贴吧：tieba.baidu.com/p/1472552048（原帖已失效）

9ch：9ch.co/t27615,1-1.html

**正文**

word文档下载:[115](http://web.archive.org/web/20121113204940/http://115.com/file/e7lbwpl3)  
**编写：小包子  
技术顾问：面汤拌菜菌**  
  
终于坑出来了...虽然我自己找了不少打错的字..但是肯定还有很多.  
嘛..问题肯定不少.有问题请跟帖.

**前言**

高级弹幕不难.基础的部分非常简单.只需要你花上一点时间就可以轻松上手.重要的是需要对作品的理解以及对时间的把握.无论什么弹幕只需要一颗有爱的心就够了.弹幕需要是爱与人参.所以我希望大家不要去使用高级弹幕去做不好的事情.

**目录**

**第一章 初步了解高级弹幕**

       1.1 变量

       1.2创建文本弹幕对象

       1.3 弹幕对象运动的设置

       1.4 绘图弹幕对象创建以及设置

              1.4.1颜色样式设置

                     1.4.1.1 beginFill 指定一种颜色进行填充

                     1.4.1.2 lineGradientStyle beginGradientFill指定一种线条样式,填充样式的渐变

                     1.4.1.3 lineStyle 指定线条样式

              1.4.2图形的绘制

                     1.4.2.1 绘制一条直线

                     1.4.2.2 绘制一条二次贝塞尔曲线

                     1.4.2.3 绘制一个圆

                     1.4.2.4 绘制一个椭圆

                     1.4.2.5 绘制一个矩形

                     1.4.2.5 绘制一个圆角矩形

**第二章 使用其他工具**

       2.1Utils 工具库

              2.1.1 hue将0-360的值映射到色相环

              2.1.2 rgb将RGB值映射到色彩值上

              2.1.3 formatTimes格式化播放时间

              2.1.4 timer 延迟执行函数

              2.1.5 interval 定时重复执行函数

              2.1.6 distance 计算坐标距离

              2.1.7 rand 返回一个前闭后开的整数

       2.2 Function 函数的创建与使用

       2.3 播放器控制

              2.3.1视频时间跳转至

              2.3.2跳转到指定av号指定页视频

**第一章 初步了解高级弹幕**

**1.1 第一节 变量**

废话不多说.我们先来看一句最基本的弹幕.

|  |
| --- |
| 1. var a = 1; |

首先一次个关键字是”var”.该关键字的意思就是声明一个变量.变量说的简单点就是在后面的过程中可以给 “a” 重新赋值.  
然后就是变量名”a”了,这玩意理解成名字就行了.至于关于变量名的规则有些关键字与保留字是不允许创建的.[保留字](http://web.archive.org/web/20121113204940/http://wiki.bilibili.tv/wiki/ECMAScript%E4%BF%9D%E7%95%99%E5%AD%97)，[关键字](http://web.archive.org/web/20121113204940/http://wiki.bilibili.tv/wiki/ECMAScript%E5%85%B3%E9%94%AE%E5%AD%97)  
第三个是运算符”=”,首先等于号并不是数学中的等于意思.而是给声明的变量a赋值.将等号右边的值赋予给左边.  
第四个”1”就是一个值了.结尾的分号表示该语句结束.

看到这里相信大家已经对这句弹幕有了基础的了解.这句话的意思就是：

声明变量”a”并且初始化其值为数字1.

写高级弹幕的时候请严格注意大小写.

|  |
| --- |
| 1. var 变量名 = 表达式; |

**1.2 第二节 创建文本弹幕对象.**

首先要知道bili的播放器的大小.像素为单位.  
**正常模式：宽541 高384  
宽屏模式：宽950 高528**  
全屏模式与网页全屏模式根据显示器分辨率来确定.  
**在高级弹幕中在 实际弹幕时间会在发布时间轴填写的时间基础上减去约0.2秒.所以请注意时间轴的问题**

高级弹幕中创建新的文本弹幕对象要用到createComment（"",{}）这个语句.请看实例

|  |
| --- |
| 1. var a = $.createComment("text ",{x:100}); |

该弹幕的意思就是 创建一个变量a并创建一个新弹幕对象赋于a.

