aPlayer:Array = [];
//对手的棋形表
private var aOpponent:Array = [];
//八个方向，从左上角开始顺时针
private var dir:Array = [[-1,-1],[0,-1],[1,-1],[1,0],[1,1],[0,1],[-1,1],[-1,0]];

public function GobangDoc() {
 mcGameState.visible = false;
 otherSide = WHITE + BLACK - mySide;
 //初始化盘面数组
 for (var i:uint=0; i<gridnum; i++) {
  aGridState[i] = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
 }
 mcChessboard.addEventListener(MouseEvent.MOUSE_DOWN,AddMyChessman);
 btnStart.addEventListener(MouseEvent.CLICK,btnStart_Handler);
 btnReplay.addEventListener(MouseEvent.CLICK,btnReplay_Handler);
 mcSelectChessman.addEventListener(MouseEvent.MOUSE_DOWN,selectChessman);
}
//初始化棋盘
```

```
private function init():void{
 btnStart.visible = false;
 for(var i:int=0;i<aChessmen.length;i++){
  mcChessboard.removeChild(aChessmen[i]);
 }
 for (var j:uint=0; j<gridnum; j++) {
  aGridState[j] = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
 }
 aChessmen = [];
 canPlay = true;
}
//玩家添加棋子
public function AddMyChessman(e:MouseEvent):void {
 //不能添加棋子的状态（棋局未开始、对方走、棋子没有落在棋盘里）
 if(!canPlay || crtSide != mySide || e.target.name != "mcChessboard")
  return;
 if (mySide == crtSide) {
  //计算鼠标落在哪一格
  var crtx:uint = Math.floor(e.localX/gridsize);
  var crty:uint = Math.floor(e.localY/gridsize);
  //如果这一格已经有棋子就返回
  if (aGridState[crty][crtx])
   return;
  //创建棋子
  var chessman:Chessman;
  if (mySide == BLACK) {
   chessman = new BlackChessman();
  } else {
   chessman = new WhiteChessman();
  }
  chessman.bPlayer = true;
  aGridState[crty][crtx] = mySide;
  chessman.x = (crtx + .5) * gridsize;
  chessman.y = (crty + .5) * gridsize;
  aChessmen.push(chessman);
  mcChessboard.addChild(chessman);
  checkWinner(crtx,crty,crtSide);
  //对方走
  crtSide = WHITE + BLACK - mySide;
  //计算机走
  var opos:Array = CalculateState(crtSide);
  var cx:int = opos[0];
  var cy:int = opos[1];
  AddChessman(cx,cy);
  checkWinner(cx,cy,crtSide);
  crtSide = mySide;
 }
}
```

```
//计算机添加棋子
public function AddChessman(toX:int,toY:int):void {
 if(!canPlay)
  return;
 var autox:int = toX;
 var autoy:int = toY;
 var chessman:Chessman;
 if (mySide == BLACK) {
  chessman = new WhiteChessman();
 } else {
  chessman = new BlackChessman();
 }
 chessman.x = (autox + .5)*gridsize;
 chessman.y = (autoy + .5)*gridsize;
 aGridState[autoy][autox] = (BLACK + WHITE) - mySide;
 aChessmen.push(chessman);
 mcChessboard.addChild(chessman);
}
//评估棋盘上每一格的分值，返回得分最高的棋格坐标
public function CalculateState(side):Array{
 var i:int,j:int,k:int;
 var otherside:int = WHITE + BLACK - side;
 //填充玩家的棋形表和对手的棋形表
 for(i = 0;i<gridnum;i++){
  for(j = 0;j<gridnum;j++){
   if(aGridState[i][j] != NOTHING){
    aOpponent[i * gridnum + j] = {val:-1,x:j,y:i};
    aPlayer[i * gridnum + j] = {val:-1,x:j,y:i};
   }
   else{
    var v1 = getScore(aGridState,j,i,side);
    aOpponent[i * gridnum + j] = {val:v1,x:j,y:i};
    var v2 = getScore(aGridState,j,i,otherside);
    aPlayer[i * gridnum + j] = {val:v2,x:j,y:i};
   }
  }
 }
 //取得分值最大的棋格
 var maxO:Object = sortArray(aOpponent);
 var maxP:Object = sortArray(aPlayer);
 var apos:Array = [0,0];
 if(maxO.val < maxP.val)
  apos = [maxP.x,maxP.y];
 else
  apos = [maxO.x,maxO.y];
 return apos
}
//检查赢家
private function checkWinner(xp:int,yp:int,side:int){
```

```
    var str:String = (side * 11111).toString();
    var winner:int = 0;
    var str1:String = getXLine(aGridState,xp,yp,side).join("");
    var str2:String = getYLine(aGridState,xp,yp,side).join("");
    var str3:String = getXYLine(aGridState,xp,yp,side).join("");
    var str4:String = getYXLine(aGridState,xp,yp,side).join("");
    if(str1.indexOf(str)>-1 || str2.indexOf(str)>-1 || str3.indexOf(str)>-1 ||
str4.indexOf(str)>-1)
     winner = side;
    if(winner){
     doWin(winner);
    }
   }
   //取胜后触发的事件
   private function doWin(side:int):void{
    //现实游戏结果说明
    mcGameState.visible = true;
    //关闭棋局
    canPlay = false;
    //期盼设为半透明
    mcChessboard.alpha = .5;
    //根据玩家输赢展示不同的游戏结果
    if(side == mySide){
     mcGameState.gotoAndStop("win");
    }
    else{
     mcGameState.gotoAndStop("lose");
    }
   }
   //为数组排序的方法
   private function sortArray(arr):Object{
    var arrLen:int = arr.length;
    var ar:Array = [];
    for(var j=0;j<arrLen;j++){
     ar[j] = arr[j];
    }
    //以数字方式对"val"字段进行排序
    ar.sortOn("val",Array.NUMERIC );
    return ar[ar.length-1];
   }
   //取得指定棋格的分数
   private function getScore(arr:Array,xp:int,yp:int,side:int):int{
    var s0:int = AnalysisLine(getXLine(arr,xp,yp,side),side);
    var s1:int = AnalysisLine(getYLine(arr,xp,yp,side),side);
    var s2:int = AnalysisLine(getXYLine(arr,xp,yp,side),side);
    var s3:int = AnalysisLine(getYXLine(arr,xp,yp,side),side);
    return (s0 + s1 + s2 + s3);
   }
```

```
// -- 取得游戏中的一方在指定位置左右两边 5 格以内的状态
private function getXLine(aposition:Array,xp:int,yp:int,side:int):Array{
 var arr:Array = [];
 var xs:int,ys:int,xe:int,ye:int;
 //起始位置
 xs = xp - 5>0 ? xp - 5:0;
 //结束位置
 xe = xp + 5>= gridnum?gridnum:xp + 5;
 for(var i:int=xs;i<=xe;i++){
  if(i == xp)
   arr.push(side);
  else{
   arr.push(aGridState[yp][i])
  }
 }
 return arr;
}
// | 取得游戏中的一方在指定位置上下两边 5 格以内的状态
private function getYLine(aposition:Array,xp:int,yp:int,side:int):Array{
 var arr:Array = [];
 var xs:int,ys:int,xe:int,ye:int;
 //起始位置
 ys = yp - 5>0 ? yp - 5:0;
 //结束位置
 ye = yp + 5>= gridnum?gridnum:yp + 5;
 for(var i:int=ys;i<ye;i++){
  if(i == yp)
   arr.push(side);
  else{
   arr.push(aposition[i][xp])
  }
 }
 return arr;
}
// \ 取得游戏中的一方在指定位置左上右下两边 5 格以内的状态
private function getXYLine(aposition:Array,xp:int,yp:int,side:int):Array{
 var arr:Array = [];
 var xs:int,ys:int,xe:int,ye:int;
 //起始位置
 xs = yp > xp ? 0 : xp - yp;
 ys = xp > yp ? 0 : yp - xp;
 //结束位置
 xe = gridnum - ys;
 ye = gridnum - xs;
 var pos:int;
 for(var i:int=0;i<(xe-xs<ye-ys?xe-xs:ye-ys);i++){
   if(ys + i == yp && xs + i == xp){
    arr.push(side);
    pos = i;
```

```
    }
    else{
     arr.push(aposition[ys + i][ xs + i]);
    }
  }
  arr = arr.slice(pos-4>0?pos-4:0,pos+5>arr.length?arr.length:pos+5);
  return arr;
}
// / 取得游戏中的一方在指定位置左下右上两边 5 格以内的状态
private function getYXLine(aposition:Array,xp:int,yp:int,side:int):Array{
  var arr:Array = [];
  var xs:int,ys:int,xe:int,ye:int;
  var num:int = gridnum;
  var half:int = Math.ceil(gridnum/2);
  //起始位置
  xs = xp + yp < num?0:(xp + yp - num + 1);
  ys = xs;
  //结束位置
  xe = xp + yp >= num?num-1:(xp + yp);
  ye = xe;
  var pos:int;
  for(var i:int=0;i<(xp + yp>=num?2*num-xp-yp-1:xp+yp+1);i++){
    if(ye - i == yp && xs + i == xp){
     arr.push(side);
     pos = i;
    }
    else
     arr.push(aposition[ye - i][ xs + i]);
  }
  arr = arr.slice(pos-4>0?pos-4:0,pos+5>arr.length?arr.length:pos+5);
  return arr;
}
//评估游戏中的一方在指定位置落子后某一方向可能取得的分值
private function AnalysisLine(aline:Array,side:int):int{
  var otherside:int = WHITE + BLACK - side;
  //以下注释中 * 为本方棋子，o 为对方棋子，_ 为空格
  // *****
  var five:String = (side * 11111).toString();
  // _****
  var four:String = "0" + (side * 1111).toString() + "0";
  // _***_
  var three:String = "0" + (side * 111).toString() + "0";
  // _**_
  var two:String = "0" + (side * 11).toString() + "0";
  // _*_*_
  var jtwo:String = "0" + (side * 101).toString() + "0";
  // ****_
  var lfour:String = otherside.toString() + (side * 1111).toString() + "0";
  // _****
```

```
    var rfour:String = "0" + (side * 1111).toString() + otherside.toString();
    // *_***
    var l_four:String = (side * 10111).toString();
    // ***_*
    var r_four:String = (side * 11101).toString();
    // o***_
    var lthree:String = otherside.toString() + (side * 111).toString() + "0";
    // _***o
    var rthree:String = "0" + (side * 111).toString() + otherside.toString();
    // o**_
    var ltwo:String = otherside.toString() + (side * 11).toString() + "0";
    // _**o
    var rtwo:String = "0" + (side * 11).toString() + otherside.toString();
    // ***_o
    var rfthree:String = (side * 111).toString() + "0" + otherside.toString();
    // o_***
    var lfthree:String = otherside.toString() + "0" + (side * 111).toString();

    var str:String = aline.join("");
    var res:int;
    if(str.indexOf(five)>=0){
     res = FIVE;
     if(side == otherSide)
      res *=2;
    }
    else if(str.indexOf(four)>=0)
     res = FOUR;
    else if(str.indexOf(three)>=0)
     res = side!=mySide?THREE+4:THREE;
    else if(str.indexOf(two)>=0 || str.indexOf(jtwo)>=0 )
     res = TWO;
    else if(str.indexOf(lfour)>=0 || str.indexOf(rfour)>=0 || str.indexOf(l_four)>=0 ||
str.indexOf(r_four)>=0)
     res = SFOUR;
    else if(str.indexOf(lthree)>=0 || str.indexOf(rthree)>=0)
     res = STHREE;
    else if(str.indexOf(ltwo)>=0 || str.indexOf(rtwo)>=0)
     res = STWO;
    else if(str.indexOf(lfthree)>=0 || str.indexOf(rfthree)>=0)
     res = FTHREE;
    else
     res = 0;

    return res;
   }
   //开始游戏按钮触发的方法
   private function btnStart_Handler(e:MouseEvent):void{
    canPlay = true;
    if(mySide == WHITE){
```

```
      AddChessman(Math.floor(gridnum/2),Math.floor(gridnum/2));
     }
     btnStart.visible = false;
    }
    //重玩一遍按钮触发的方法
    private function btnReplay_Handler(e:MouseEvent):void{
     mcGameState.visible = false;
     mcChessboard.alpha = 1;
     init();
     if(mySide == WHITE){
      AddChessman(Math.floor(gridnum/2),Math.floor(gridnum/2));
     }
    }
    //选择棋子按钮触发的方法
    private function selectChessman(e:MouseEvent):void{
     if(canPlay){
      mcSelectChessman.buttonMode = false;
      return;
     }else{
      mcSelectChessman.buttonMode = true;
      mySide = otherSide;
      otherSide = WHITE + BLACK - mySide;
      if(mySide == WHITE){
       mcSelectChessman.gotoAndStop("white");
      }else{
       mcSelectChessman.gotoAndStop("black");
      }
      crtSide = mySide;
     }
    }

}
}
```

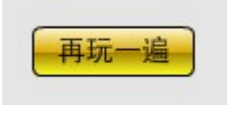
好了！代码编写完毕，下一节中我们来制作界面！

这是“五子棋游戏”的最后一节！本节中我们要来绘制界面。其实这个也不用我说太多。因为每个人心目中的样子都是不一样的！所以这个我只是起到一个抛砖引玉的作用。大家可以在这个基础之上自由发挥！

首先，舞台背景是 330*400，背景颜色最好选一个反差大一些的，比如暗灰色！

接下来我们来做两个按钮，就是开始按钮和重玩一次的按钮。这两个按钮可以用同一个背景，然后把上面的文字改一下即可！外观如下：

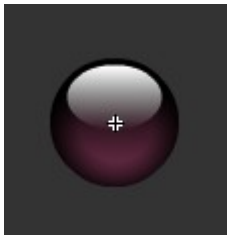
正常状态：



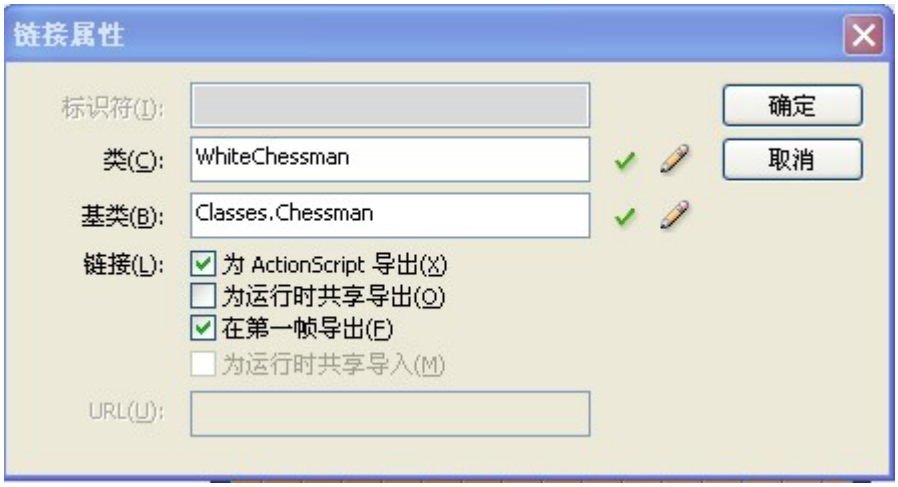
鼠标经过状态：

这样我们就制作完成了 2 个按钮！然后将他们放置到舞台上，2 个按钮要重叠，“开始游戏”这个按钮要放在上面，并且它的实例名是“btnStart”，另外一个按钮的实例名是“btnReplay”。

然后我们来做黑白棋子，这里我们就只制作黑色棋子，白色棋子制作方法完全相同，更改颜色即可！

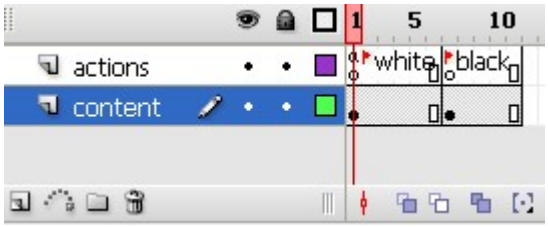


这样制作两个棋子的影片剪辑，并将他们的链接进行设置！如图：



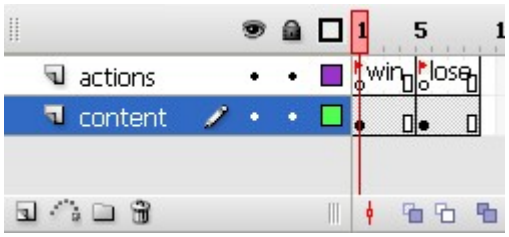
这步完成后我们将这两个影片剪辑放到一个影片剪辑中！

影片剪辑的时间轴如下：



第一帧上写上 stop()语句即可！

最后！我们来制作提示！这个影片剪辑的时间轴如下：



第一帧写上“恭喜您，您胜利了！”，第二帧写上“对不起！您输了！”。然后将这个元件放到舞台上，实例名为“mcGameState”。

好了!现在还剩下棋盘!我们可以简单的绘制!并将它放到舞台上,实例名为“mcChessboard”。还有一个黑白棋选择按钮。还是用刚才的一个元件。就是有黑白棋子的元件，放好后设置实例名为“mcSelectChessman”。

我们基本的元素都设置完成了！最后大家可以自由发挥，画出你想要的图案！这里我截取一下我的图案!

flash九宫对对碰思路讲解

作者：小 S

文章来源：http://www.xiaos8.com/article.asp?id=46

实际上对对碰这种东西不是很难的东西，但是对于新手，读懂代码还是有一定的难度，但是根据着我写的这篇文章，一步一步去看源代码，相信会有帮助

大体分析：
利用二维数组记录每个小方块的状态，并且为每一个小方块做一个独立的类，进行数据保存等
从 0-9 每一个数字代表了小方块当前的状态形状，同样的数字则表示当前形状一样
在每移动一次的时候，都会去验证是否有新的成功项需要消除
验证的方法是采用遍历每个小方块可能的成功项，并且保存到一个数组中，然后将得到的数组数据，清除同样数据，并执行消除方块

文件作用讲解：
Vessel.as 类：程序主类
Thing.as 类：定义每个小方块的类
Painting.as 类：用来生成各种不同小方块的类
TwoArray.as 类：导入的二维数组类

纯 AS 打造的颜色拾取器

作者：粉色男孩

文章来源：http://hi.baidu.com/fsnhf/blog/item/76e0be0904e807a72eddd493.html

在 做 AS 开发时，使用组件会大大提高开发效率，这一方面最优秀的当然是 Flex，但是，它有一个最大的缺点，就是生成的文件太大，swf 文件主要是做为网页 的插件来使用的，如果生成的 swf 文件有 300k，客户机是 1M 宽带（实际速度 100KB/S 左右），那么用户就要等待 3 秒，这已经是一个很大的数字了。

所以 Flex 程序不适合做基于网页的应用，用它做桌面应用（AIR）目前来说还是最好的选择，如果做基于网页的应用，我们会选择用纯 AS 语言或者 flash 来开发。

如果做纯 AS 开发时，比较理想的选择是用 ASwing，但 ASwing 也只是比较理想的选择而已，不是非常理想的，它生成的 Flash 文件也比较大了(100KB 左右)，并且 ASwing 组件并没有 Flex 组件好看。

事实上 Flash 中所用的组件已经占空间很小了，但我们还不满足，因为你添加一个 Flash 组件所生成的 swf 文件最少要大于 20KB，这还是有点大，要知道中国的带宽和外国没法比，考虑到国情，我们还是尽可能的减小 swf 文件的体积吧。

以上几个框架给我们一种强制捆绑的感觉，就像是去买白菜时，卖主硬把香菜也给绑在一起，要买的话就一块买，这让我们感觉很不爽，我们希望的是买白菜就只买白菜，其它什么都不买。

有什么办法来解决这个问题呢？

有，那就是我们自己种菜，当我们想吃什么菜时就去拔什么菜。

所以我自己首先做一个 ColorPicker 组件来玩玩，也想分享给大家来玩玩。

先看一下效果：http://remotedu.net/blog/article/36.htm

下面就是这个组件源码：