我们可以这么理解createComment.他的括号内有2个参数.”text”是弹幕内容.｛x:100｝是该弹幕参数x轴上的位置是100.  
当然括号内的不可能只有x这一个参数.我们可以用到的有.

**x**— 新创建元件的X轴座标

**y**— 新创建元件的Y轴座标

**z**— 新创建元件的Z轴座标

**lifeTime**— 元件的生存时间(以秒为单位)注：一旦设置不可更改

**alpha —** 元件的透明度

**color —** 文字类元件的色彩

**fontsize**— 文字类元件的大小

**parent**— 元件的父元件 (进阶应用)

**motion**— 元件移动策略

**scale —** 缩放

那么我们来定义一个稍微复杂点的高级弹幕.  
例：

|  |
| --- |
| 1. var c = $.createComment("text",{ 2. x:100,y:100, 3. lifeTime:10, 4. fontsize:20, 5. color:0x666666, 6. alpha:0.8 7. }); |

(注：参数之间以英文逗号区分,最后一个参数结尾无逗号)  
大家可以预览一下这条弹幕.就是在x轴100,y轴100的位置创建一个内容是text的弹幕.该弹幕生存时间为10秒.字体大小为20号.十六进制颜色为灰色.透明度0.8.

那么.到了这里大家应该已经知道怎么去在一个点创建弹幕并设置该弹幕各项参数.  
接下来就来说一下在创建后各项属性的更改与设置.

现在.声明变量c并创建了一个弹幕内容为"测试"的弹幕.它的x轴为50,y轴为100.  
例：

|  |
| --- |
| 1. var c = $.createComment("测试",{x:50,y:100}); |

那么如果我们现在需要让他的位置x轴不变y轴变更到50.那么我们这个时候就需要直接更改弹幕的某一个参数.只需要在弹幕后面加上.

|  |
| --- |
| 1. c.y = 50; |

（注：更改颜色的语句不是c.color而是c.textColor.注意大小写）

此外还有比较常用的就是旋转3个轴的设置.  
c.rotationX,rotationY,rotationZ.  
例:

|  |
| --- |
| 1. c.rotationX=90; |

**设置字体为**：

|  |
| --- |
| 1. c.font="宋体"; |

**设置光昏效果**：

|  |
| --- |
| 1. c.filters=[$.createGlowFilter(0xFF0000,1,16,17,2,1,false,true)]; |

createGlowFilter 参数**color** **上例参数 0xFF0000**  光晕颜色，采用十六进制格式 0xRRGGBB。默认值为 0xFF0000。  
**alpha** **上例参数 1**  颜色的 Alpha 透明度值。有效值为 0 到 1。例如，0.25 设置透明度值为 25%。  
**blurX 上例参数 16**  水平模糊量。有效值为 0 到 255（浮点）。2 的乘方值（如 2、4、8、16 和 32）经过优化，呈示速度比其他值更快。  
**blurY 上例参数 17**  垂直模糊量。有效值为 0 到 255（浮点）。2 的乘方值（如 2、4、8、16 和 32）经过优化，呈示速度比其他值更快。  
**strength** **上例参数 2**  印记或跨页的强度。该值越高，压印的颜色越深，而且发光与背景之间的对比度也越强。有效值为 0 到 255。  
**quality** **上例参数 1**  应用滤镜的次数。使用 BitmapFilterQuality 常量：low，middle，high  
**inner 上例参数false**  指定发光是否为内侧发光。值 true 指定发光是内侧发光。值 false 指定发光是外侧发光（对象外缘周围的发光）。  
**knockout** **上例参数true**   指定对象是否具有挖空效果。值为 true 将使对象的填充变为透明，并显示文档的背景颜色。  
**设置字体无黑边：**

|  |
| --- |
| 1. c.filters=null; |

另还有些参数也可以设置.[详细参考](http://web.archive.org/web/20121113204940/http://wiki.bilibili.tv/wiki/CommentField)

**1.3 弹幕对象运动的设置**

在上一个章节中学习了如何创建文本对象和设置文本对象的参数.这次就讲如何让创建的弹幕对象运动起来.这里我们要用到motion.下面请看一个例子.

|  |
| --- |
| 1. var c = $.createComment("移动测试",{ 2. motion:{ 3. x:{fromValue:100,toValue:250,lifeTime:3} 4. }, 5. lifeTime:3 6. }); |

让我们来分析这条弹幕.

首先上已经了解过了createComment的用法.那么我直接来看到多出来的这个属性motion.

motion:{ x:{fromValue:100,toValue:250,lifeTime:3}}

在这里面其中x为需要移动的参数.x中的fromValue是x的起始位置toValue是结束位置.lifeTime是这个运动过程所花费的时间.(注:如果在motion里面x.y有起始值并且在motion外也设置x.y的值,那么将显示motion里面的设置位置)

此外motion支持多属性同时运动.

例：

|  |
| --- |
| 1. var c = $.createComment("移动测试",{ 2. motion:{ 3. x:{fromValue:100,toValue:250,lifeTime:3}, 4. alpha:{fromValue:0,toValue:1,liftTime:3} 5. }, 6. lifeTime:3 7. }); |

这次除了移动x轴还添加了一个透明度从0到1的变化.

下面我们说一下在motion中有哪些是可以做变化的.

**x**:移动x轴位置.

**y**:移动y轴位置.

**alpha**:透明度变换（有效值0-1）.

**fontsize**:字体大小变换（不推荐使用.字号的变换较明显.

**rotationX, rotationY, rotationX**:旋转x轴,旋转Y轴,旋转Z轴.

其中的属性值有:

必填       **fromValue**   起始移动属性值

可选       **toValue**   结束移动属性值如留空则不移动

可选       **lifeTime**    以秒为单位的移动生存时间如留空则与整体生存时间一致

可选       **startDelay**    以毫秒为单位的起始移动延时时间（毫秒为单位）

可选       **easing**详细请看[补间效果](http://web.archive.org/web/20121113204940/http://static.loli.my/MotionSample.swf" \t "http://web.archive.org/web/20121113204940/9ch.co/_blank)

可选       **repeat**效果重复次数

下面我们来看一个相对复杂的例子

|  |
| --- |
| 1. var c = $.createComment("移动测试",{ 2. motion:{ 3. x:{fromValue:100,toValue:250,lifeTime:3,startDelay:500}, 4. y:{fromValue:50,toValue:250,lifteTime:3}, 5. alpha:{fromValue:0,toValue:1,lifeTime:0.5,repeat:6} 6. }, 7. lifeTime:3}); |

在这个例子里面我们用到了startDelay这个属性.我们把这个代码拆开分了3段来看.

首先x轴的是从100移动到250.移动时间3秒.然后起始移动时间是500毫秒.也就是说x轴是在500毫秒后开始移动的.

然后是y轴从50移动到250移动时间3秒.

最后是透明度从0到1变化时间为0.5秒.重复执行6次.

把这三个参数整合在一起就变成了.移动测试从x：100,y：50的位置开始移动,并且透明度最开始是0.由于x有一个起始移动时间.但是y轴跟透明度变化已经开始.所以首先"移动测试"会先渐入的向下移动.500毫秒后x开始移动这个时候y轴的移动还没有停第一次的透明度改变已经结束.所以x,y轴这个时候是同时移动的.最后结果回是往右下移动并且透明度会重复的从0到1..

虽然可以使用repeat这个属性执行重复的操作.但是这个是远远不能满足复杂的移动效果的.然而去创建很多motion也是很麻烦的.这里我就需要用到motionGroup了.motionGroup能执行多个motion连续运动.