```
package cn.asmax.controls
{
    import flash.display.DisplayObject;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.KeyboardEvent;
    import flash.events.MouseEvent;

    /**
    * 颜色拾取器
    * @author Jaja as-max.cn
    */
    public class ColorPicker extends Sprite
    {
        private var clicker:Clicker = new Clicker;
        private var colorArea:ColorFormArea = new ColorFormArea(colorChangeHandler);

        /**
        * 最近的一次颜色值
        */
        private var lastestColor:String = "0x000000";

        public function ColorPicker() :void
        {
            //add clicker
            super.addChild(clicker);
```

```
        colorArea.visible = false;
        super.addChild(colorArea);

        this.addEventListener(Event.ADDED_TO_STAGE, addThis);
    }

    /**
    * 获得颜色值
    */
    public function get color():uint {
        return uint(lastestColor);
    }

    /**
    * 获得颜色字符串
    */
    public function get colorString():String {
        return lastestColor;
    }

    private function colorChangeHandler(color:String):void {
        clicker.color = color;
    }

    /**
    * 控制是否显示颜色区域
    * @param    event
    */
    private function clickThis(event:MouseEvent = null):void {
        if (colorArea.visible) {
            if (this.mouseY >= colorArea.y + 30) {
                colorArea.visible = false;
                sendSelectEvent();
            }
        }else {
            colorArea.visible = true;

            if (this.stage.mouseX > stage.stageWidth - colorArea.width) {
                colorArea.x = - colorArea.width - 5;
            }else {
                colorArea.x = clicker.x + clicker.width + 5;
            }
            if (this.stage.mouseY > stage.stageHeight - colorArea.height) {
                colorArea.y = -colorArea.height + 20;
            }else {
                colorArea.y = clicker.y + 5;
            }
        }
    }
```

```
private function keyDown(event:KeyboardEvent):void {
    switch(event.keyCode) {
        case 27://Esc Key
            showLatestColor();
            break;
        case 13://Enter Key
            colorArea.visible = false;
            sendSelectEvent();
            break;
    }
}

private function lostFocus(event:MouseEvent):void {
    if (this.mouseX < colorArea.x ||
        this.mouseX > colorArea.width + colorArea.x ||
        this.mouseY < colorArea.y ||
        this.mouseY > colorArea.height + colorArea.y)
    {
        showLatestColor();
    }
}

private function lostSystemFocus(event:Event):void {
    showLatestColor();
}

/**
* 显示最近的一次颜色
*/
private function showLatestColor():void {
    colorArea.visible = false;
    clicker.color = lastestColor;
}

private function addThis(event:Event):void {
    if(!hasEventListener(MouseEvent.CLICK))
        this.addEventListener(MouseEvent.CLICK, clickThis);
    if(!hasEventListener(Event.REMOVED_FROM_STAGE))
        this.addEventListener(Event.REMOVED_FROM_STAGE, removeThis);
    stage.addEventListener(KeyboardEvent.KEY_DOWN, keyDown);
    stage.addEventListener(MouseEvent.MOUSE_DOWN, lostFocus);
    if(!hasEventListener(Event.DEACTIVATE))
        this.addEventListener(Event.DEACTIVATE, lostSystemFocus);
}

private function removeThis(event:Event):void {
    this.removeEventListener(MouseEvent.CLICK, clickThis);
    this.removeEventListener(Event.REMOVED_FROM_STAGE, removeThis);
```

```
        stage.removeEventListener(KeyboardEvent.KEY_DOWN, keyDown);
        stage.removeEventListener(MouseEvent.MOUSE_DOWN, lostFocus);
        this.removeEventListener(Event.DEACTIVATE, lostSystemFocus);
    }

    /**
    * @eventType flash.events.Event.SELECT
    */
    [Event(name = "select", type = "flash.events.Event")]
    private function sendSelectEvent():void {
        lastestColor = clicker.color;
        dispatchEvent(new Event(Event.SELECT));
    }

    //disable functions
    public override function addChild(child:DisplayObject):DisplayObject
    {
        throw new Error("此方法不可用");
    }
    public override function addChildAt(child:DisplayObject, index:int):DisplayObject
    {
        throw new Error("此方法不可用");
    }
    public override function contains(child:DisplayObject):Boolean
    {
        throw new Error("此方法不可用");
    }
    public override function removeChild(child:DisplayObject):DisplayObject
    {
        throw new Error("此方法不可用");
    }
    public override function removeChildAt(index:int):DisplayObject
    {
        throw new Error("此方法不可用");
    }
    public override function setChildIndex(child:DisplayObject, index:int):void
    {
        throw new Error("此方法不可用");
    }
    public override function swapChildren(child1:DisplayObject, child2:DisplayObject):void
    {
        throw new Error("此方法不可用");
    }
    public override function swapChildrenAt(index1:int, index2:int):void
    {
        throw new Error("此方法不可用");
    }
    public override function set width(value:Number):void
    {
```

```
            throw new Error("尝试对只读属性进行赋值");
        }
        public override function set height(value:Number):void
        {
            throw new Error("尝试对只读属性进行赋值");
        }

    }

}


import flash.display.CapsStyle;
import flash.display.JointStyle;
import flash.display.Shape;
import flash.display.Sprite;
import flash.events.Event;
import flash.events.MouseEvent;
import flash.filters.BevelFilter;
import flash.filters.BitmapFilterQuality;
import flash.filters.BitmapFilterType;
import flash.text.TextField;
import flash.text.TextFieldType;
class Clicker extends Sprite {

    /**
    * 拾色器的头
    */
    private var myColorArea:ClickColorArea = new ClickColorArea;

    /**
    * 包含的颜色的字符串表示形式
    */
    private var theColor:String = "0x000000";
    public function Clicker():void {
        with(graphics){
            lineStyle(1, 0xFFFFFF, 1, true, "normal", CapsStyle.SQUARE, JointStyle.MITER);
            moveTo(0, 25);
            lineTo(0, 0);
            lineTo(25, 0);
            lineStyle(1, 0xAAAAAA, 1, true, "normal", CapsStyle.SQUARE, JointStyle.MITER);
            lineTo(25, 25);
            lineTo(0, 25);
            lineStyle(1, 0xEEEEEE, 1, true, "normal", CapsStyle.SQUARE, JointStyle.MITER);
            moveTo(1, 24);
            lineTo(1, 1);
            lineTo(24, 1);
            lineTo(24, 24);
            lineTo(1, 24);
```

```
            lineStyle(1, 0xCCCCCC, 1, true, "normal", CapsStyle.SQUARE, JointStyle.MITER);
            moveTo(2, 23);
            lineTo(2, 2);
            lineTo(23, 2);
            lineStyle(1, 0xFFFFFF, 1, true, "normal", CapsStyle.SQUARE, JointStyle.MITER);
            lineTo(23, 23);
            lineTo(2, 23);
        }

        myColorArea.x = 2.5;
        myColorArea.y = 2.5;
        addChild(myColorArea);

        var blackArr:Shape = new Shape;
        blackArr.graphics.beginFill(0xEEEEEE);
        blackArr.graphics.drawRect(0, 0, 8, 6);
        blackArr.graphics.endFill();
        blackArr.graphics.beginFill(0);
        blackArr.graphics.lineStyle(.01, 0xCCCCCC, 1, true, "normal", CapsStyle.SQUARE,
JointStyle.MITER);
        blackArr.graphics.moveTo(1, 1);
        blackArr.graphics.lineTo(7, 1);
        blackArr.graphics.lineTo(4, 5);
        blackArr.graphics.lineTo(1, 1);
        blackArr.graphics.endFill();
        blackArr.x = this.width - blackArr.width - 2;
        blackArr.y = this.height - blackArr.height - 2;
        addChild(blackArr);
    }

    /**
    * 改变选中的颜色
    * @param    color
    */
    public function changeColor(color:uint):void {
        myColorArea.changeColor(color);
    }

    /**
    * 设置颜色
    */
    public function set color(color:String):void {
        theColor = color;
        myColorArea.changeColor(uint(color));
    }
    /**
    * 获得颜色
    */
    public function get color():String {
```

```
        return theColor;
    }
}

class ClickColorArea extends Sprite {

    /**
    * 拾色器头显示当前颜色的部分
    * @param    color 默认为黑色
    */
    public function ClickColorArea(color:uint = 0x000000):void {
        changeColor(color);
    }

    /**
    * 改变此显示区域的颜色
    * @param    color
    */
    public function changeColor(color:uint):void {
        graphics.clear();
        graphics.beginFill(color);
        graphics.drawRect(0, 0, 20, 20);
        graphics.endFill();
    }
}

class ColorForm extends Sprite {

    private var theHandler:Function;

    /**
    * 颜色块的颜色
    */
    private var theColor:String = "";

    private var sharp:Shape = new Shape;

    /**
    * 颜色块
    * @param    color
    * @param    mouseOverHandler
    */
    public function ColorForm(color:String, mouseOverHandler:Function = null):void {
        theColor = color;
        theHandler = mouseOverHandler;

        //绘制中部
        graphics.beginFill(uint(color));
        graphics.drawRect(0, 0, 15, 15);
```

```
            graphics.endFill();
            //绘制外圈
            sharp.graphics.lineStyle(1, 0xFFFFFF);
            sharp.graphics.drawRect(0, 0, 15, 15);
            addChild(sharp);
            sharp.visible = false;

            this.addEventListener(Event.REMOVED_FROM_STAGE, removeThis);
            this.addEventListener(MouseEvent.MOUSE_OVER, mouseOverThis);
        }

        private function mouseOverThis(event:MouseEvent):void {
            if (Boolean(theHandler)) {
                theHandler(theColor);
            }
        }

        private function mouseOutThis(event:MouseEvent):void {
            select = false;
        }

        private function removeThis(event:Event):void {
            this.removeEventListener(Event.REMOVED_FROM_STAGE, removeThis);
            this.removeEventListener(MouseEvent.MOUSE_OVER, mouseOverThis);
        }

        /**
        * 获得或设置选中状态
        */
        public function set select(value:Boolean):void {
            sharp.visible = value;
        }
        public function get select():Boolean {
            return sharp.visible;
        }

        /**
        * 获得本色块的颜色
        */
        public function get color():String {
            return theColor;
        }
    }

    class ColorFormArea extends Sprite {

        private var colorArr:Array = ["00", "33", "66", "99", "CC", "FF"];

        //color text
```

```
    private var txt:TextField = new TextField;
    /**
    * 颜色块数组
    */
    private const RECTS_ARR:Array = new Array;

    /**
    * 颜色改变时的处理函数
    */
    private var theHandler:Function;

    /**
    * 颜色块区域
    */
    private var colorRects:Sprite = new Sprite;

    /**
    * 构造一个新的 ColorFormArea 实例
    */
    public function ColorFormArea(colorChangeHandler:Function = null):void {
        theHandler = colorChangeHandler;

        var bg:Sprite = new Sprite;
        with(bg){
            graphics.beginFill(0xFFFFFF);
            graphics.drawRect(0, 0, 308, 230);
            graphics.endFill();
            graphics.lineStyle(1, 0xCCCCCC, 1, true, "normal", CapsStyle.SQUARE, JointStyle.MITER);
            graphics.moveTo(1, 1);
            graphics.lineTo(1, 229);
            graphics.moveTo(308, 1);
            graphics.lineTo(308, 229);
        }
        //添加滤镜
        var bevel:BevelFilter = new BevelFilter(1.5, 90, 0xFFFFFF, 1, 0x666666, 1, 0, 4, 1,
BitmapFilterQuality.LOW, BitmapFilterType.INNER, false);
        bg.filters = [bevel];
        addChild(bg);

        //add text
        txt.width = 60;
        txt.height = 20;
        txt.border = true;
        txt.type = TextFieldType.INPUT;
        txt.x = 10;
        txt.y = 5;
        txt.maxChars = 7;
        txt.restrict = "0-9a-f#";
        txt.text = "0x000000";
```

```
        txt.addEventListener(Event.CHANGE, txtChange);
        addChild(txt);

        //add color rects bg
        var colorBG:Sprite = new Sprite;
        colorBG.graphics.lineStyle(1);
        colorBG.graphics.beginFill(0);
        colorBG.graphics.drawRect(0, 0, 300, 300);
        colorBG.graphics.endFill();
        addChild(colorBG);

        //add color rects
        colorRects.x = txt.x + 1;
        colorRects.y = txt.y + txt.height + 5;
        addChild(colorRects);
        for (var i:int = 0; i < 18; i++) {
            for (var j:int = 0; j < 12; j++) {
                var color:String = "";
                color = "0x" + colorArr[Math.floor(i / 6) + Math.floor(j / 6) * 3] + colorArr[i % 6]
+ colorArr[j % 6];
                var colorForm:ColorForm = new ColorForm(color, selectColorHandler);
                colorForm.x = i * (colorForm.width);
                colorForm.y = j * (colorForm.height);
                colorRects.addChild(colorForm);
                RECTS_ARR.push(colorForm);
            }
        }
        colorBG.width = colorRects.width + 1;
        colorBG.height = colorRects.height + 1;
        colorBG.x = colorRects.x - 1;
        colorBG.y = colorRects.y - 1;

        this.addEventListener(Event.REMOVED_FROM_STAGE, removeThis);
    }

    /**
    * 移除侦听器
    * @param    event
    */
    private function removeThis(event:Event):void {
        txt.removeEventListener(Event.CHANGE, txtChange);
        this.removeEventListener(Event.REMOVED_FROM_STAGE, removeThis);
    }

    private function selectColorHandler(color:String):void {
        txt.text = "#" + color.slice(2, 8);
        txtChange();

        callHandler(color);
```

```
    }

    /**
    * 最近一次选中的方块
    */
    private var latestIndex:int = 0;

    private function txtChange(event:Event = null):void {
        var txtValue:String = txt.text;
        var txtColor:String = "";

        if (txtValue.slice(0, 1) == "#") {
            txt.maxChars = 7;
            txtColor = "0x" + txtValue.slice(1, 7);
        }else {
            txt.maxChars = 6;
            txtColor = "0x" + txtValue.slice(0, 6);
        }
        callHandler(txtColor);

        RECTS_ARR[latestIndex].select = false;
        for (var i:int = 0; i < RECTS_ARR.length; i++) {
            var colorForm:ColorForm = RECTS_ARR[i] as ColorForm;
            if (uint(colorForm.color) == uint(txtColor)) {
                colorForm.select = true;
                latestIndex = i;
                break;
            }
        }
    }

    /**
    * 颜色改变时调用此函数
    * @param    color
    */
    private function callHandler(color:String):void {
        if (Boolean(theHandler)) {
            theHandler(color);
        }
    }
}
```

请下载源文件查看使用示例。

物理运动学公式汇总

作者：蜡笔工作室

文章来源：http://hi.baidu.com/%C0%AF%B1%CA%B9%A4%D7%F7%CA%D2/blog/item/1ea017c43cf579c238db496c.html

一、质点的运动（1）------直线运动

1）匀变速直线运动

1. 平均速度 V＝s/t（定义式）

2. 有用推论 Vt*Vt-Vo*Vo＝2as

3. 中间时刻速度 Vt/2＝V 平＝(Vt+Vo)/2

4. 末速度 Vt＝Vo+at

5. 中间位置速度 Vs/2＝[(Vo*Vo+Vt*Vt)/2]1/2

6. 位移 s＝V 平 t＝Vot+at*t/2＝Vt/2t

7. 加速度 a＝(Vt-Vo)/t｛以 Vo 为正方向，a 与 Vo 同向(加速)a>0；反向则 a<0｝

8. 实验用推论 Δs＝aT*T｛Δs 为连续相邻相等时间(T)内位移之差｝

9. 主要物理量及单位:初速度(Vo):m/s；加速度(a):m/s*s；末速度(Vt):m/s；时间(t)秒(s)；位移(s):米（m）；路程:米；速度单位换算：1m/s=3.6km/h。

注：

(1)平均速度是矢量;

(2)物体速度大,加速度不一定大;

(3)a=(Vt-Vo)/t 只是量度式，不是决定式;

(4)其它相关内容：质点、位移和路程、参考系、时间与时刻〔见第一册 P19〕/s--t 图、v--t 图/速度与速率、瞬时速度〔见第一册 P24〕。

2)自由落体运动

1. 初速度 Vo＝0?

2. 末速度 Vt＝gt

3. 下落高度 h＝gt*t/2（从 Vo 位置向下计算）

4. 推论 Vt*Vt＝2gh

注:

(1)自由落体运动是初速度为零的匀加速直线运动，遵循匀变速直线运动规律；

(2)a＝g＝9.8m/s*s≈10m/s*s（重力加速度在赤道附近较小,在高山处比平地小，方向竖直向下）。

（3)竖直上抛运动

1.位移 s＝Vot-gt*t/2

2.末速度 Vt＝Vo-gt（g=9.8m/s*s≈10m/s*s）

3.有用推论 Vt*Vt-Vo*Vo＝-2gs

4.上升最大高度 Hm＝Vo*Vo/2g(抛出点算起）

5.往返时间 t＝2Vo/g（从抛出落回原位置的时间）

注:

(1)全过程处理:是匀减速直线运动，以向上为正方向，加速度取负值；

(2)分段处理：向上为匀减速直线运动，向下为自由落体运动，具有对称性；

(3)上升与下落过程具有对称性,如在同点速度等值反向等。

二、质点的运动（2）----曲线运动、万有引力

1)平抛运动

1.水平方向速度：Vx＝Vo

2.竖直方向速度：Vy＝gt

3.水平方向位移：x＝Vot

4.竖直方向位移：y＝gt*t/2

5.运动时间 t＝(2y/g) 1/2(通常又表示为(2h/g)1/2)

6.合速度 Vt＝(Vx*Vx+Vy*Vy)1/2＝[Vo*Vo+(gt)*(gt)]1/2 合速度方向与水平夹角 β:tgβ＝Vy/Vx＝gt/V0

7.合位移：s＝(x*x+y*y)1/2,位移方向与水平夹角 α:tgα＝y/x＝gt/2Vo

8.水平方向加速度：ax=0；竖直方向加速度：ay＝g

注：

(1)平抛运动是匀变速曲线运动，加速度为 g，通常可看作是水平方向的匀速直线运与竖直方向的自由落体运动的合成；

(2)运动时间由下落高度 h(y)决定与水平抛出速度无关；

（3）θ 与 β 的关系为 tgβ＝2tgα；

（4）在平抛运动中时间 t 是解题关键；(5)做曲线运动的物体必有加速度，当速度方向与所受合力(加速度)方向不在同一直线上时，物体做曲线运动。

2）匀速圆周运动

1.线速度 V＝s/t＝2πr/T

2.角速度 ω＝Φ/t＝2π/T＝2πf

3.向心加速度 a＝V2/r＝ω2r＝(2π/T)2r

4.向心力 F 心＝mV2/r＝mω2r＝mr(2π/T)2＝mωv=F 合

5.周期与频率：T＝1/f

6.角速度与线速度的关系：V＝ωr

7.角速度与转速的关系 ω＝2πn(此处频率与转速意义相同)

8.主要物理量及单位：弧长(s):米(m)；角度(Φ)：弧度（rad）；频率（f）：赫（Hz）；周期（T）：秒（s）；转速（n）：r/s；半径(r):米（m）；线速度（V）：m/s；角速度（ω）：rad/s；向心加速度：m/s*s。

注：

（1）向心力可以由某个具体力提供，也可以由合力提供，还可以由分力提供，方向始终与速度方向垂直，指向圆心；

（2）做匀速圆周运动的物体，其向心力等于合力，并且向心力只改变速度的方向，不改变速度的大小，因此物体的动能保持不变，向心力不做功，但动量不断改变。

3)万有引力

1.开普勒第三定律：T2/R3＝K(＝4π2/GM) {R:轨道半径，T:周期，K:常量(与行星质量无关，取决于中心天体的质量)}

2.万有引力定律：F＝Gm1m2/r2（G＝6.67×10-11Nm2/kg2，方向在它们的连线上）

3.天体上的重力和重力加速度：GMm/R2＝mg；g＝GM/R2 {R:天体半径(m)，M：天体质量（kg）}

4.卫星绕行速度、角速度、周期：V＝(GM/r)1/2；ω＝(GM/r3)1/2；T＝2π(r3/GM)1/2 {M：中心天体质量}

5.第一(二、三)宇宙速度 V1＝(g 地 r 地)1/2＝(GM/r 地)1/2＝7.9km/s；V2＝11.2km/s；V3＝16.7km/s

6.地球同步卫星 GMm/(r 地+h)2＝m4π2(r 地+h)/T2 {h≈36000km，h:距地球表面的高度，r 地:地球的半径}

注:

(1)天体运动所需的向心力由万有引力提供,F 向＝F 万；

(2)应用万有引力定律可估算天体的质量密度等；

(3)地球同步卫星只能运行于赤道上空，运行周期和地球自转周期相同；

(4)卫星轨道半径变小时,势能变小、动能变大、速度变大、周期变小（一同三反）；

(5)地球卫星的最大环绕速度和最小发射速度均为 7.9km/s。

三、力（常见的力、力的合成与分解） 1）常见的力

1.重力 G＝mg（方向竖直向下，g＝9.8m/s*s≈10m/s*s，作用点在重心，适用于地球表面附近）

2.胡克定律 F＝kx｛方向沿恢复形变方向，k：劲度系数(N/m)，x：形变量(m)｝

3.滑动摩擦力 F＝μFN｛与物体相对运动方向相反，μ：摩擦因数，FN：正压力(N)｝

4.静摩擦力 0≤f 静≤fm（与物体相对运动趋势方向相反，fm 为最大静摩擦力）

5.万有引力 F＝Gm1m2/r2（G＝6.67×10-11Nm2/kg2,方向在它们的连线上）

注:

(1)劲度系数 k 由弹簧自身决定;

(2)摩擦因数 μ 与压力大小及接触面积大小无关，由接触面材料特性与表面状况等决定;

(3)fm 略大于 μFN，一般视为 fm≈μFN;

2）力的合成与分解

1.同一直线上力的合成同向:F＝F1+F2，反向：F＝F1-F2(F1>F2)

2.互成角度力的合成：

F＝(F12+F22+2F1F2cosα)1/2（余弦定理）F1⊥F2 时:F＝(F12+F22)1/2

3.合力大小范围：|F1-F2|≤F≤|F1+F2|

4.力的正交分解：Fx＝Fcosβ，Fy＝Fsinβ（β 为合力与 x 轴之间的夹角 tgβ＝Fy/Fx）

注：

(1)力(矢量)的合成与分解遵循平行四边形定则;

（2）合力与分力的关系是等效替代关系,可用合力替代分力的共同作用,反之也成立;

(3)除公式法外，也可用作图法求解,此时要选择标度,严格作图;

(4)F1 与 F2 的值一定时,F1 与 F2 的夹角(α 角)越大，合力越小;

（5）同一直线上力的合成，可沿直线取正方向，用正负号表示力的方向，化简为代数运算。

四、动力学（运动和力）

1.牛顿第一运动定律(惯性定律）：物体具有惯性，总保持匀速直线运动状态或静止状态,直到有外力迫使它改变这种状态为止

2.牛顿第二运动定律：F 合＝ma 或 a＝F 合/ma{由合外力决定,与合外力方向一致}

3.牛顿第三运动定律：F＝-F′{负号表示方向相反,F、F′各自作用在对方，平衡力与作用力反作用力区别，实际应用：反冲运动}

4.共点力的平衡 F 合＝0，推广{正交分解法、三力汇交原理}

5.超重：FN>G，失重：FN<G{加速度方向向下，均失重，加速度方向向上，均超重}

6.牛顿运动定律的适用条件：适用于解决低速运动问题，适用于宏观物体，不适用于处理高速问题，不适用于微观粒子

注:平衡状态是指物体处于静止或匀速直线状态,或者是匀速转动。

五、振动和波（机械振动与机械振动的传播）

1.简谐振动 F＝-kx{F:回复力，k:比例系数，x:位移，负号表示 F 的方向与 x 始终反向}

2.单摆周期 T＝2π(l/g)1/2{l:摆长(m)，g:当地重力加速度值，成立条件:摆角 θ<100;l>>r}

六、冲量与动量(物体的受力与动量的变化）

1.动量：p＝mv{p:动量(kg/s)，m:质量(kg)，v:速度(m/s)，方向与速度方向相同}

3.冲量：I＝Ft{I:冲量(Ns)，F:恒力(N)，t:力的作用时间(s)，方向由 F 决定}

4.动量定理：I＝Δp 或 Ft＝mvt－mvo{Δp:动量变化 Δp＝mvt－mvo，是矢量式}

5.动量守恒定律：p 前总＝p 后总或 p＝p’′也可以是 m1v1+m2v2＝m1v1′+m2v2′

6.弹性碰撞：Δp＝0；ΔEk＝0{即系统的动量和动能均守恒}

7.非弹性碰撞 Δp＝0；0<ΔEK<ΔEKm{ΔEK：损失的动能，EKm：损失的最大动能}

8.完全非弹性碰撞 Δp＝0；ΔEK＝ΔEKm{碰后连在一起成一整体}

9.物体 m1 以 v1 初速度与静止的物体 m2 发生弹性正碰:

v1′＝(m1-m2)v1/(m1+m2)v2′＝2m1v1/(m1+m2)

10.由 9 得的推论-----等质量弹性正碰时二者交换速度(动能守恒、动量守恒)

11.子弹 m 水平速度 vo 射入静止置于水平光滑地面的长木块 M，并嵌入其中一起运动时的机械能损失 E 损
=mVo*Vo/2-(M+m)Vt*Vt/2＝fs 相对{vt:共同速度，f:阻力，s 相对子弹相对长木块的位移}

一个水波效果类

作者：DN_Web

文章来源：http://hi.baidu.com/dn%5Fweb/blog/item/dfe37939531a11f5b311c745.html

转载自http://www.5las.com

Rippler 类可以轻松创建水波效果

```
package
{
    import flash.display.BitmapData;
    import flash.display.BitmapDataChannel;
    import flash.display.BlendMode;
    import flash.display.DisplayObject;
    import flash.events.Event;
    import flash.filters.ConvolutionFilter;
    import flash.filters.DisplacementMapFilter;
    import flash.geom.ColorTransform;
    import flash.geom.Matrix;
    import flash.geom.Point;
    import flash.geom.Rectangle;

      public class Rippler
    {
        // The DisplayObject which the ripples will affect.
        private var _source : DisplayObject;

        // Two buffers on which the ripple displacement image will be created, and swapped.
        // Depending on the scale parameter, this will be smaller than the source
        private var _buffer1 : BitmapData;
        private var _buffer2 : BitmapData;

        // The final bitmapdata containing the upscaled ripple image, to match the source
DisplayObject
        private var _defData : BitmapData;

        // Rectangle and Point objects created once and reused for performance
        private var _fullRect : Rectangle;     // A buffer-sized Rectangle used to apply
filters to the buffer
        private var _drawRect : Rectangle;    // A Rectangle used when drawing a ripple
        private var _origin : Point = new Point(); // A Point object to (0, 0) used for
the DisplacementMapFilter as well as for filters on the buffer

        // The DisplacementMapFilter applied to the source DisplayObject
        private var _filter : DisplacementMapFilter;
        // A filter causing the ripples to grow
        private var _expandFilter : ConvolutionFilter;
```

```
        // Creates a colour offset to 0x7f7f7f so there is no image offset due to the
DisplacementMapFilter
        private var _colourTransform : ColorTransform;

        // Used to scale up the buffer to the final source DisplayObject's scale
        private var _matrix : Matrix;

        // We only need 1/scale, so we keep it here
        private var _scaleInv : Number;

        /**
         * Creates a Rippler instance.
         *
         * @param source The DisplayObject which the ripples will affect.
         * @param strength The strength of the ripple displacements.
         * @param scale The size of the ripples. In reality, the scale defines the size
of the ripple displacement map (map.width = source.width/scale). Higher values are therefor
also potentially faster.
         *
         */
        public function Rippler(source : DisplayObject, strength : Number, scale : Number
= 2)
        {
            var correctedScaleX : Number;
            var correctedScaleY : Number;

            _source = source;
            _scaleInv = 1/scale;

            // create the (downscaled) buffers and final (upscaled) image data, sizes depend
on scale
            _buffer1 = new BitmapData(source.width*_scaleInv, source.height*_scaleInv,
false, 0x000000);
            _buffer2 = new BitmapData(_buffer1.width, _buffer1.height, false, 0x000000);
            _defData = new BitmapData(source.width, source.height);

            // Recalculate scale between the buffers and the final upscaled image to prevent
roundoff errors.
            correctedScaleX = _defData.width/_buffer1.width;
            correctedScaleY = _defData.height/_buffer1.height;

            // Create reusable objects
            _fullRect = new Rectangle(0, 0, _buffer1.width, _buffer1.height);
            _drawRect = new Rectangle();

            // Create the DisplacementMapFilter and assign it to the source
            _filter = new DisplacementMapFilter(_buffer1, _origin, BitmapDataChannel.BLUE,
BitmapDataChannel.BLUE, strength, strength, "wrap");
```

```
            _source.filters = [_filter];

            // Create a frame-based loop to update the ripples
            _source.addEventListener(Event.ENTER_FRAME, handleEnterFrame);

            // Create the filter that causes the ripples to grow.
            // Depending on the colour of its neighbours, the pixel will be turned white
            _expandFilter = new ConvolutionFilter(3, 3, [0.5, 1, 0.5, 1, 0, 1, 0.5, 1, 0.5],
3);

            // Create the colour transformation based on
            _colourTransform = new ColorTransform(1, 1, 1, 1, 127, 127, 127);

            // Create the Matrix object
            _matrix = new Matrix(correctedScaleX, 0, 0, correctedScaleY);

        }

        /**
         * Initiates a ripple at a position of the source DisplayObject.
         *
         * @param x The horizontal coordinate of the ripple origin.
         * @param y The vertical coordinate of the ripple origin.
         * @param size The size of the ripple diameter on first impact.
         * @param alpha The alpha value of the ripple on first impact.
         */
        public function drawRipple(x : int, y : int, size : int, alpha : Number) : void
        {
        var half : int = size >> 1;   // We need half the size of the ripple
            var intensity : int = (alpha*0xff & 0xff)*alpha; // The colour which will be
drawn in the currently active buffer

            // calculate and draw the rectangle, having (x, y) in its centre
            _drawRect.x = (-half+x)*_scaleInv;
            _drawRect.y = (-half+y)*_scaleInv;
            _drawRect.width = _drawRect.height = size*_scaleInv;
            _buffer1.fillRect(_drawRect, intensity);
        }

       /**
       * Returns the actual ripple image.
       */
        public function getRippleImage() : BitmapData
        {
        return _defData;
        }

        /**
         * Removes all memory occupied by this instance. This method must be called before
```

```
discarding an instance.
         */
        public function destroy() : void
        {
            _source.removeEventListener(Event.ENTER_FRAME, handleEnterFrame);
            _buffer1.dispose();
            _buffer2.dispose();
            _defData.dispose();
        }

        // the actual loop where the ripples are animated
        private function handleEnterFrame(event : Event) : void
        {
        // a temporary clone of buffer 2
            var temp : BitmapData = _buffer2.clone();
            // buffer2 will contain an expanded version of buffer1
            _buffer2.applyFilter(_buffer1, _fullRect, _origin, _expandFilter);
            // by substracting buffer2's old image, buffer2 will now be a ring
            _buffer2.draw(temp, null, null, BlendMode.SUBTRACT, null, false);
            // scale up and draw to the final displacement map, and apply it to the filter
            _defData.draw(_buffer2, _matrix, _colourTransform, null, null, true);
            _filter.mapBitmap = _defData;
            _source.filters = [_filter];
            temp.dispose();
            // switch buffers 1 and 2
            switchBuffers();
        }

        // switch buffer 1 and 2, so that
        private function switchBuffers() : void
        {
            var temp : BitmapData;
            temp = _buffer1;
            _buffer1 = _buffer2;
            _buffer2 = temp;
        }
    }
}
```

由于百度的不能上传附件。只好把代码硬粘上来了。。如果看着很费劲的话。就复制到开发环境里

看好了。fla 文件代码 可以写如下的

```
stage.addEventListener(MouseEvent.CLICK,onClick);
function onClick(e:MouseEvent):void {
rippler.drawRipple(mouseX,mouseY,15,1);
}
```

一种 AS3 减速运动的算法（有惯性的小球）

作者：粉色男孩

文章来源：http://hi.baidu.com/fsnhf/blog/item/45164e180619b8bc4aedbc18.html

AS3 减速运动在做游戏开发时很重要，今天我们就介绍一种标准的 AS3 减速运动，其实就是高中物理知识。废话少说，先看效果：

下面我们就介绍使用 flashdevelop 来做这个程序，打开 flashdevelop，新建一个 as3 项目（工程－＞新建工程－＞AS3 Project），然后在生成的 Main.as 中输入以下代码：

```
package
{
    import flash.display.Sprite;
    import flash.events.Event;

    /**
    * ...
    * @author Jaja as-max.cn
    */
    public class Main extends Sprite
    {

        public function Main():void
        {
            if (stage) init();
            else addEventListener(Event.ADDED_TO_STAGE, init);
        }

        private function init(e:Event = null):void
        {
            removeEventListener(Event.ADDED_TO_STAGE, init);
            // entry point

            for (var i:int = 0; i < 5; i++) {
                var ball:Ball = new Ball;
                ball.start(int(Math.random() * 60), int(Math.random() * 60));
                addChild(ball);
            }
        }