我们来看一个实例.

|  |
| --- |
| 1. var c = $.createComment("弹幕测试",{ 2. x:100,y:50, 3. motionGroup:[ 4. {x:{fromValue:100,toValue:200,lifeTime:1},alpha:{fromValue:0,toValue:1,lifeTime:1}}, 5. {y:{fromValue:50,toValue:150,lifeTime:1},alpha:{fromValue:1,toValue:0,lifeTime:1}} 6. ] 7. });) |

以上弹幕就是使用一个组将两个移动连接在了一起→↓.需要注意的是motionGroup的冒号后面的是英文中括号.并且每一次移动包括透明度都最好加上lifeTime并且每一个motion中的lifeTime的时间必须一致..每一次motion之间用英文逗号相隔.最后一个结尾无符号.

**1.4 绘图弹幕对象创建以及设置**

使用命令：createShape

例子：

|  |
| --- |
| 1. var g = $.createShape({x:50,y:50,lifeTime:3,alpha:0.5}); |

首先创建一个新的图形对象.里面可以设置的内容同文本对象（位置,生存时间,透明度,运动设置等方法同creatComment）.

绘图需要使用到的：

**curveTo**  
使用当前线条样式绘制一条二次贝塞尔曲线

**drawCircle**  
绘制一个圆.

**drawEllipse**  
绘制一个椭圆

**drawRect**  
绘制一个矩形

**drawRoundRect**  
绘制一个圆角矩形

**lineTo**  
绘制一条直线

**beginFill**指定一种颜色进行填充

**lineStyle**指定线条样式

**lineGradientStyle**  
指定一种线条样式的渐变

**beginGradientFill**  
指定一种填充样式的渐变

我们来一个个的来了解.首先要了解的并不是怎么去绘制而是设定绘制的样式

**1.4.1颜色样式设置**

**1.4.1.1 beginFill指定一种颜色进行填充**

这里用到：

**beginFill**指定一种颜色进行填充

在beginFill里面有2个参数.第一个是十六进制的颜色,第二个是填充的透明度（可选）.

实例：

|  |
| --- |
| 1. var g=$.createShape({x:50,y:50,lifeTime:3,alpha:0.5}); 2. g.graphics.beginFill(0xff0000,0.5); 3. g.graphics.beginFill(0xff0000); |

这里用的一长串就是设置g的填充颜色以及透明度.两个都是有效的.

graphics则是是提供绘图的API.以后使用的时候都会用到这个.后面小括号内就是参数了.注：graphics前面的g必须是创建图形对象的名字

**1.4.1.2 lineGradientStyle beginGradientFill 指定一种线条样式,填充样式的渐变**

这里需要用到：

**lineGradientStyle**  
指定一种线条样式的渐变

**beginGradientFill**  
指定一种填充样式的渐变

详细请看：[链接](http://web.archive.org/web/20121113204940/http://wiki.bilibili.tv/wiki/Display.createGraphic.graphics#lineGradientStyle)

|  |
| --- |
| 1. g.graphics.beginGradientFill("linear",[0xFF0000, 0x00FF00,0x0000FF],[1,1,1],[0x00,0x7f,0xff] ,$.createGradientBox(20, 20, 0, 0, 0),"reflect","rgb",0); |

**type** **上例参数"linear"**  
用于指定要使用哪种渐变类型的 GradientType 类的值：GradientType.LINEAR 或 GradientType.RADIAL。

**colors** **上例参数[0xFF0000, 0x00FF00,0x0000FF]**  
要在渐变中使用的 RGB 十六进制颜色值数组（例如，红色为 0xFF0000，蓝色为 0x0000FF 等等）。

**alphas** **上例参数[1,1,1]**

colors 数组中对应颜色的 alpha 值数组；有效值为 0 到 100。 如果值小于 0，Flash playe将使用 0。 如果值大于100，Flash Player 将使用 100。

**Ratios** **上例参数[0x00,0x7f,0xff]**  
颜色分布比率的数组；有效值为 0 到 255。该值定义 100% 采样的颜色所在位置的宽度百分比。 值 0 表示渐变框中的左侧位置，255表示渐变框中的右侧位置。 该值表示渐变框中的位置，而不是最终渐变的坐标空间，坐标空间可能比渐变框宽或窄。 为 colors 参数中的每个值指定一个值。

**matrix** **上例参数$.createGradientBox(20, 20, 0, 0, 0)**  
一个由 Matrix 类定义的转换矩阵。 Matrix 类包括 createGradientBox() 方法，通过该方法可以方便地设置矩阵，以便与 beginGradientFill() 方法一起使用，亦可使用$.createGradientBox()。

createGradientBox(20, 20, 0, 0, 0)中5个数值分别是.