    }

}
```

这里我们需要一个 Ball 类,在 Main.as 同目录下新建一个 Ball.as 文件,在其中输入以下代码：

```
package
{
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.MouseEvent;

    /**
    * ...
    * @author Jaja as-max.cn
    */
    public class Ball extends Sprite
    {
        /**
        * 开始拖动时的小球 x 坐标
        */
        private var dragStartX:int = 0;

        /**
        * 开始拖动时的小球 y 坐标
        */
        private var dragStartY:int = 0;

        /**
        * 开始拖动时的鼠标 x 坐标
        */
        private var dragStartMouseX:int = 0;

        /**
        * 开始拖动时的鼠标 y 坐标
        */
        private var dragStartMouseY:int = 0;

        /**
        * 上一帧小球 x 坐标
        */
        private var lastX:Number = 0;

        /**
        * 上一帧小球 y 坐标
        */
        private var lastY:Number = 0;

        /**
        * 小球的 x 轴速度，单位（像素/帧）
        */
        private var speedX:Number = 0;

        /**
        * 小球的 y 轴速度
```

```
*/
private var speedY:Number = 0;

/**
* 指定当前是否为正在拖动
*/
private var isDraging:Boolean = false;

/**
* 小球加速度，单位(像素/帧/帧);
*/
private var acceleration:Number = 0.2;

public function Ball() :void
{
    //build ui
    graphics.beginFill(0x999999);
    graphics.lineStyle(2);
    graphics.drawCircle(15, 15, 15);

    this.addEventListener(Event.ADDED_TO_STAGE, addThis);
}

private function addThis(event:Event):void {
    this.addEventListener(Event.REMOVED_FROM_STAGE, removeThis);
    this.addEventListener(MouseEvent.MOUSE_DOWN, downThis);
    stage.addEventListener(MouseEvent.MOUSE_UP, upStage);
}

private function removeThis(event:Event):void {
    this.removeEventListener(Event.REMOVED_FROM_STAGE, removeThis);
    this.removeEventListener(Event.ADDED_TO_STAGE, addThis);
    this.removeEventListener(MouseEvent.MOUSE_DOWN, downThis);
    stage.removeEventListener(MouseEvent.MOUSE_UP, upStage);
    this.removeEventListener(Event.ENTER_FRAME, enterFrame);
}

private function downThis(event:MouseEvent):void {
    dragStartX = this.x;
    dragStartY = this.y;
    dragStartMouseX = this.parent.mouseX;
    dragStartMouseY = this.parent.mouseY;

    isDraging = true;

    start(speedX, speedY);
}

private function enterFrame(event:Event):void {
```

```
    if(isDraging){
        this.x = dragStartX + this.parent.mouseX - dragStartMouseX;
        this.y = dragStartY + this.parent.mouseY - dragStartMouseY;


        speedX = this.x - lastX;
        speedY = this.y - lastY;

        lastX = this.x;
        lastY = this.y;
    }else {
        this.x += speedX;
        this.y += speedY;

        if (afterSpeedX == 0 && afterSpeedY == 0) {
            this.removeEventListener(Event.ENTER_FRAME, enterFrame);
        }

        var afterSpeedX:Number = Math.abs(speedX) - acceleration;
        var afterSpeedY:Number = Math.abs(speedY) - acceleration;



        if (afterSpeedX < 0) {
            afterSpeedX = 0;
        }
        if (afterSpeedY < 0) {
            afterSpeedY = 0;
        }

        speedX = speedX >= 0?afterSpeedX: -afterSpeedX;
        speedY = speedY >= 0?afterSpeedY: -afterSpeedY;
    }

    if (this.x < 0) {
        speedX = Math.abs(speedX);
    }
    if (this.x > stage.stageWidth - this.width) {
        speedX = -Math.abs(speedX);
    }
    if (this.y < 0) {
        speedY = Math.abs(speedY);
    }
    if (this.y > stage.stageHeight - this.height) {
        speedY = -Math.abs(speedY);
    }
}

private function upStage(event:MouseEvent):void {
```

```
            isDraging = false;
        }

        public function start(speedx:int, speedy:int):void {
            speedX = speedx;
            speedY = speedy;

            if(!this.hasEventListener(Event.ENTER_FRAME)){
                this.addEventListener(Event.ENTER_FRAME, enterFrame);
            }
        }

    }

}
```

保存文件，按 F5 或者 Ctrl+Enter 调试我们的程序，就可以看到效果了，可以用鼠标拖动小球试试。当然如果你电脑上没有安装 flashdevelop，用 flash 也可以编译这些文件，将 flash 的场景绑定 Main 类就可以编译了。

ActionScript 3.0 的垃圾回收机制

作者：A 客网

文章来源：http://www.51as.com/as3/ActionScript-3-0delajihuishoujizhi_151.html

众所周知，在 as3 的 flash 运行器中新增了垃圾回收的机制，即自动从内存中清除一些不可访问的对象。这个过程我们是无法控制的，不过可以通过一个例子来观察这个过程：

```
package
{
import flash.display.Sprite;
import flash.text.TextField;
import flash.utils.Timer;
import flash.events.Event;
import flash.events.TimerEvent;
import flash.system.System;
public class GarbageCollection extends Sprite
{
public function GarbageCollection()
{
var s:Sprite = new Sprite;
s.graphics.beginFill(0, 1);
s.graphics.drawRect(0, 0, 100, 100);
//addChild(s);
s.addEventListener(Event.ENTER_FRAME, enterframelistener);
var timer:Timer = new Timer(1);
timer.addEventListener(TimerEvent.TIMER, timelistener);
timer.start();
}
private function timelistener(e:TimerEvent):void
{
new TextField();
}
private function enterframelistener(e:Event):void
{
trace('Flash Player 当前所用内存(字节)：', System.totalMemory);
}
}
}
```

当输出窗口停止输出信息时就意味着在构造方法中创建的局部变量 s 被当成垃圾给回收了（请注意：s 及其在其上侦听的 Event.ENTER_FRAME 一起被清掉了）。

如果将 addChild(s)取消注释，即将 s 放到场景中，它就不会被回收。系统转而回收 timelistener 里创建的 n 多没用的 new TextField，可以看到：

…省略…

Flash Player 当前所用内存： 3403776 字节

Flash Player 当前所用内存： 3407872 字节
Flash Player 当前所用内存： 3420160 字节
Flash Player 当前所用内存： 2142208 字节
Flash Player 当前所用内存： 2146304 字节
…省略…
很明显可以看到垃圾回收的过程。

ps: 此例原型是 Essential ActionScript 3.0 P277 的 Example 14-1. Garbage collection demonstration

Array 和 Object 两种实例化方法效率的比较

作者：A 客网

文章来源：http://www.51as.com/as3/ArrayheObject-liangzhongshilihuafangfaxiaolvdebijiao_197.html

Array 和 Object

其实

```
var list:Array = new Array;
var arg:Object = new Object;
```

比

```
var list:Array = [];
var arg:Object = {};
```

慢

证明：

new Object和{}进行比较

```
var length:int = 1000000;
for(var i:int = 0; i < length; i ++){
   var list:Object = new Object;
}
trace(getTimer());
// trace: 754
var length:int = 1000000;
for(var i:int = 0; i < length; i ++){
   var list:Object = {};
}
trace(getTimer());
// trace: 547
```

new Array 和[]进行比较

```
var length:int = 1000000;

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

   var list:Array = new Array;

}

trace(getTimer());

// trace: 1803

var length:int = 1000000;

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

   var list:Array = [];

}

trace(getTimer());

// trace: 604
```

从这个例子还可以看出实例化 Object 比 Array 快

AS3 hack 技术|垃圾回收

作者：DN_Web

文章来源：http://hi.baidu.com/dn%5Fweb/blog/item/391b2f22aebe8d589922ed70.html

摘抄代码 源自:www.actionscript3.cn/magicianzrh/archives/2007/10/zzas3_hack.html

```
package {
import flash.display.Bitmap;
import flash.display.BitmapData;
import flash.display.Sprite;
import flash.net.LocalConnection;

public class MoonSpirit extends Sprite {
private const SQR_AMOUNT   : int = 10000;    //方块数量
private var _container_sp : Sprite;          //容器 sprite
private var _sqrList    : Array;           //所有方块的引用

public function MoonSpirit() {
   init( );
}

private function init( ) : void{
   _container_sp = new Sprite( );
   addChild(_container_sp);
   //initNoBitmapDataView( );//未强制清理
   initBitmapDataView( );//强制清理
}

//初始化 通过通常手段 显示
private function initNoBitmapDataView( ) : void {
   layoutTenThousandSqr( );
}

//初始化 通过 BitmapData 快照 显示
private function initBitmapDataView( ) : void {
   layoutTenThousandSqr( );
   var myBitmapDataObject : BitmapData = new BitmapData(150, 150, false, 0xFF0000);
   var myImage:Bitmap = new Bitmap(myBitmapDataObject);
   addChild(myImage);
   unLayoutTenThousandSqr( );
   _sqrList = null;
   doClearance( );
}

private function layoutTenThousandSqr( ) : void {
   _sqrList = new Array( );
   for(var i : int = 0; i < SQR_AMOUNT; i++){
```

```
     _sqrList.push(new Sprite());
     _sqrList[i].graphics.beginFill(0xff0000);
     _sqrList[i].graphics.drawRect(0,0,100,100);
     _sqrList[i].graphics.endFill();
     _container_sp.addChild(_sqrList[i]);
    }
}

//不显示
private function unLayoutTenThousandSqr( ) : void {
    for(var i : int = 0; i < SQR_AMOUNT; i++){
     _container_sp.removeChild(_sqrList[i]);
     delete _sqrList[i];
    }
}

//精髓，垃圾回收机强制调用
private function doClearance( ) : void {
    trace("clear");
    try{
     new LocalConnection().connect("foo");
     new LocalConnection().connect("foo");
    }catch(error : Error){
      throw new Error("垃圾");
    }
}
}
}////
```

后来又问了下茄子跟汪汪,他们说 fp 是 catch 一次 Error 就会清理一次,看来 fp 的容错功能还是满强.也避免了因

为错误,而产生的资源浪费.想实验一下,把代码贴进去运行一下看看吧.

AS3 关于 Depth 的问题

作者：A 客网

文章来源：http://www.51as.com/as3/AS3-guanyuDepthdewenti_78.html

As2 里面里有关于深度的函数方法 SwapDepths ,getNextHighestDepth. 但 AS3 里没有这样的函数来处理.所以 AS2 转 AS3 关于深度很多朋友不适应.

AS3 里的深度其实处理起来并不复杂,不过得弄清楚机制.

对于已经在舞台存在的对象，不管你文档原来有多少个层，AS3 只会看对象的数量，每多一个对象，层的序数自动加 1.
比方说，你在Flash里面有 3 个层，第一层（最低的），有 1 个MC，叫AAAA，和一些用绘图工具画的东西，第二层，也有一个MC ，叫BBBB，同样也有一些绘图工具画的东西。第三层，有个MC叫CCCC .好了，那么不管你第一层 用绘图工具画了什么东西，有多少个组，组的层叠情况怎样，统统都只属于一个 Shape 对象，AAAA是一个MC对象，那么第一层就有 2 个对象了，他们分别获得 0 和 1 的序数顺序，第二层也是，无论用绘图工具画了什么也被视作一个Shape对象，还有一个 BBBB的MC对象，他们获得，2 和 3 序数，第三层CCCC获得 4。那么这个文件就一共有 5 个对象。

所谓深度自动分配，就是中间不会出现空的索引，都是连续的序数来的。不象AS2 那样可以随便指定深度。

AS3 要卸载任何显示对象，都可以用removeChild()来从显示列表中删除。但removeChild不会把对象完全销毁。可以用AddChild来重新加载他。

所以实际上 AS3 里已经把深度变成索引(index)的概念. DisplayObject 对象也就是 MovieClip , Sprite 里提供一些函数了,改变或者获取这个索引.

例如:

getChildAt(index:int):DisplayObject
返回位于指定索引处的子显示对象实例。

getChildIndex(child:DisplayObject):int
返回 DisplayObject 的 child 实例的索引位置。

setChildIndex(child:DisplayObject, index:int):void
更改现有子项在显示对象容器中的位置。

swapChildren(child1:DisplayObject, child2:DisplayObject):void
交换两个指定子对象的 Z 轴顺序（从前到后顺序）。

swapChildrenAt(index1:int, index2:int):void
在子级列表中两个指定的索引位置，交换子对象的 Z 轴顺序（前后顺序）。

removeChildAt(index:int):DisplayObject
从 DisplayObjectContainer 的子列表中指定的 index 位置删除子 DisplayObject。

AS3 正则表达式的使用

作者：A 客网

文章来源：http://www.51as.com/as3/AS3-zhengzebiaodashideshiyong_38.html

AS3 中的正则表达式

一、定义方式，可以有两种

```
var pattern1:RegExp = new RegExp("test-\\d", "i");
var pattern2:RegExp = /test-\d/i;
```

1) 使用 new 来新建一个 RegExp 对象，其中参数为 1）表达式字符串 2）表达式的参数，这种方式如果要用\,一定要用\\来转义。

2) 直接采用/形式，以把表达式的内容写到/……/里面，在后面跟上表达式的参数，参数字符可以同时添加多个,例如：/……/gi

二、参数介绍

1)Dotall 属性，用 s 字符表示参数，指定字符（.）在表达式里是不是匹配新行，如果使用了 s 参数，那就表示 dotall 表示真

例：

```
var str:String = "<p>Hello\n"
+ "again</p>"
+ "<p>Hello</p>";
var pattern:RegExp = /<p>.*?<\/p>/;
trace(pattern.dotall) // false
trace(pattern.exec(str)); // <p>Hello</p>
pattern = /<p>.*?<\/p>/s;
trace(pattern.dotall) // true
trace(pattern.exec(str));
```

2) Extended 属性，用 x 参数表示，指是否在表达式定义的时候是否忽略空格

例：

```
var rePhonePattern1:RegExp = /\d{3}-\d{3}-\d{4}|\(\d{3}\)\s?\d{3}-\d{4}/;
var str:String = "The phone number is (415)555-1212.";

trace(rePhonePattern1.extended) // false
trace(rePhonePattern1.exec(str)); // (415)555-1212

var rePhonePattern2:RegExp = / \d{3}-\d{3}-\d{4} | \( \d{3} \) \ ? \d{3}-\d{4} /x;
trace(rePhonePattern2.extended) // true
trace(rePhonePattern2.exec(str)); // (415)555-1212
```

3)global 属性，用 g 参数表示，指是否用表达式在匹配以后在下次匹配的时候是从头再来还是从上次匹配过的地方开始，其 lastIndex 属性会保存起来。

例：

```
var pattern:RegExp = /foo\d/;
var str:String = "foo1 foo2";
trace(pattern.global); // false
trace(pattern.exec(str)); // foo1
trace(pattern.lastIndex); // 0
trace(pattern.exec(str)); // foo1

pattern = /foo\d/g;
trace(pattern.global); // true
trace(pattern.exec(str)); // foo1
trace(pattern.lastIndex); // 4
trace(pattern.exec(str)); // foo2
```

4)ignoreCase 属性，用 i 参数表示，指表达式匹配的时候是否区别大小写。

例：

```
var pattern:RegExp = /bob/;
var str:String = "Bob bob";
trace(pattern.ignoreCase); // false
trace(pattern.exec(str)); // bob

pattern = /bob/i;
trace(pattern.ignoreCase); // true
trace(pattern.exec(str)); // Bob
```

5)lastIndex 属性，指定下次查询的起始位置，这个属性影响两个方法 exec()和 test()，match()，replace()，search()方法是忽略这个属性的，他们总是从头开始的。

这个属性要和 global 结合使用，当 global 为 true 时，执行 exec()和 test()后，lastIndex 属性会被设置为下一个字符，如果是 false，则会从头开始。

例：

6)multiline 属性，用 m 参数表示，指表达式匹配的时候用字符（^）和（$）分别表示在之前或之后有新的一行。

例：

```
var pattern:RegExp = /^bob/;
var str:String = "foo\n"
+ "bob";
trace(pattern.multiline); // false
trace(pattern.exec(str)); // null

pattern = /^bob/m;
trace(pattern.multiline); // true
trace(pattern.exec(str)); // bob
```

7)source 属性，返回表达式的定义字符串。

例：

```
var re1:RegExp = /aabb/gi;
```

```
trace (re1.source); // aabb

var re2:RegExp = new RegExp("x+y*", "i");
trace(re2.source); // x+y*
```

三、方法介绍

1) Exec()方法：

i. 输入：传入一个 String 类型的参数，表示要查询的字符串。

ii. 返回：如果没有匹配到就返回 null，否则返回一个 Object 对象。

这个 Object 对象的属性：

a) 一个 Array（数组），元素 0 包含一个匹配得到的子串，1 到 n 包含，其中定义的组所匹配的字符子串

b) Index 匹配子串在字符串里的位置

c) Input 输入的原始字符串。

例：

```
var myPattern:RegExp = /(\w*)sh(\w*)/ig;
var str:String = "She sells seashells by the seashore";
var result:Object = myPattern.exec(str);
trace(result);
```

输出：

```
* result[0]是"she"
* result[1]是一个空串(第一个\w 是匹配到空的子串)
* result[2]是"e"
* result.index 是 0
* result.input 是" She sells seashells by the seashore"
```

设置了 g(global)属性的例子：

输出：

0 She,,e

10 seashells,sea,ells

27 seashore,sea,ore

2) Test()方法：

i. 输入：传入一个 String 类型的参数，表示要查询的字符串。

ii. 返回：如果匹配返回 true，否则返回 false.

例：

```
var re1:RegExp = /\w/g;
var str:String = "a b c";
trace (re1.lastIndex); // 0
trace (re1.test(str)); // true
trace (re1.lastIndex); // 1
trace (re1.test(str)); // true
trace (re1.lastIndex); // 3
```

```
trace (re1.test(str)); // true
trace (re1.lastIndex); // 5
trace (re1.test(str)); // false
```

四、综合例子：

```
package {

import flash.display.Sprite;

public class RegExpExample extends Sprite {
public function RegExpExample() {
var formalGreeting:String = "Hello, John Smith.";
trace(informalizeGreeting(formalGreeting)); // Hi, John.

var validEmail:String = "name@domain.com";
trace(validateEmail(validEmail)); // true

var invalidEmail:String = "foo";
trace(validateEmail(invalidEmail)); // false

var validPhoneNumber:String = "415-555-1212";
trace(validatePhoneNumber(validPhoneNumber)); // true

var invalidPhoneNumber:String = "312-867-530999";
trace(validatePhoneNumber(invalidPhoneNumber)); // false
}
private function informalizeGreeting(str:String):String {
var pattern:RegExp = new RegExp("hello, (\\w+) \\w+", "i");
return str.replace(pattern, "Hi, $1");
}
private function validateEmail(str:String):Boolean {
var pattern:RegExp = /(\w|[_.\-])+@((\w|-)+\.)+\w{2,4}+/;
var result:Object = pattern.exec(str);
if(result == null) {
return false;
}
return true;
}
private function validatePhoneNumber(str:String):Boolean {
var pattern:RegExp = /^\d{3}-\d{3}-\d{4}$/;
var result:Object = pattern.exec(str);
if(result == null) {
return false;
}
return true;
}
}
```

```
}



var myPattern:RegExp = /(\w*)sh(\w*)/ig;
var str:String = "She sells seashells by the seashore";
var result:Object = myPattern.exec(str);
while (result != null) {
trace ( result.index, "\t", result);
result = myPattern.exec(str);
}
```

AS3 调用GPU渲染

作者：大头

文章来源：http://www.cnblogs.com/cwin5/archive/2009/08/18/1549335.html

内容转自天地会

先帖代码，后讲解：

AS 代码(TestShader.as)：

⊞⊟Code

Pixel Bender 代码(opengltest.pbk)：

⊞⊟Code

OpenGL 的一些知识（针对 FP10 的知识）：

OpenGL 是个专业的 3D 程序接口，是一个功能强大，调用方便的底层 3D 图形库。目前主流的显卡是支持 OpenGL 2.0，而 FP10 支持 OpenGL 2.0。

百度百科资料：[url=http://baike.baidu.com/view/9222.htm]http://baike.baidu.com/view/9222.htm[/url]。

GLSL 是 OpenGL 的开发语言，是用 GPU 运算的。

Pixel Bender 是建立在 GLSL 基础上的语言，其实是 adobe 自己专用的，官方说是有效的在 CPU 或 GPU 上运算(怎么感觉比 GLSL 还智能化，哈哈)。

pbk 是 Pixel Bender Toolkit 开发工具开发 Pixel Bender 的代码源文件的后缀名。

pbj 是 Pixel Bender Toolkit 发布编译后的 Pixel Bender 代码文件的后缀名，给 FP10 调用的文件。

因为说 GLSL 比较顺口，下面都是用 GLSL 表示，不用 Pixel Bender :loveliness:

例子简解：

as 部分：

flash.display.Shader： 着色器，调用 GLSL 就靠她了。

flash.display.Shader： 该类中有 data 属性，data 是 GLSL 代码数据，这个属性就是用于调用 GLSL 中的代码；

_shader.byteCode = byArr： 以字节流获取 GLSL 代码。

_shader.data.src.input = _bitmapData： 这句中的 src 对应 GLSL 中的 input image4 src;，意思是把位图数据传给 GLSL 代码处理。

_shader.data.exposure.value = [0.5-Math.random()]： 这句中的 exposure 是 GLSL 代码中的 exposure 属性，value 是赋值给 GLSL 代码中的 exposure。

GLSL 部分：

<languageVersion：1.0;>： Pixel Bender Toolkit 编译必须的文件头。

kernel ExposureFilter： kernel 基本的图形渲染核心单元，如果不明白可以先理解为 AS 中的 Class，ExposureFilter 渲染单元名，可以先理解为 AS 中的自定义类名，必须的。

void evaluatePixel()： 这个可以理解为主入口运行函数，相当于 C 语言的 main，必须的。

parameter ： 声明给 AS 调用的属性。

input image4 src： GLSL 代码获取 AS 传入的位图数据，src 为自定义属性名，image4 是 src 的类型，input 声明 src 为输入。

output pixel4 dst ： GLSL 代码处理后输出给 flash.display.Shader 着色显示的。dst 为自定义属性名，pixel4 是 dst 的类型，output 声明 dst 为输出。

注意细节：

AS 中的 shader.data.*** 所调用的属性 要与 GLSL 中提供的属性对应；

AS 中给 GLSL 的属性赋值是用[]，而不是直接=，如: _shader.data.exposure.value = [0.5];

具体类型参考 ShaderParameterType 类中的静态属性：

```
package flash.display {
        final public class ShaderParameterType extends Object
        {
            static public var BOOL:String = "bool";
            static public var BOOL2:String = "bool2";
            static public var BOOL3:String = "bool3";
            static public var BOOL4:String = "bool4";
            static public var FLOAT:String = "float";
            static public var FLOAT2:String = "float2";
            static public var FLOAT3:String = "float3";
            static public var FLOAT4:String = "float4";
            static public var INT:String = "int";
            static public var INT2:String = "int2";
            static public var INT3:String = "int3";
            static public var INT4:String = "int4";
            static public var MATRIX2X2:String = "matrix2x2";
            static public var MATRIX3X3:String = "matrix3x3";
            static public var MATRIX4X4:String = "matrix4x4";

            public function ShaderParameterType();
        }
}
```

如果显卡驱动不支持 OpenGL 2.0，程序将使用 CPU 运算，而不是使用 GPU 运算，没有显卡硬件加速的效果。请更新最新显卡驱动，支持 OpenGL 2.0。

相信不久的将来，将会有用 Pixel Bender 开发的提供给 AS 调用的着色渲染引擎。

AS3 动态改变影片帧速

作者：A 客网

文章来源：http://www.51as.com/as3/AS3dongtaigaibianyingpianzhengsu_146.html

在 AS3 中你可以通过 Stage 类动态改变影片的帧速

所有的 DisplayObject 都有一个 stage 属性，通过他可以访问当前 DisplayObject 所在实例的 Stage.

修改 stage 的 frameRate 属性就可以改变帧速，他的值可以是 0.01 ~ 1000

```
package {

import flash.display.Sprite;


public class MySprite extends Sprite

{

public function MySprite () {

// 修改影片帧速到 12 帧 / 秒

stage.frameRate = 12;

}

}

}
```

还记得 as2 中被大量使用的 Delegate 么，as3 中他已经不那么重要了，因为函数可以记住自己的作用域

as2 的例子：

```
import mx.utils.Delegate;

var obj = new Object();

obj.traceThis = traceThis;

obj.traceThis(); // 输出 true，也就是 obj 自己
```

```
obj.traceThis = Delegate.create(this, traceThis);
obj.traceThis(); // 输出 false, 因为我们已经手动把函数作用域指向到了 _root, (as3 中是 Timeline0)

function traceThis () {
trace(this == obj);
}
```

as3 的例子, 在 flash 9 alpha 中测试

```
var obj = new Object();
obj.traceThis = traceThis;
obj.traceThis(); // 输出 false, 没有作用域的问题了~

function traceThis () {
trace(this == obj);
}
```

与 as1, 2 一样, as3 也有一套画图接口 (Drawing API) 使得我们能够动态的使用 as 在 Sprite* 和 MovieClip 中画矢量图。

不过在 as3 中这些方法都没有被定义在 DisplayObject 中 (如 Sprite, MovieClip 等), 而是在他们的 graphics (flash.display.Graphic) 属性中, 这个 graphics 相当于一个动态绘图层。

同时 as3 也提供了一组让我们更方便绘制矩形, 圆, 甚至远郊边框的函数, 他们包含

```
drawCircle(x:Number, y:Number, radius:Number):void
drawEllipse(x:Number, y:Number, width:Number, height:Number):void
drawRect(x:Number, y:Number, width:Number, height:Number):void
drawRoundRect(x:Number, y:Number, width:Number, height:Number, ellipseWidth:Number,
ellipseHeight:Number):void
```

绘制一个蓝色圆角边框：

```
var square:Sprite = new Sprite();
square.graphics.beginFill(0xFF);
square.graphics.drawRoundRect(0, 0, 100, 50, 10, 10);
square.graphics.endFill();
addChild(square);
```

Sprite 是 MovieClip 的简化版本，可以理解成只有一帧的 MovieClip 。

as3 中新增了一些变量类型，基本类型 (顶级类, Top Level Class) 包括简单类型：

Boolean

int

null

Number

String

uint

undefined

复合类型 (或许叫做类会比较顺口)：

Object

Array

Date

Error

Function

RegExp

XML

XMLList

另外还有一些分类到各个包中的诸如 Matrix (flash.geom.Matrix), Shape (flash.display.Shape), URLRequest (flash.net.URLRequest), 等等

一些提示：

Void 类型在 as3 中使用小写, Void -> void

新增了万能类型 * 用来表示所有类型. 如果你没有为变量指定类型, 那么他即是默认类型。

var anyThing:*

XML 类不再是 as1, 2 中的 XML. as3 中的 XML 基于 E4X, 允许你像操作普通变量一样操作 XML (置顶帖中的教程有介绍)

新增 int 和 uint 类型, 他们在数组遍历等不需要小数的场合相当有用. int 类型会比使用 Number 快一点点, 而 uint 通常用来表示颜值等 (据测试 uint 比 Number 慢, int 比 Number 快不了多少)

AS3 接口详解

作者：A 闪

文章来源：http://hi.baidu.com/%B0%B5%BA%DA%B2%E0%CE%C0/blog/item/76861dc8a405db1d7f3e6fbe.html

接口是什么？

这是我们先要提到的一个问题。这个问题直接影响这我们的编程工作，如果你熟悉接口，那么这就会是你的程序锦上添花。笔者对接口这一个概念并不能说是十分的了解，也是在研究中。

在黑羽的殿堂之路中提到，接口就是包含一组方法声明而没有实际代码实现。这是从表面上来看接口，实际上接口还有更重要的东西。在实际的运用中，接口实际上就是将一些没有关联但有着相同方法的类组织起来。这就是接口的重要作用。这样**能够以其他数据类型向上转型，是接口的核心**（黑体字引用殿堂之路第 180 页）。不过，初次接触接口还是不太明白这个是做什么用的。

别急！我们慢慢来！首先，我们要知道，ActionScript3.0 是面向对象编程，也就是说一切皆对象。那么没个对象都应该有它的类型。举个例子！我们自己定义一个 Sprite 对象。

```
var mysprite:Sprite = new Sprite();
```

我们看，在这个语句中。我们声明的时候写到了 mysprite 对象的类型，其实就是冒号后面的 Sprite。那么这个 Sprite 就是类型的声明。下面我们来看一个类文件！

```
package{

        public class foo extends Sprite{

        }

}
```

我们不向类中写任何内容，当我们想生成这个类的一个实例时，我们会这样写：

```
var mysprite:foo = new foo();
```

我们看，此时我们的类型声明就改变了！而是编程这个子类的名称。好了！那么这个和接口有什么关系呢？实际上，接口就是定义另外一个类型。让一个类同时拥有多个类型。这种效果类似与多继承，但并非是多继承。因为此时的接口只是一个类型声明，同时又有一个或多个方法声明。我们来看一个实例！

```
package{
import flash.display.Sprite;
public class Main extends Sprite{
   public function Main():void{
   var ftt:IALL = new foo();
    trace(ftt.clone());
   }
}
}
```

```
interface IALL{
function clone():IALL;
}
import flash.display.Sprite;
class foo extends Sprite implements IALL{
public function clone():IALL{
   return new foo;
}
}
```

我们来看这段代码。不难发现，这里有一句话值得我们注意！就是红色的这句。你会发现有什么地方不一样呢？

可以看出，这个 ftt 对象实际上应该是 foo 类的一个实例，然而在定义的时候，我们写的数据类型却是 IALL！这是为什么呢？其实这就是接口在 AS3 中的作用。我们可以把接口看作是一种抽象的数据类型。依靠这种抽象的数据类型，我们可以把多个不同且无关联的类结合到一起。利用这个特点我们可以实现向上反转和向下反转（关于向上反转和向下反转将在以后介绍）！

下面我们来看看接口的标准写法：

```
package 包路径{

	public interface 接口名称{

		function 方法名(参数:参数类型):返回类型;

		static function 方法名(参数:参数类型):返回类型;

		function get 方法名():返回类型;

		function set 方法名(参数:参数类型):void;

	}

}
```

0 了！~今天就先写到这里吧！以后再想起什么再补充！

AS3 里的工厂模式和模板方法模式

作者：A 客网

文章来源：http://www.51as.com/as3/AS3lidegongchangmoshihemobanfangfamoshi_103.html

抽象类(Abstract Classes)

抽象类在工厂和模板方法模式里扮演着重要角色。虽然 ActionScript 3.0 并没有原生支持它们，但我们依然可以在 ActipScript 3.0 里套用抽象类和抽象方法的思想。一个抽象类就是一个总被用来继承且永不会直接被实例化的类。它的用途跟接口类相似，但有一个最大的不同之处就是：接口类只定义公有方法的名称而没有具体的执行（就像是只有函数名而没有函数体），但抽象类两者都有。

抽象类使用抽象方法，这些抽象方法没有具体的执行，只用作占位符。在其它语言，例如 C#、Java，你可以使用 abstract 关键字来声明抽象方法，它的子类必须重写这些抽象方法。但在 ActionScript 3.0 里并没有 abstract 关键字，所以我们得想办法解决此问题。在 ActionScript 3.0 里使用抽象类，你必须知道两个特别的关键字：override 和 final。子类必须使用 override 关键字重写抽象类的方法，方法名称必须相同。使用 final 关键字来声明那些不被子类重写的方法，在模板方法模式里就用到 final 关键字了。

模板方法

定义在抽象类里的模板方法包含一些方案，这些方案来用组织抽象方法。请看以下这个抽象类，在 initialize()方法里，我们定义 games 初始化的方式：

```
1. package com.peachpit.aas3wdp.factoryexample {
2.
3. public class AbstractGame {
4.
5. // 模板函数
6. public final function initialize():void {
7. createField();
8. createTeam("red");
9. createTeam("blue");
10. startGame();
11. }
12.
13. public function createField():void {
14. throw new Error("Abstract Method!");
15. }
16.
17. public function createTeam(name:String):void {
18. throw new Error("Abstract Method!");
19. }
20.
21. public function startGame():void {
22. throw new Error("Abstract Method!");
23. }
24. }
25.
26. }
```

在上面例子里，initialize() 方法是一个模板函数。它定义了 game 是这样初始化的：首先调用 createField() 方法，然后调用 createTeam() 方法来创建队伍（teams），最后调用 startGame() 方法。然而在这个抽象类里面，这几个函数都没干什么实际的事情。它们是给子类重写的，让子类来决定具体需要做的事情。抽象类都不能被直接实例化，而且抽象类的方法都必须被重写，所以在上面那三个方法里都放置了 throw 语句，这样就可以防止它们被直接调用了。
接下来我们创建一个 FootballGame 类来继承 AbstractGame 类。这个子类重写抽象类的方法，然后在 initialize() 方法里被调。

```
1. package com.peachpit.aas3wdp.factoryexample {
2.
3. public class FootballGame extends AbstractGame {
4.
5. public override function createField():void {
6. trace("Create Football Field");
7. }
8.
9. public override function createTeam(name:String):void {
10. trace("Create Football Team Named " + name);
11. }
12.
13. public override function startGame():void {
14. trace("Start Football Game");
15. }
16.
17. }
18.
19. }
```

正如你所看见的，FootballGame 类重写了 createField()、createTeam()以及 startGame() 方法，使这些方法更适合 football。然而那个 initialize() 方法没有改变。你可以通过这种方式来写一个 BaseballGame 和 BastketballGame 类。可以像下面代码那样使用 FootballGame 类：

```
1. package com.peachpit.aas3wdp.factoryexample {
2.
3. import com.peachpit.aas3wdp.factoryexample.FootballGame;
4. import flash.display.Sprite;
5.
6. public class FactoryExample extends Sprite {
7.
8. public function FactoryExample() {
9. // Create an instance of FootballGame
10. var game:FootballGame = new FootballGame();
11. // Call the template method defined in AbstractGame
12. game.initialize();
13. }
14.
15. }
16.
```

```
17. }
```

执行上面的代码输出如下。你可以看到，重写后的方法会在那个模板函数（initialize()）里调用，这个模板函数没有改变。

1. Create Football Field
2. Create Football Team Named red
3. Create Football Team Named blue
4. Start Football Game

也许你会疑惑抽象类有什么用，现在你应该明白了一些，这样做的好处，很显然，扩展性强，以后 createField()、createTeam()以及 startGame() 内容需要更改，我们只要再创建一个子类就可以，不必修改其他应用代码。

工厂方法

现在我们拿上文所举的模板方法例子（点击这里）来说一下工厂方法。很普遍的做法是在模板方法里应用工厂方法。

在上文所说的例子里，createField() 方法并没有返回任何东西；它只输出“Create Football Field”。让我们来改一下它，让它可以创建并返回一个 field 对象。因为不同的游戏有不同的 field 类型，所以我们将要创建一个叫 IField 的接口类，所以 field 类都会实现这个接口。以下这个接口类定义一个 drawField() 方法：