**width**— 渐变框的宽度。

**height**— 渐变框的高度。

**rotation**—旋转量（以弧度为单位）。

**tx**— 沿 x 轴向右平移的距离（以像素为单位）。此值将偏移 width 参数的一半。

**ty**— 沿 y 轴向下平移的距离（以像素为单位）。此值将偏移 height 参数的一半。

**spreadMethod** **上例参数"reflect"**  
用于指定要使用哪种 spread 方法的

**interpolationMethod 上例参数"rgb"**  
用于指定要使用哪个值的 值：linearRGB 或 rgb

例如，假设有两种颜色之间的简单线性渐变（spreadMethod 参数设置为 reflect）。

**focalPointRatio** **上例参数"0"**  
一个控制渐变的焦点位置的数字。 0 表示焦点位于中心。 1 表示焦点位于渐变圆的一条边界上。 -1 表示焦点位于渐变圆的另一条边界上。 小于 -1 或大于 1 的值将舍入为 -1 或 1。

实际运用：

|  |
| --- |
| 1. var g = $.createShape({x:50,y:50,lifeTime:5}); 2. g.graphics.beginGradientFill("linear",[0xFF0000, 0x00FF00,0x0000FF],[1,1,1],[0x00,0x7f,0xff] , null,"reflect",0); 3. g.graphics.drawCircle(0,0,50); 4. g.graphics.endFill(); |

在实际运用中后面几个可以不填写或者值为**null**但是其中的**colors ,alphas, Ratios**数组必须存在并且对应数组长度.另**Ratios**数组内数值必须从小到大

**1.4.1.3 lineStyle 指定线条样式**

这里用到：

**lineStyle**指定线条样式

|  |
| --- |
| 1. g.graphics.lineStyle(1, 0x000000,1, false, "vertical","none", "miter", 10); |

**thickness** **上例参数1**  
一个整数，以磅为单位表示线条的粗细；有效值为 0 到 255。如果未指定数字，或者未定义该参数，则不绘制线条。 如果传递的值小于 0，则默认值为 0。 值 0 表示极细的粗细；最大粗细为255。 如果传递的值大于 255，则默认值为 255。

**color** **上例参数0x000000**  
线条的十六进制颜色值（例如，红色为 0xFF0000，蓝色为 0x0000FF 等）。 如果未指明值，则默认值为 0x000000（黑色）。 可选。

**alpha** **上例参数1**  
表示线条颜色的 Alpha 值的数字；有效值为 0 到 1。如果未指明值，则默认值为 1（纯色）。 如果值小于 0，则默认值为 0。 如果值大于 1，则默认值为 1。

**pixelHinting** **上例参数false**  
用于指定是否提示笔触采用完整像素的布尔值。

**scaleMode** **上例参数"vertical"**  
用于指定要使用哪种缩放模式的 LineScaleMode 类的值：

**normal**-- 在缩放对象时总是缩放线条的粗细（默认值）。

**none** -- 从不缩放线条粗细。

**vertical**-- 如果仅 垂直缩放对象，则不缩放线条粗细。

**horizontal**-- \*如果仅 水平缩放对象，则不缩放线条粗细。

**caps** **上例参数"none"**  
用于指定线条末端处端点类型的 CapsStyle 类的值。 有效值为：none、round 和 square. 如果未指示值，则 Flash 使用圆头端点。

**joints** **上例参数"miter"**  JointStyle 类的值，指定用于拐角的连接外观的类型。 有效值为：bevel、miter 和 round。 如果未指示值，则 Flash 使用圆角连接。

**miterLimit** **上例参数10**  
一个表示将在哪个限制位置切断尖角的数字。 有效值的范围是 1 到 255（超出该范围的值将舍入为 1 或 255）。

**miterLimit 值：  
小于此角度将被切断：**

**1.414      90度**

**2     60度**

**4     30度**

**8     15度**

实例：

|  |
| --- |
| 1. var g = $.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.lineStyle(1, 0x000000,1, false, "vertical","none", "miter", 10); 3. g.graphics.lineTo(100,0); 4. g.graphics.endFill(); |