```
1. package com.peachpit.aas3wdp.factory {
2.
3. public interface IField {
4.
5. function drawField():void;
6.
7. }
8.
9. }
```

接下来我们新建一个 FootballField 类来实现 IField 接口。FootballField 类定义如下：

```
1. package com.peachpit.aas3wdp.factoryexample {
2.
3. import com.peachpit.aas3wdp.factoryexample.IField;
4.
5. public class FootballField implements IField {
6.
7. public function drawField():void {
8. trace("Drawing the Football Field");
9. }
10.
11. }
```

```
12.
13. }
```

工厂方法的目的是连接两个或以上分离但相关的类层次结构。在这个例子里，最顶层的是 AbstractGame 类，它有这些子类：FootballGame、BaseballGamet 和 BastketballGame 类。然后第二层的是 IField 接口类，这些类实现这个接口：FootballField、BaseballField 和 BasketballField 类。这个 abstractGame 和 IField 对象是相关的，但它们具体的实例化视它们的子类而定。下图表示了这些类的层次结构：

05fig01_cs_alt.png

注意下面的 createField() 和 initialize() 方法，createField() 方法是一个工厂方法，它返回一个实现 IField 接口的对象。请看：

```
1. package com.peachpit.aas3wdp.factoryexample {
2.
3. import com.peachpit.aas3wdp.factoryexample.IField;
4.
5. public class AbstractGame {
6.
7. // 模板方法
8. public final function initialize():void {
9. var field:IField = createField();
10. field.drawField();
11. createTeam("red");
12. createTeam("blue");
13. startGame();
14. }
15.
16. // 工厂方法
17. public function createField():IField{
18. throw new Error("Abstract Method!");
19. }
20.
21. public function createTeam(name:String):void {
22. throw new Error("Abstract Method!");
23. }
24.
25. public function startGame():void {
26. throw new Error("Abstract Method!");
27. }
28. }
29.
30. }
```

这个抽象类和里面的模板方法都是匿名的，具体需要执行的事情在子类里定义。请看以下的 FootballField 类：

```
1. package com.peachpit.aas3wdp.factory {
2.
3. import com.peachpit.aas3wdp.factory.FootballField;
4. import com.peachpit.aas3wdp.factory.IField;
5.
6. public class FootballGame extends AbstractGame {
7.
8. public override function createField():IField {
9. return new FootballField();
10. }
11.
12. public override function createTeam(name:String):void {
13. trace("Create Football Team Named " + name);
14. }
15.
16. public override function startGame():void {
17. trace("Start Football Game");
18. }
19.
20. }
21.
22. }
```

运行以上例子，会输出如下：

1. Drawing the Football Field
2. Create Football Team Named red
3. Create Football Team Named blue
4. Start Football Game

简单工厂

请注意，厂模式和工厂方法模式不同，以下这个为简单工厂模式：

```
1. package com.peachpit.aas3wdp.factoryexample {
2.
3. public class GameFactory {
4. public static function createGame(gameType:String):IGame {
5. switch(gameType){
6. case "football":
7. return new FootballGame();
8. case "baseball":
9. return new BaseballGame();
10. case "basketball":
11. default:
12. return new BasketballGame();
13. }
```

```
14. }
15. }
16. }
```

小明：很认真写blog的ASer ,大家去Y.boy的blog留言支持下! http://www.riahome.cn

AS3 中加载的机制(loading)

作者：A 客网

文章来源：http://www.51as.com/as3/AS3zhongjiazaidejizhi-loading-_97.html

摸了好一阵子,才弄明白 AS3.0 的加载机制.

还是坚持自己的原则,从适用的角度做记录!下面分别讲述 AS3 各加载事件与类!

AS3Loader.rar]相关例子下载(内带 swf 与 jpg 加载例子及舞台加载条和 CS3 的加载条应用类)

带 swf 预览

一: Loader 类

在 AS3.0 里把所有事件,属性,加载等都集中在某个对象上了.且加载对象与各触发事件也进行了分工,这和 2.0 时期,用 onEnterFrame 和不断检测加载百分比强多了!

Loader 继承了基类 DisplayObjectContainer,所以他可以也必须当作一个对象用 addChild 添加才能工作.

Loader 类可用于加载 SWF 文件或图像（JPG、PNG 或 静态 GIF）文件。 使用 load() 方法来启动加载。 被加载的显示对象将作为 Loader 对象的子级添加。

例:

```
var loadimg: Loader = new Loader();
var url:String = "http://www.shch8.com/v2007/up/UploadFile/200769182617-1.gif"
var urlReq:URLRequest = new URLRequest();
urlReq.url=url;
loadimg.load(urlReq);
addChild(loadimg);
```

和 2.0 和比,还有一点区别,这里加载时要先把字符串地址转化为 url 加载对象,在程序中的第三行是直接设置加载对象的 url 属性的,也可以这样写 urlReq = new URLRequest(url),除了 url 属性还有几个公共属性，一般很少用到如：method 用来控制 get 还是 post 提交方式。

因为把加载当作一个对象了,所以就不需要象 2.0 时期一样,新建一个影片来装载加对的物体 loadMovie("myimg.jpg","mv")，我们 可以直接设置他的 x/y 轴或宽高。还有 Loader 对象是二进制方式加载 swf 了，在 flash9 之前我们做加载条是用影片的 getBytesLoaded 和 getBytesTotal 来检测的是否加载完成。这不是真正意义上的加载，只是判断帧的加载数，所以会出现类似情况，加载到 20%停了很久突然跳到 90%因为那一帧放了整个影片 70%的数据。以前在蓝色里有讨论过这个话题，还有人说是 MM 在走江湖！呵，现在解决了！

Loader 的所有方法：

1. Loader()

创建一个可用于加载文件（如 SWF、JPEG、GIF 或 PNG 文件）的 Loader 对象。

2. close(): void

取消当前正在对 Loader 实例执行的 load() 方法操作。

3.load(request: URLRequest, context: LoaderContext = null):void

将 SWF、JPEG、渐进式 JPEG、非动画 GIF 或 PNG 文件加载到此 Loader 对象的子对象中。

4.loadBytes(bytes:ByteArray, context: LoaderContext = null):void

从 ByteArray 对象中所存储的二进制数据中加载。

5.unload():void

删除此 Loader 对象中使用 load() 方法加载的子项。

二: LoaderInfo 事件机制

LoaderInfo 非常好用，他是继承 EventDispatcher 对象用来检测网络加载状态。可以把加载动作细细地解剖出来。

原来在 flash9 之前，我们绞尽脑汁去获取加载来的 swf 的宽度、高度、帧频、版本等数据，但一直没研究出好的方法现在 LoaderInfo 可以做到能获取加载对象的各属性，这点很好有时用 swf 来加载不确定 swf 时很有用，可以用那些属性来重新设置主 swf。

加载对象所加载数据的实时检测上，PROGRESS 事件可以取代以前用 onEnterFrame 的疯狂检测工作。当然 2.0 的也有自己的事件，

只是很少人用！

LoaderInfo 所继承的所有事件：

1.complete(事件参数 Event. COMPLETE)

成功加载数据后调度。

2.HttpStatus(事件参数 HTTPStatusEvent.HTTP_STATUS)

在通过 HTTP 发出网络请求并且 Flash Player 可以检测到 HTTP 状态代码时调度。

3. Init(事件参数 Event.INIT)

已加载的 SWF 文件的属性和方法可访问时调度。

4.IoError(事件参数 IOErrorEvent.IO_ERROR)

在发生导致加载操作失败的输入或输出错误时调度。

5.Open(事件参数 Event.OPEN)

在加载操作开始时调度。

6.Progress(事件参数 ProgressEvent.PROGRESS)

在下载操作过程中收到数据时调度。

7.Unload(事件参数 Event.UNLOAD)

每次使用 Loader 对象的 unload() 方法删除已加载对象时，或者当同一 Loader 对象执行第二次加载并且在加载开始之前删除了原始内容时，由 对象调度。

具体测试例子请看 loadjpg.swf 与 loadswf.swf

LoaderInfo 的获取 swf 属性时要等 swf 加载完才能获取，也就是在 COMPLETE 事件里获取

如:

```
loadswf.contentLoaderInfo.addEventListener(Event.COMPLETE, loadcom)
function loadcom(the:Event):void {
trace("AS 版本:AS"+the.target.actionScriptVersion+".0")
trace("swf 版本:flash"+the.target.swfVersion+".0")
trace("swf 宽:"+the.target.width+"swf 高:"+the.target.height)
trace("swf 帧频:"+the.target.frameRate+"帧/秒")
}
```

事件触发的各个顺序分别是

OPEN>>INIT>>HTTP>> COMPLETE

加载操作开始时调度>>进入事件 SWF 文件的属性和方法调度>>状态事件通过 HTTP 发出网络请求并且 Flash Player 检测到 HTTP 状态代码>>加载完成

例(需要 flashPlay9.0 播放器才能正常测试):

三：跨域加载的安全机制

你可以加载来自任何可访问源的内容。

如果执行调用的 SWF 文件位于网络沙箱中并且要加载的文件是本地的，则不允许加载。

如果加载的内容为用 ActionScript 3.0 编写的 SWF 文件，那么除非可以通过调用加载的内容文件中的 System.allowDomain() 或 System.allowInsecureDomain() 方法来允许跨脚本排列，否则另一个安全沙箱中的 SWF 文件不能对它执行跨脚本操作。

如果被加载的内容为 AVM1 SWF 文件（用 ActionScript 1.0 或 2.0 编写），则 AVM2 SWF 文件（用 ActionScript 3.0 编写）不能对它执行跨脚本操作。 但是，可以通过使用 LocalConnection 类在两个 SWF 文件之间实现通信。

如果被加载的内容为图像，则除非该 SWF 文件的域包含在该图像原始域的跨域策略文件中，否则安全沙箱之外的 SWF 文件无法访问其数据。

在只能与本地文件系统的内容交互的沙箱中的影片剪辑不能对只能与远程内容交互的沙箱中的影片剪辑使用脚本，反之亦然。

四：主场景加载条制作

我们都知道，AS2.0 是用_root. getBytesLoaded()和_root. getBytesTotal()来判断 swf 是否被加载完，但在 3.0 里面_root，_global，_parent 等原来的"骨干职工"都被开除了！ 取代他的是 stage，对于场影设置，他是一手遮天了如设置 swf 全屏，对齐方式，显视品质等。但 stage 只继承了 DisplayObjectContainer 一部分属性，可用的还太少了。

在做场影加载条时，我们要获取场影的加载情况。要想办法把上面讲的 LoadInfo 事件添加到主场景去，用 stage 是做不到的。但可以用显视类 DisplayObject 添加，他继承了 EventDispatcher，可以直接用他的属性指定主场景来添加，DisplayObject 和属性和原来的 movieClip 还是很像的，只是前 面不用加下划线作区分了如：root.loaderInfo.addEventListener()
用 root 调度 loaderInfo：

```
root.loaderInfo.addEventListener(ProgressEvent.PROGRESS, loadshow)
function loadshow(the: ProgressEvent):void {
var loadnum:Number=int(the.bytesLoaded/the.bytesTotal*100);
showtxt.text=”load:”+loadnum + “%”;
gotoAndStop(loadnum);
}
```

五：使用系统图片与加载条等组件写的加载类

```
package myAs{
import fl.containers.UILoader;//图片加载组件
import fl.controls.Label;//文本组件
import fl.controls.ProgressBar;//进度条
import flash.events.Event;//输入事件类,如果代码写在帧上,可不用输入
import flash.events.ProgressEvent;//输入事，件类
import flash.text.TextField;//引进文本类
import flash.net.URLRequest;
import fl.controls.ProgressBarMode;
import flash.display.Sprite;
import flash.text.TextFieldAutoSize;//调整类
public class loadimg extends Sprite {
private var url:String = “http://image.cnool.net/picn/2005/ompic030b.jpg”
private var loadbox: ProgressBar = new ProgressBar();
private var imgbox: UILoader = new UILoader();
private var showtxt: Label = new Label();
private var titleTxt:TextField=new TextField();
//注意:上面的几个对象要声明在这里,不能放在主函数里声明,这样 loadeven()等事件函数里才能认到
public function loadimg() {
titleTxt.htmlText=”图片加载示例:”;
addChild(titleTxt);
titleTxt.x=0;
titleTxt.y=10;
showtxt.autoSize = TextFieldAutoSize.LEFT;
showtxt.text = “”;
showtxt.move(150, 10);
addChild(showtxt);
loadbox.mode = ProgressBarMode.MANUAL;
loadbox.move(150, 30);
addChild(loadbox);
imgbox.load(new URLRequest(url));
imgbox.addEventListener(ProgressEvent.PROGRESS, loadeven);
imgbox.addEventListener(Event.COMPLETE, loadend);
```

```
imgbox.setSize(550,400);
imgbox.move(0, 40);
addChild(imgbox);
}
private function loadeven(event: ProgressEvent):void {//事件:加载进度显视
var uiLdr:UILoader = event.currentTarget as UILoader;
var kbLoaded:String = Number(uiLdr.bytesLoaded / 1024).toFixed(1);
var kbTotal:String = Number(uiLdr.bytesTotal / 1024).toFixed(1);
showtxt.text = kbLoaded + "/" + kbTotal + " KB" + " (load:" + Math.round(uiLdr.percentLoaded) + "%)";
loadbox.setProgress(event.bytesLoaded, event.bytesTotal);
}
private function loadend(event: Event):void {//事件:加载完成 删除事件
//showtxt.visible = false;
//loadbox.visible = false;//加载完成后隐藏进度条
imgbox.removeEventListener(ProgressEvent.PROGRESS, loadeven);
imgbox.removeEventListener(Event.COMPLETE, loadend);
}
}
}
```

使用组件就轻松多了,不用考虑那么多。只是开发出一个适用的产品最好不要去用官方的组件。自己去写过，写过适合自己的组件。不过 CS3 的 Uiloader 感觉很不错！

as3 中一些不太常见的代码写法

作者：zoujun1314

文章来源：http://hi.baidu.com/zoujun1314/blog/item/331520dedcfe2d51ccbf1ac7.html

1、

```
mc.gotoAndPlay(Math.random()*10>>0);
```

Math.random()*10 取 10 以内带小数的随机数。
>> 是移位符号 >>0 的功能去除小数点后的数！！（这个方法的确高明）
所以 Math.random()*10>>0 整句话的意思是 获取 10 以内的随机整数！

推广的想
用在加载进度条中

```
trace((已经加载的字节数/总字节数}*100>>0+"%")
```

的确很方便啊
2、

```
if(i&1){}
```

i&1 的意思就是判断 i 是奇数还是偶数
如果是奇数 i 的 2 进制表示最后一位是 1, i&1=1 真
偶数 i 的 2 进制表示最后一位是 0, i&1=0 假。
3、

```
var a:Array = new Array();
for (var i:int=0; i<10; a.push(i++)){
//....
}
```

此循环 执行后 a 数组的值为 0,1,2,3,4,5,6,7,8,9.
一个简化程序的方法
能用上的地方有很多。。具体地方具体分析。。
4、

```
var temp:Number=mouseX;
stage.addEventListener(MouseEvent.MOUSE_MOVE,moveHandler);
function moveHandler(e:MouseEvent):void {
var d:Number=(-temp+(temp=mouseX));
}
```

这段代码功能是 获取鼠标移动距离 d
用在鼠标拖拽窗体 或者鼠标拖拽游戏上 真是不要太爽！！！
代码完全可以改为

```
Mouse.hide();
var temp:Number=mouseX;
stage.addEventListener(MouseEvent.MOUSE_MOVE,moveHandler);
function moveHandler(e:MouseEvent):void {
var d:Number=(-temp+(temp=mouseX));
spriteMouse.x += d;
e.updateAfterEvent();
}
```

这样 spriteMouse 这个影片剪辑就成了一个鼠标
5、

```
function fn1 () {
trace("1");
}
function fn2 () {
trace("2");
}
([fn1,fn2][Math.random()*2>>0])();
```

这个是随机执行函数的命令 挺有意思的

如 一个欢迎界面

```
function fn1 () {
trace("你来啦欢迎啊");
}
function fn2 () {
trace("来了你别走啊 ");
}
function fn3 () {
trace("你这么才来啊");
}
function fn4 () {
trace("就猜到你要来");
}
([fn1,fn2,fn3,fn4][Math.random()*4>>0])();
```

6、

```
var a:int=3;
if (a>0&&(trace("a is positive number"),a&1))
trace("a is positive odd number");
```

先执行 a>0 真

再执行 trace("a is positive number")

最后判断 a 是不是奇数

这里的逗号表达式是从左向右执行的。

此用法主要 用来判断有没有执行某处语句 吧

7、

```
var a:int=13;
var b:int=-12;
if((a^b)>0||(b*=-1,a>10))
trace(b);
```

又是一个逗号表达式

a^b = -7

b*=-1 b=12

a>10

输出 b=12

Fanction 指定 thisObj

作者：Lite

文章来源：http://www.lite3.cn/blog/?p=112

函数里可以使用 **this** 关键字,可以很方便的引用一个对象

一般函数的 this 可以通过 **function.apply** 或 **function.call** 来修改 **this** 指向的

不过 在作为类的固有方法,则不能改变 **this** 的指向

View Code ACTIONSCRIPT3

```
package
{
	import flash.display.Sprite;
	/**
	 * 欢迎访问我的博客
	 * www.lite3.cn
	 * lite3@qq.com
	 * @author lite3
	 */
	public class FunctionTest extends Sprite
	{
		public function FunctionTest()
		{
			// 实例化一个对象，把这个对象分别传递给他们俩作为 thisObj
			var obj:Object = { };

			trace("前者是真正的 thisObj， 后者是传递的 thisObj");

			// 不能改变 thisObj
			test.call(this, this);
			test.apply(this, [this]);
			test.call(obj, obj);
			test.apply(obj, [obj]);
			test.call(null, null);
			test.apply(null, [null]);
```