**1.4.2图形的绘制**

这里用到

**curveTo**  
使用当前线条样式绘制一条二次贝塞尔曲线

**drawCircle**  
绘制一个圆.

**drawEllipse**  
绘制一个椭圆

**drawRect**  
绘制一个矩形

**drawRoundRect**  
绘制一个圆角矩形

**lineTo**  
使用当前线条样式绘制一条直线

**moveTo**

移动绘画位置到一个点

这里先说明绘画位置的问题.

使用createShape命令内参数x.y轴的位置为初始绘画位置.

使用moveTo命令可以移动当前绘画位置.

例

|  |
| --- |
| 1. g.graphics.moveTo(100,100); |

移动绘画位置到x轴100,y轴100的位置.

**1.4.2.1 绘制一条直线**

命令：lineTo();

参数：lineTo(x,y);

参数说明：

**x**:相对于父显示对象的水平坐标.

**y**:相对于父显示对象的垂直坐标.

例：

|  |
| --- |
| 1. var g =$.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.lineTo(200,200); 3. g.graphics.endFill(); |

以上弹幕看不到直线的？因为绘制线条之前必须设置线条样式或者设置填充样式绘制一个封闭图形.

正确的使用方法：

|  |
| --- |
| 1. var g =$.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.lineStyle(1, 0x000000,1, false, "vertical","none", "miter", 10); 3. g.graphics.lineTo(200,200); 4. g.graphics.endFill(); |

说明:该直线绘制直线的位置并不是播放器的x:200，y:200而是相对与createShape中100，100的基础上的x:200，y:200.也就是实际位置在播放器的x:300,y:300.

设置填充样式绘制一个封闭图形：

|  |
| --- |
| 1. var g = $.createShape({x:200,y:200,lifeTime:5}); 2. g.graphics.beginFill(0xff0000); 3. g.graphics.lineTo(200,200); 4. g.graphics.lineTo(100,200); 5. g.graphics.lineTo(100,100); 6. g.graphics.endFill(); |

当需要绘制多条不连续的直线时则需要使用到moveTo();

例：

|  |
| --- |
| 1. var g = $.createShape({x:200,y:200,lifeTime:5}); 2. g.graphics.lineStyle(1, 0x0000FF, 1, false, "vertical","none", "miter", 10); 3. g.graphics.lineTo(100,100); 4. g.graphics.moveTo(0,0); 5. g.graphics.lineTo(-100,100); 6. g.graphics.endFill(); |

**1.4.2.2 绘制一条二次贝塞尔曲线**

命令：curveTo();

参数：curveTo(**controlX**,**control**Y,**anchorX**,**anchorY**);

参数说明：

**controlX**: 指定控制点相对于父显示对象注册点的水平位置

**controlY**: 指定控制点相对于父显示对象注册点的垂直位置

**anchorX:**指定下一个锚点相对于父显示对象注册点的水平位置

**anchorY:**指定下一个锚点相对于父显示对象注册点的垂直位置

这里我们先不看例子**.**先来理解一下什么是控制点,什么是锚点.详细请看图.

以这样一个三角来确定绘制曲线.

实例:

|  |
| --- |
| 1. var g = $.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.lineStyle(1, 0x0000FF, 1, false, "vertical","none", "miter", 10); 3. g.graphics.curveTo(50,-50,100,0); 4. g.graphics.endFill(); |

**1.4.2.3 绘制一个圆**

命令：drawCircle ();

参数：drawCircle (**x**,**y**,**radius**);

参数说明：

**x**: 相对于父显示对象注册点的水平位置

**y**: 相对于父显示对象注册点的垂直位置

**radius:**圆的半径

**实例:**

|  |
| --- |
| 1. var g = $.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.beginFill(0xff0000); 3. g.graphics.drawCircle(0,0,30); 4. g.graphics.endFill(); |

**1.4.2.4 绘制一个椭圆**

命令：drawEllipse();

参数：drawEllipse(**x**,**y**,**width,height**);

参数说明：

**x**: 相对于父显示对象注册点的水平位置

**y**: 相对于父显示对象注册点的垂直位置

**width:**椭圆的宽度

**height:**椭圆的高度

**实例:**

|  |
| --- |
| 1. var g = $.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.beginFill(0xff0000); 3. g.graphics.drawEllipse(0,0,30,50); 4. g.graphics.endFill(); |

**1.4.2.5 绘制一个矩形**

命令：drawRect ();

参数：drawRect (**x**,**y**,**width,height**);

参数说明：

**x**: 相对于父显示对象注册点的水平位置

**y**: 相对于父显示对象注册点的垂直位置

**width:**矩形的宽度

**height:**矩形的高度

**实例:**

|  |
| --- |
| 1. var g = $.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.beginFill(0xff0000); 3. g.graphics.drawRect(0,0,30,50); 4. g.graphics.endFill(); |

**1.4.2.6 绘制一个圆角矩形**

命令：drawRoundRect();

参数：drawRoundRect(**x**,**y**,**width,height,ellipseWidth,ellipseHeight**);

参数说明：

**x**: 相对于父显示对象注册点的水平位置

**y**: 相对于父显示对象注册点的垂直位置

**width:**矩形的宽度

**height:**矩形的高度

**ellipseWidth**:绘制圆角的宽度.

**ellipseHeight**:绘制圆角的高度.

**实例:**

|  |
| --- |
| 1. var g = $.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.beginFill(0xff0000); 3. g.graphics.drawRoundRect(0,0,30,50,10,10); 4. g.graphics.endFill(); |

至此关于绘制的图像也基本到了一段落.这里需要说明的是在高级弹幕设置参数中都是可以先前设定的变量代替参数.

例：

|  |
| --- |
| 1. var width = 30;var g = $.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.beginFill(0xff0000); 3. g.graphics.drawRoundRect(0,0,width,50,10,10); 4. g.graphics.endFill(); |

这样写是完全没问题.文本对象也是完全可以这么使用的.

**至此高级弹幕里最最基础的部分也已经告了一段落.接下来的教程中讲重点讲创建后具体使用方面的知识.怎么运用其他工具让高级弹幕更上一层楼**

**第二章 使用其他工具**

**2.1Utils 工具库**

使用命令：

**hue**：将0-360的值映射到色相环上

**rgb**：将RGB值映射到色彩值上

**formatTimes**：格式化播放时间

**delay**：延迟执行函数

**interval**：定时执行函数

**distance**：计算座标距离

**rand**：rand返回一个伪随机数 n，其中 min <= n < max。

**2.1.1hue将0-360的值映射到色相环上**

命令：Utils.hue();

参数：Utils.hue(v);

参数说明：

**v**: 取值范围:一个正整数. 0-360,如果超过360将自动换成除以360后的余数.例361等同与1；

例：

|  |
| --- |
| 1. var v = 140;var color = Utils.hue(v);var g = $.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.beginFill(color); 3. g.graphics.drawCircle(0,0,30); 4. g.graphics.endFill(); |

**2.1.2 rgb将RGB值映射到色彩值上**

命令：Utils.rgb();

参数：Utils.rgb(r,g,b);

参数说明：

**r**: RGB中红色的值.取值范围:一个正整数,0-255,如果超过255将自动换成除以255后的余数.

**g**: RGB中绿色的值.同上

**b:** RGB中蓝色的值.同上

|  |
| --- |
| 1. var color = Utils.rgb(255,255,255);var g = $.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.beginFill(color); 3. g.graphics.drawCircle(0,0,30); 4. g.graphics.endFill(); |

**2.1.3 formatTimes格式化播放时间**

命令：Utils.formatTimes ();

参数：Utils.formatTimes (time);