```
                    // 可以改变 thisObj
                    foo.call(this, this);
                    foo.apply(this, [this]);
                    foo.call(obj, obj);
                    foo.apply(obj, [obj]);
                    foo.call(null, null);
                    foo.apply(null, [null]);

                    // 可以改变 this
                    function foo(_this:*):void
                    {
                          trace("function  foo! ", this, "\t\t", _this);
                    }
              }

              // 不能改变 this this 始终指向 FunctionTest 实例
              private function test(_this:*):void
              {
                    trace("function test! ", this, "\t\t", _this);
              }
        }
}
```

运行结果为:

[Copy to clipboard]View Code TEXT

```
前者是真正的 thisObj， 后者是传递的 thisObj
function test!  [object FunctionTest]         [object FunctionTest]
function test!  [object FunctionTest]         [object FunctionTest]
function test!  [object FunctionTest]         [object Object]
function test!  [object FunctionTest]         [object Object]
function test!  [object FunctionTest]         null
function test!  [object FunctionTest]         null
function  foo!  [object FunctionTest]         [object FunctionTest]
function  foo!  [object FunctionTest]         [object FunctionTest]
```

```
function  foo!  [object Object]         [object Object]
function  foo!  [object Object]         [object Object]
function  foo!  [object global]         null
function  foo!  [object global]         null
```

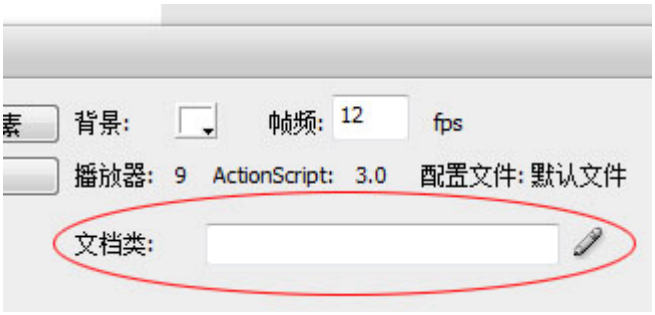
Flash CS3 直接编译文档类的方法

作者：小 S

文章来源：http://www.xiaos8.com/article.asp?id=53

新建一个 fla 文件，在属性面板中，有下面这个文档类填写部分，填写类路径，即可编译成功

图片名称：**出错图片**



图片地址：**uploads/200801/24_180517_1.jpg**

这样直接编译文档类需要注意的是，该类必须继承 Sprite 类，否则会报错！

FLASH 调用上级文件夹中的文件

作者：A 闪

文章来源：http://hi.baidu.com/%B0%B5%BA%DA%B2%E0%CE%C0/blog/item/d59f193931c815f93b87ce44.html

今天在 QQ 群讨论的时候有人问道 FLASH 调用上一级文件夹中的文件。一开始觉得不可能！因为平常我们只是调用同意文件夹中的文件，很少涉及到调用上一级中的文件。不过，群里高手很多，突然说到了一种和 C 语言中的一个相同的用法（AS 真是一个大杂烩）。这个功能还真是能够实现的。

假如说现在我们有一个名称为 a.jpg 的图片，然后在同一个目录下又有一个文件夹，在这个文件夹中我们放入了一个 swf 文件，这时，我们想让 swf 文件去加载这个 a.jpg 图片，此时就是一个调用上一级文件夹中的文件。实现方法也很简单，这里我使用的是 AS3 语言！代码如下：

```
var my:Loader = new Loader();
my.load(new URLRequest("../a.jpg"));
addChild(my);
```

这段代码中最重要的就是红色的文字了！这里就正确的指出了文件的路径，其实上一级就是“../”这样写就可以了！如果是上两级就要写成“../../”。道理相同，一次类推！非常简单！希望对大家有用！

关于构造函数里的stage操作

作者：大头

文章来源：http://www.cnblogs.com/cwin5/archive/2009/02/23/1396720.html

如果一个类不是文档类的话..那么在它的构造函数里操作 stage 的话就会出现空对象..

而解决的办法可以用:AddedToStage 事件侦听...

如:

```
package
{
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.MouseEvent;

    /**
     * ...
     * @author cwin5
     */
    public class test extends Sprite
    {
        public function test() {
            this.addEventListener(Event.ADDED_TO_STAGE, onAddedToStage);
        }

        private function onAddedToStage(e:Event):void
        {
            stage.addEventListener(MouseEvent.CLICK, onClick);
        }

        private function onClick(e:MouseEvent):void
        {
            trace("click");
        }
    }
```

}

个人认为..文档类初始化时先初始化其它实例...再把本身加入舞台,,然后再执行构造函数，而其由于实例的父级.即文档类未加入舞台..对 stage 操作则为空..自然会出错..

精确的碰撞检测

作者：粉色男孩

文章来源：http://hi.baidu.com/fsnhf/blog/item/c6fbc2d18d3549da562c8463.html

在做游戏时，我们会用到碰撞检测，可是不管是 hitTestPoint 还是 hitTestObject 都不能满足我们的使用，因为我们要进行碰撞检测的两个物体可能都是不规则的。

因为我们要对显示对象进行像素级检测，所以我们要把图像转成位图，然后进行检测，原理就是这样，很简单，下面我们来实现。

首先看一下效果：http://remotedu.net/blog/article/51.htm

为了方便使用，我们还是先写出自己的碰撞物体类：HSprite

```
package cn.asmax.display
{
import cn.asmax.display.interfaces.IHitTestable;
import flash.display.BitmapData;
import flash.display.DisplayObject;
import flash.display.Sprite;
import flash.geom.Point;

/**
* ...
* @author Jaja
*/
public class HSprite extends Sprite
{
public function HSprite()
{

}

private var point1:Point;
private var point2:Point;
private var bitmapData1:BitmapData;
private var bitmapData2:BitmapData;

/**
* 对显示对象执行像素级精确碰撞检测
* @param    hitArea    检测碰撞的显示对象
* @return
*/
public function hitTest(hitArea:DisplayObject):Boolean {
if (bitmapData1 != null) {
bitmapData1.dispose();
}
```

```
if (bitmapData2 != null) {
bitmapData2.dispose();
}

bitmapData1 = new BitmapData(this.width, this.height, true, 0x00000000);
bitmapData1.draw(this);
bitmapData2 = new BitmapData(hitArea.width, hitArea.height, true, 0x00000000);
bitmapData2.draw(hitArea);

if (point1 == null) {
point1 = new Point(this.x, this.y);
}
point1.x = this.x;
point1.y = this.y;
if (point2 == null) {
point2 = new Point(hitArea.x, hitArea.y);
}
point2.x = hitArea.x;
point2.y = hitArea.y;

return bitmapData1.hitTest(point1, 1, bitmapData2, point2, 1);
}

}

}
```

然后我们打开Flash，做几个MovieClip，将它们所绑定的类都扩展HSprite类，这样我们的MovieClip就拥有了像素级检测的功能了，关于详细的实现代码，您可以下载源文件看看。

来一次简单截图学习—BitmapData.draw()方法

作者：小 S

文章来源：http://www.xiaos8.com/article.asp?id=35#comment3383

```
package index.base.program{

  import flash.geom.Matrix;
  import flash.geom.Rectangle;
  import flash.display.BitmapData
  import flash.display.DisplayObject;

  public function photography(photo:DisplayObject,transparent:Boolean = true,fillColor:uint = 0):
BitmapData{
    var tmpRect:Rectangle = photo.getRect(photo);
    var picture:BitmapData = new BitmapData(photo.width,photo.height,transparent,fillColor);
    picture.draw(photo,new Matrix(1,0,0,1,- tmpRect.left,- tmpRect.top));
    return picture;
  }
}
```

首先共享一个函数给大家，这个函数干的事情就是在 DisplayObject 中自动找出存在的图像区域，并且进行截图

然后讲解一下这个函数
首先取出 photo 照片的图像区域的矩形区域
然后创建 BitmapData 对象，前两个参数是创建的 BD(BitmapData)图像的宽高，后两个分别是是否透明和填充颜色
（填充颜色拿出来讲一下，在 flash 中标准的颜色表示方法是应该有 8 位的即 0x11223344，11 的位置指的是透明度，223344 分别是 rgb，因此虽然设置第三个参数是透明，但是使用 draw 方法时，填充颜色默认属性是 0xffffffff，即表示用纯白色填充，因此一样是不透明）
创建了 BitmapData 之后，我们可以使用 draw 方法了，draw 有很多属性可以设置，这儿就不多说了，简单截图学习嘛。。先学会截图再说

然后函数返回 picture 对象，然后外面用

```
var bmp:Bitmap = new Bitmap(photography(mc));
addChild(bmp);
```

这样我们就创建了一张简单的照片了~~并且学会了简单的截图功能~

扩展一下这个函数，可以设置出血现，自行研究多出来的怎么使用吧 [lol]

```
package index.base.program{

  import flash.geom.Matrix;
  import flash.geom.Rectangle;
  import flash.display.BitmapData
  import flash.display.DisplayObject;

  public function photography(photo:DisplayObject,fillColor:uint = 0,hemorrhage:Number = 0,transp
```

```
arent:Boolean = true):BitmapData{
    var tmpRect:Rectangle = photo.getRect(photo);
    var picture:BitmapData = new BitmapData(photo.width + hemorrhage * 2,photo.height + hemorrhage
 * 2,transparent,fillColor);
    picture.draw(photo,new Matrix(1,0,0,1,- tmpRect.left + hemorrhage,- tmpRect.top + hemorrhag
e));
    return picture;
  }
}
```

利用mouseChildren做封印容器

作者：小 S
文章来源：http://www.xiaos8.com/article.asp?id=16

做虚拟社区的时候，虚拟形象，要响应的一些鼠标事件等，或许你可以采用给所有的虚拟形象单独加侦听事件，那这样用户一多，相应的事件也多了，不但消耗了资源，管理事件也难多了。
如果采用对虚拟人物存在容器进行统一鼠标事件侦听，那样就方便多了，只要触发了某个鼠标事件，找出他的目标(target)，那么就可以对他进行相应的操作
不但管理方便，做起来也非常方便。

但是虚拟形象佩戴装备，或者制作素材的美工，在里面做了好几个 MC，或者其他的，有可能找到的目标不是真正虚拟形象，而是虚拟形象的一部分，那么就得不到自己想要的东西了
因此，这样就需要把整个虚拟形象类给封印起来，让找目标(target)的时候，无论怎么找都是虚拟形象。

先来看这个简单的封印容器类

```
package base.org{

  import flash.display.Sprite;

  public class Seal extends Sprite{

    public function Seal(){
      mouseChildren = false;
    }

  }

}
```

从上面可以看到，只是在构造函数中加了 mouseChildren 属性等于 false，现在我们来测试写了这个属性有什么不同

```
import base.org.Seal;

var s:Seal = new Seal;
s.addChild(new test);
addChild(s);

s.addEventListener(MouseEvent.CLICK,fun);
function fun(e:MouseEvent){
  trace(e.target)
}
```

尝试使用这个类，trace 出来的结果是[object Seal]

但是我们把 mouseChildren 这个属性去掉
trace 出来的结果是[object test]

再来看看 mouseChildren 的官方解释：
确定对象的子项是否支持鼠标。 如果对象支持鼠标，则用户可以使用鼠标与其交互。

打个比喻，就好像一个盒子，你放了很多东西在里面，这个时候你设置了这个属性，就好比把盒子的盖子关上了，这个时候，你怎么看这个盒子，他就是一个盒子，你拿着这个盒子去问别人，这是什么，得到答案肯定也是盒子，这个时候你把盖子打开，也就是把这个属性去掉，你指着盒子里面，再去问别人这是什么，他肯定不会说盒子了，而会说盒子装的那个东西

那么我们在做虚拟形象类等等，加了这个属性后，你怎么往虚拟形象类里面放东西，给他穿衣服，戴首饰等等，外面来看，目标(target)抓取的只是虚拟形象类，这么就达到我们先头要实现的东西。

使用 SWFObject 插入 Flash 在 IE 下导致 stageWidth 为 0 的解决方法

作者：A 客网

文章来源：http://www.51as.com/as3/shiyongSWFObjectcharuFlashzaiIExiadaozhistageWidthwei0dejiejuefangfa_232.html

SWFObject

SWFObject是一个用于在HTML中方面插入Adobe Flash媒体资源（*.swf文件）的独立、敏捷的JavaScript模块。该模块中的JavaScript脚本能够自动检测PC、Mac机器上各种主流浏览器对Flash插件的支持情况。它使得插入Flash媒体资源尽量简捷、安全。而且它是非常符合搜索引擎优化的原则的。此外，它能够避免您的 HTML、XHTML中出现object、embed等非标准标签，从而符合更加标准。

SWFObject 在 IE 下的 BUG

如果 Flash 里绘制的对象的宽高是自适应 Flash 的宽高的，那么，使用 SWFObject 来插入 Flash 在 IE 会导致一个问题，当这个 Flash 被缓存后，也就是第二次访问该页面时，在该 Swf 文件被加载时，获取到的 stage.stageWidth 和 stage.stageHeight 为 0，绘制的对象也就看不到了。

在 Flash 里监听 resize 事件，找出解决方法

AS 里的解决方案

通过监听 resize 事件，当 stage.stageWidth 和 stage.stageHeight 大于 0 时再进行初始化

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.text.TextField;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
   public class AutoSizeExample extends Sprite
    {
        private var txt:TextField;
        public function AutoSizeExample()
        {
            stage.align = StageAlign.TOP_LEFT;
            stage.scaleMode = StageScaleMode.NO_SCALE;
```

```
        txt = new TextField();
         txt.multiline = true;
         txt.wordWrap = true;
         txt.text = "info\n";
         addChild(txt);
        if (stage.stageWidth>0 && stage.stageHeight>0){
            createChild();
         }else{
            stage.addEventListener(Event.RESIZE,onResize);
         }
     }
    private function onResize(e:Event):void
     {
         if (stage.stageWidth>0 && stage.stageHeight>0){
            stage.removeEventListener(Event.RESIZE,onResize); //删除事件监听
            createChild();
         }
         //否则继续监听事件，直到 stage.stageWidth 和 stage.stageHeight 大于 0 时才初始化
     }
    private function createChild():void
     {
         //进行初始化操作，创建各对象
         //.....................
         var w:Number = stage.stageWidth;
         var h:Number = stage.stageHeight;
        txt.appendText(w + " x " + h + "\n");
```

```
        }

    }

}
```

SWFObject.js 里的解决方案

在 swfobject.js 里找到函数 function createSWF(attObj, parObj, id)

里面针对 winIE 的处理方法是：el.outerHTML = '<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"' + att + '>' + par + '</object>';

看了一下 Adobe 自己的 AC_OETags.js，他是用 document.write 方法直接写入，没有这个问题；

直接写静态 html(object)没有这种问题；

直接访问这个flash的地址，也没有这种问题。

至于 el.outerHTML 的方式为什么会导致这种问题，还没有深究，暂时还不太清楚是什么原因导致的~

谈谈加密和混淆

作者：天地会

文章来源：http://flash.9ria.com/thread-25123-1-1.html

还是谈谈自己一直在做的一个应用：flash加密及混淆。从去年 8，9 月份开始研究。先是研究swf的文件结构，然后是abc的结构。慢慢也明了swf运行的原理。特别是研究abc结构后，收获很多。然后在写as代码的时候会联想到这些会编译后对应的指令。比如 var a:Number =3。对应指令就先在local数据中设置一个空间：null，接着一个指令将该空间转换为Number类型，然后添加一个byte数据 3 到scope，最后才是指令将 3 复制到给先前的空间。a字段是不会被编译进去的，这就是为什么用编译软件看到的临时变量都不是原先的名称；再引申下，你可以尽量明了的命名临时变量，比如：var theFirstValue:Number = 0，而不用var b:Number =0。很明显theFirstValue很适合阅读，而且反正也不会编译到swf中去，不用担心太长而增加swf体积。

接下来自己便开始做加密。先想到的是加壳的方法。后来和别人一交流，才发现，这种方法其实已经早已不是什么新想法了。只要你研究写swf的文件结构，就很容易想到这个方法。虽然后来想久，尝试很久，还是没有更好的加密方法。其实对于swf而言，已没有加密可言。除非借助与第三方交互。或者adobe自己的人去加密——毕竟swf文件结构白皮书公布的信息只是一部分而已。所以，最好还是混淆。

加密完成之后，我就开始研究混淆。研究过abc结构的人，应该都很清楚：要混淆一个方法、一个类名其实很容易。比如你有一个类ABCD，这时候你只要在abc数据中找到这个ABCD，替换为你想混淆成的样子，比如aaa1。重新组装abc，就完成了混淆。但是如果你有一个类alpha呢？你将alpha混淆成aaa1 了。然后你会发现你的代码：(newMovieClip).alpha = 0.3;变成了(new MovieClip).aaa1=0.3;——swf是一个高压缩的文件格式，abc也是如此。它不可能为同一个字段保存两字。也就是说你的类名alpha和属性alpha引用的是同一数据，当这个数据被改变是，一切引用该数据的地方都改变了。于是，你需要一个操作：将不混淆字段复制。就上面这个例子：将alpha数据复制一份给“newMovieClip”使用，而类alpha使用的一份进行混淆。
好，这样完成了一小部分。但是还会有问题。比如你从服务器接受了一个xml数据:

```
<root>
<title>best wish to demi</title>
</root>
```

然后你会通过xml.title来得到这个数据。但是你恰恰这个时候有类方法:public function gettitle():String。然后你混淆了title为aaa1，然后xml.titlt变为xml.aaa1 了，就获取不到数据了。于是又涉及到一个问题了：特殊字段。还比如：
getDefinitionByName("MainAPP")； MainAPP是特殊字
ExternalInterface.call("eval", "alert('asdf');"); eval是关键字
等等。有了这些特殊字和copy字段后，就可以进行大力混淆了。当然本人研究也是有限，很多情况没有遇到过，所以肯定会漏掉一些情况。

以上只是自己的对加密和混淆的一些心得。有不对的地方，敬请指正。

ps:最近很不开心~~~~ 汗~~
另外就是可以下载DoSWF,里面有上面谈到的加密和混淆功能。

小谈ActionScript 3.0 与AS 2.0，1.0 的swf兼容性

作者：黑羽翔天

文章来源：http://www.kingda.org/archives/kingda/2006/06/actionscript_30as_2010swf.html#more

众所周知，Flash Player 9 中为了解决兼容性问题，内置了两个虚拟机。AVM1 来对付 AS1.0&2.0，AVM2 用来专门处理 AS3.0。(注: 本文中 ActionScript 均简写成 AS)
用 AS3.0 中的 SWFLoader 来装载 swf, img 非常方面爽快。于是黑羽就想到一个问题:
可不可以用 SWFLoader 来装载使用 AS2.0 的组件(或 1.0)编译过的 swf 呢？
可不可以通过 AS2.0 来装载 AS3.0 的 swf 呢？如果可以，那么海量的 Flex2 组件会让大家爽死！
如果可以的话，那岂不是可以轻松重用以前编的 2.0 组件了吗？

如果真的可以，那么虚拟机到底怎么工作呢？AVM2 与 AVM1 通信？？

经过黑羽的探索、试验、查询资料后，答案是折衷的，不是很爽。 听俺慢慢道来：

1.使用 AS3.0 确实可以加载 AS2.0 或者 1.0 的 swf。但是 AS3.0 不可以访问加载 swf 中变量和函数。为了方便理解，我们可以想成两个虚拟机并行工作，但是不能通信。事实上，我猜实现机制可能就和这个差不多。LocalConnection。
2.使用 AS2.0 或 1.0 编写的 swf 是不可以加载 AS3.0 的。换句话说 Flash 8＆Flex 1.5 及之前所有工具生成的 swf 都不可以加载(load)AS 3.0 swf 的。
3.如果想让 AS2.0 或 1.0 的 swf 与 AS3.0 swf 协同工作，那么 AS2.0＆1.0 的文件必须进行移植。就是说转成 3.0。

黑羽在 yy:不知道 Blaze 出来后会不会有另存为自动转换成 AS3.0 的功能，霍霍。

4.单个的 swf 文件中是不能混合使用 AS3.0＆AS2.0(或者 1.0)的。 不会像 AS2.0＆1.0 那样混用了，毕竟是 AS2.0&AS3.0 是两个不相同的虚拟机。

一句话总结，就是Flash Player 9（AS3.0）可以加载以前的所有版本的swf,但是只是简单加载，不能访问AS2.0(或1.0)的swf内部变量＆函数，无法交互。

如果一定要和AS2.0 或 1.0 的swf通讯的话，只能使用ExternalInterface类，具体教程看
http://www.flex2.org/node/97

用位运算优化你的 AS3 代码

作者：A 客网

文章来源：http://www.51as.com/as3/yongweiyunsuanyouhuanideAS3daima_145.html

在AS3 中位操作是非常快的，这里列出一些可以加快某些计算速度的代码片段集合。我不会解释什么是位运算符，也不会解释怎么使用他们，只能告诉大家如果想清楚其中的原理这里有一篇极好的文章在gamedev.net上叫做 'Bitwise Operation in C' .

如果你知道任何下边没有列出来的不错的技巧，请留下个评论或者给我发个邮件。所有这些都是基于AS3 的

左位移几就相当于乘以 2 的几次方（ Left bit shifting to multiply by any power of two ）
//将一个值向左侧移动一位与这个值乘以 2 等效 。浮点数通过舍去小数点后面的所有位来转换为整数。
大约快了 300%

```
x = x * 2;
x = x * 64;
//相当于：
x = x << 1;
x = x << 6;
```

右位移几就相当于除以 2 的几次方（Right bit shifting to divide by any power of two）
//将一个值右移一位等效于将它除以 2 并舍去余数。浮点数通过舍去小数点后面的所有位来转换为整数。
大约快了 350%

```
x = x / 2;
x = x / 64;
//相当于：
x = x >> 1;
x = x >> 6;
```

Number 到 integer(整数)转换

在 AS3 中使用 int(x)快了 10% 。尽管如此位操作版本在 AS2 中工作的更好

```
x = int(1.232)
//相当于：
x = 1.232 >> 0;
```

提取颜色组成成分

不完全是个技巧，是正常的方法 (Not really a trick, but the regular way of extracting values using bit masking and shifting.)

```
//24bit
var color:uint = 0x336699;
var r:uint = color >> 16;
var g:uint = color >> 8 & 0xFF;
var b:uint = color & 0xFF;
//32bit
var color:uint = 0xff336699;
```

```
var a:uint = color >>> 24;
var r:uint = color >>> 16 & 0xFF;
var g:uint = color >>> 8 & 0xFF;
var b:uint = color & 0xFF;
```

合并颜色组成成分

替换值到正确位置并组合他们 (‘Shift up’ the values into the correct position and combine them.)

```
//24bit
var r:uint = 0x33;
var g:uint = 0x66;
var b:uint = 0x99;
var color:uint = r << 16 | g << 8 | b;
//32bit
var a:uint = 0xff;
var r:uint = 0x33;
var g:uint = 0x66;
var b:uint = 0x99;
var color:uint = a << 24 | r << 16 | g << 8 | b;
```

使用异或运算交换整数而不需要用临时变量

很可爱的技巧, 在本页顶端的链接里有详细的解释 ，这里快了 20%

```
var t:int = a;
a = b;
b = t;
//相当于:
a ^= b;
b ^= a;
a ^= b;
```

自增/自减(Increment/decrement)

这个比以前的慢不少，但却是个模糊你代码的好方法；-）

```
i = -~i; // i++
i = ~-i; // i--
```

取反（Sign flipping using NOT or XOR）

另人奇怪的是这个居然快了 300%!

```
i = -i;
//相当于：
i = ~i + 1;
//或者
i = (i ^ -1) + 1;
```

使用 bitwise AND 快速取模 （Fast modulo operation using bitwise AND）

如果除数是 2 的次方，取模操作可以这样做：

模数= 分子 & (除数 - 1);

这里大约快了 600%

x = 131 % 4;

//相当于：

x = 131 & (4 - 1);

检查是否为偶数（Check if an integer is even/uneven using bitwise AND）

这里快了 600%

isEven = (i % 2) == 0;

//相当于：

isEven = (i & 1) == 0;

绝对值

忘记 Math.abs()吧 (Forget Math.abs() for time critical code.)

version 1 比 Math.abs() 快了 2500% ，version 2 居然比 version 1 又快了 20% ！

```
//version 1
i = x < 0 ? -x : x;
//version 2
i = (x ^ (x >> 31)) - (x >> 31);
```

//后两个没看懂

Comparing two integers for equal sign

This is 35% faster.

```
eqSign = a * b > 0;
//equals:
eqSign = a ^ b >= 0;
Fast color conversion from R5G5B5 to R8G8B8 pixel format using shifts
R8 = (R5 << 3) | (R5 >> 2)
G8 = (R5 << 3) | (R5 >> 2)
```

翻译原文出处:http://www.nshen.net/blog/article.asp?id=531 译者:N神

小编:感谢YangPuxiao 提出转载注明的问题，我们其实一直尽力能找到原作者和原出处地址,但有时候是非常无力的. 所以如果各位同学发现本站标注的作者和出处有误请联系我 : 51as.com@gmail.com 或者 直接发表评论指出. 我们即时纠正,再次感谢YangPuxiao,向N神兄表示歉意

在 AS3 中实现运行时强制的 Abstract Class（抽象类）

作者：DN_Web

文章来源：http://hi.baidu.com/dn%5Fweb/blog/item/64a1bb1a6b89ecdcac6e7523.html

转载(http://zkp1997.blog.163.com/blog/static/800976720089150541520/)

大家都知道，要实现抽象类必须满足以下要求：1.抽象类不能被实例化 2.抽象类内中的抽象方法必须被子类重写。

在许多标准的 OOP 语言中都有抽象类的实现，如 Java、C 语言等。但是，ActionScript3 中没有抽象类的实现，必须要我们手动实现。

笔者用String函数解决了这个问题，借鉴了Tink（详见http://www.tink.ws/blog/stricter-abstracts/）的方法并加以改进。例子如下。

```
package zhou.kunpeng.test{
import flash.display.Sprite;


public class SampleAbstract extends Sprite {

   public function SampleAbstract() {
     var foo:AbstractFoo=new AbstractFoo();

    //报错：异常：不能创建抽象类 AbstractFoo 的实例。
     var bar:ExtenderFoo=new ExtenderFoo();
     bar.AbstractMethod();

    //报错：异常：不能实现抽象方法。抽象方法应被子类重写。
   }
}
}
class AbstractFoo extends Object{
public function AbstractFoo() {
   if (String(this) == "[object AbstractFoo]" ) {
    throw new Error("异常：不能创建抽象类 AbstractFoo 的实例。");
   }
}
public function AbstractMethod():void {
   throw new Error("异常：不能实现抽象方法。抽象方法应被子类重写。");
}
}
class ExtenderFoo extends AbstractFoo {}
```

转师傅的--事件流那点事

作者：DN_Web
文章来源：http://hi.baidu.com/dn%5Fweb/blog/item/9b34cf1fa5a5d01540341724.html

在群里有人问了个问题：
为什么我把父剪辑的 mouseEnabled 设置为 false 但父剪辑还是可以侦听到 鼠标事件？

很有意思的一个问题，不是么！

AS3 里使用了事件流机制。简单说来一个完整的事件流应该从 Flash Player 到目标然后再回到 Flash Player。我们用一个简单的例子看看如何描述这个问题。

假如有如下结构的 SWF 文件，场景里有一个 MovieClip(以下简称 MC) A，A 里面包含一个名为 B 的 MC，B 里面又包含 MC D。 有些复杂了，还是画个简单的图吧。
|-stage
..|-A
....|-B
......|-D

然后我们假设 D 触发了 CLICK 事件，那么事件流程应该是：

stage -> A -> B -> D -> B -> A -> stage

这里有几个定义， 先看这一段：[stage -> A -> B]， 这段在事件流当中我们定义为 捕获阶段(是否记得 addEventListener 的第三个参数？)；

我们触发事件的对象 [D] 也给它定义一个阶段，命名为：目标阶段；

最后 [B -> A -> stage] 这阶段定义为 冒泡阶段。

PS：小建议，如果觉得搞不清为什么事件为什么会从 stage 跑到目标再回到 stage 这样的问题的话，还是别去管它了。只要记住不管这个事件在哪里发生， 总是会从 stage 出发，然后找到触发事件的对象，再回到 stage。

这样，一个完整的事件流由 捕获, 目标和冒泡 这三个阶段构成。

更多事件流的信息请访问黑羽博客：http://www.kingda.org/archives/kingda/2006/07/as305.html

扯了这么远，回到开始的话题。

当我们设置对象的 mouseEnabled 值的时候，它只影响对象本身不会触发事件，但这并不影响对象在事件件中的地位。也就是说当对象的子对象触发鼠标事件的时候，自身也会伴随发送事件。想想整个事件流，也就不难理解为什么我把父剪辑的 mouseEnabled 设置为 false 但父剪辑还是可以侦听到 鼠标事件。

所以，要禁止一个对象的触发交互事件，完整的做法是 mouseEnabled=false; mosueChildren=false; 同时设置这两个值。

好了，说了一堆全是文字。动手试一下：

1. 在场景里画一个圆，选个红色或是什么的，保存为影片剪辑；
2. 继续在场景里画一个方框，颜色和前面的圆区分开就行，位置最好重叠；
3. 同时选中的圆(影片剪辑)和刚画好的方框，保存影片剪辑；
4. 再在主时间轴上添加代码：getChildAt(0).addEventListener(MouseEevnt.MOUSE_OVER, function (e){trace("overed");});
5. Ctrl+ENTER 运行测试，看看运行效果。
6. 再添加一行代码：getChildAt(0).mouseEnabled = false;
7. 再测试运行。是不是注意到区别了呢？

文件制作简单，就不附了。enjoy!

真佩服师傅,琢磨东西以及记录东西的耐心跟细心,而且不乏幽默感!.容易懂!

做飞机游戏经常使用的方向控制类 Direction.as

作者：小 S

文章来源：http://www.xiaos8.com/default.asp?cat=1&page=12

把焦点置于 flash 中，按键盘的上下左右方向键，就可以看到效果

点击播放/隐藏媒体

http://www.xiaos8.com/uploads/pro/direction.swf

首先是代码！

Direction.as 文件

```
package index.base.game{

  import flash.events.EventDispatcher;
  import flash.events.KeyboardEvent;
  import flash.events.Event;
  import flash.display.InteractiveObject;

  import index.base.events.DirectionEvent;

  public class Direction extends EventDispatcher{

    //方向表示
    public static const UP:uint = 0;
    public static const DOWN:uint = 1;
    public static const LEFT:uint = 2;
    public static const RIGHT:uint = 3;

    //作用区域
    public var area:InteractiveObject;

    //上下左右键值
    private const directionAr:Array = new Array(4);

    //是否上下左右
    private var _up:Boolean = false;
    private var _down:Boolean = false;
    private var _left:Boolean = false;
    private var _right:Boolean = false;

    public function Direction(_area:InteractiveObject,_up:uint = 38,_down:uint = 40,_left:uint =
37,_right:uint = 39){
      area = _area;
      directionAr[UP] = _up;
      directionAr[DOWN] = _down;
```

```
    directionAr[LEFT] = _left;
    directionAr[RIGHT] = _right;
    start();
  }

  //开始获取事件
  public function start():void{
    area.addEventListener(KeyboardEvent.KEY_DOWN,onKeyDown);
    area.addEventListener(KeyboardEvent.KEY_UP,onKeyUp);

    area.addEventListener(Event.ENTER_FRAME,onEnterFrame);
  }

  //事件帧频繁触发
  private function onEnterFrame(e:Event):void{
    var num:uint = Number(_up) + Number(_down) + Number(_left) + Number(_right);
    if(num == 0){
      return;
    }

    var eve:DirectionEvent = new DirectionEvent(DirectionEvent.DO);
    eve.up = _up;
    eve.down = _down;
    eve.left = _left;
    eve.right = _right;
    dispatchEvent(eve);
  }

  //停止获取事件
  public function stop():void{
    area.removeEventListener(KeyboardEvent.KEY_DOWN,onKeyDown);
    area.removeEventListener(KeyboardEvent.KEY_UP,onKeyUp);
  }

  //鼠标按下去事件
  private function onKeyDown(e:KeyboardEvent):void{
    key(e.keyCode,true)
  }

  //鼠标弹上来事件
  private function onKeyUp(e:KeyboardEvent):void{
    key(e.keyCode,false)
  }

  //变化状态
  private function key(num:uint,isDown:Boolean):void{
    switch(num){
      case directionAr[UP]:
        _up = isDown;
```

```
        break;
        case directionAr[DOWN]:
          _down = isDown;
        break;
        case directionAr[LEFT]:
          _left = isDown;
        break;
        case directionAr[RIGHT]:
          _right = isDown;
        break;
      }
    }

    //设置按钮
    public function setKey(num:uint,vars:uint){
      directionAr[num] = vars;
    }
  }
}
```

DirectionEvent.as 文件

```
package index.base.events{

  import flash.events.Event;

  public class DirectionEvent extends Event{

    public var up:Boolean;
    public var down:Boolean;
    public var left:Boolean;
    public var right:Boolean;

    public static const DO:String = "do";

    public function DirectionEvent(type:String){
      super(type);
    }
  }
}
```

使用情况！

```
import index.base.game.Direction;
import index.base.events.DirectionEvent;

new Direction(stage).addEventListener(DirectionEvent.DO,doFun);

function doFun(e:DirectionEvent){
  if(e.up) mc.y -= 5;
  if(e.down) mc.y += 5;
```

```
  if(e.left) mc.x -= 5;
  if(e.right) mc.x += 5;
}
```

初略讲解：
利用事件管理的特性，当点击向上键，则 up 的逻辑值为 true，当松开向上键，则 up 的逻辑值为 false，且只要存在一个 true 值，也就是说存在一个人按钮被按下去，就持续调度事件 DirectionEvent.DO！

静态属性讲解：
文章一开始有 UP，DOWN 等大写字母表示的数字，这个是方便在使用 setKey 的时候，第一个参数可以直接填写 Direction.UP，则表示修改 up 的键值

面对新手需要讲解的几个地方：
1、

```
    //事件帧频繁触发
    private function onEnterFrame(e:Event):void{
      var num:uint = Number(_up) + Number(_down) + Number(_left) + Number(_right);
      if(num == 0){
        return;
      }

      var eve:DirectionEvent = new DirectionEvent(DirectionEvent.DO);
      eve.up = _up;
      eve.down = _down;
      eve.left = _left;
      eve.right = _right;
      dispatchEvent(eve);
    }
```

上面这段代码，关于 num 值的获取，也许会有些人看不明白，其实意思就是把 4 个逻辑值强行转换为数字型，如果为 false 那么结果为 0，如果 4 个都为 false 则加起来
num 应该等于 0，这就表明没有任何一个方向键按下，则 return！不执行后面事件 DirectionEvent.DO 的调度
反之，不等于 0，就说明必定有一个以上的按键按下，则调度事件！

2、
构造函数中使用了抽象类 InteractiveObject
也就是说继承了 InteractiveObject，或者间接继承了 InteractiveObject，都可以作为方向的焦点
比如：Stage，Sprite，SimpleButton，Loader，MovieClip……

到此结束，剩下的个人认为都好理解，如果有啥不明白的，可以联系我！