参数说明：

**time**: 一个数字.格式化后将变成 00:00的文本形式.

例：

|  |
| --- |
| 1. var t = Utils.formatTimes(Player.time/1000); 2. var c = "视频时间："+t; 3. var g = $.createComment(c,{x:100,y:100,lifeTime:5}); |

**2.1.4 timer 延迟执行函数**

命令：timer ();

参数：timer (f,time);

参数说明：

**f**: 要延迟执行的函数.

**time**: 延迟执行的时间.以毫秒为单位.

例:

|  |
| --- |
| 1. timer(function(){ 2. var g = $.createShape({x:100,y:100,lifeTime:5}); 3. g.graphics.beginFill(0xffffff); 4. g.graphics.drawCircle(0,0,30); 5. g.graphics.endFill(); 6. },1000); |

**2.1.5 interval 定时重复执行函数**

命令：interval ();

参数：interval(f,time,times);

参数说明：

**f**: 要延迟执行的函数.

**time**: 延迟执行的时间.以毫秒为单位.

**times**:执行函数的次数.

例:

|  |
| --- |
| 1. var g = $.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.beginFill(0xffffff); 3. g.graphics.drawCircle(0,0,30); 4. g.graphics.endFill(); 5. interval(function(){g.x+=40;},1000,5); |

说明：每隔1秒绘制的圆向右移40像素.

**2.1.6 distance 计算坐标距离**

命令：Utils.distance ();

参数：Utils.distance(x1,y1,x2,y2);

参数说明：

**x1**: 计算起始坐标x轴

**y1**: 计算起始坐标y轴

**x2**: 计算结束坐标x轴

**y2**: 计算结束坐标y轴

|  |
| --- |
| 1. var d = Utils.distance(100,100,200,200); 2. var g = $.createShape({x:100,y:100,lifeTime:5}); 3. g.graphics.lineStyle(1, 0x0000FF, 1, false, "vertical","none", "miter", 10); 4. g.graphics.lineTo(0,d); 5. g.graphics.endFill(); |

创建新的Point（一个点）.

命令：$.createPoint();

参数：$.createPoint(x,y);

参数说明：

**x**: 坐标x轴

**y**: 坐标y轴

例：

|  |
| --- |
| 1. var p1 = $.createPoint (100,100);var p2 = $.createPoint (200,200); 2. var d = Utils.distance(p1.x,p1.y,p2.x,p2.y); 3. var g = $.createShape({x:100,y:100,lifeTime:5}); 4. g.graphics.lineStyle(1, 0x0000FF, 1, false, "vertical","none", "miter", 10); 5. g.graphics.lineTo(0,d); 6. g.graphics.endFill(); |

解释：创建2个新的Point对象并计算坐标距离.并绘制一条长为此距离的直线.

**2.1.7 rand 返回一个前闭后开的整数**

命令：Utils.rand ();

参数：Utils.rand(min,max);

参数说明：

**min**: 伪随机数最小值

**max**: 伪随机数最大值

例：

|  |
| --- |
| 1. var g = $.createShape({x:100,y:100,lifeTime:5}); 2. g.graphics.beginFill(0xffffff); 3. g.graphics.drawCircle(0,0,30); 4. g.graphics.endFill(); 5. interval(function(){g.x+=Utils.rand(30,500);},1000,5); |

说明：该弹幕绘制了一个圆,然后每隔1秒重复5次将该圆相对x轴位置移动到30-500中随机位置.

**2.2 Function 函数的创建与使用**

函数在高级弹幕中有着至关紧要的作用.在今后的运用中是一个必不可少的存在.

函数就是定义一套对应规则.使得集合A的元素对应到另一个集合B的唯一元素.

函数具有一下特性:

1.拥有输入和输出两组数据.

2.函数定义了一套对应规则，该规则是固定的,也就是函数的行为是固定的.

3.某一输出值拥有唯一的输出值.

我们下面直接来看函数语句在高级弹幕中的用法.

|  |
| --- |
| 1. function fname(pa1,pa2) 2. { 3. trace(pa1+pa2); 4. }; |

上面就是一个最基本的函数.在每函数语句中包含着3个要素:

**函数名**：**上例参数fname**

用来作为函数引用标识.

**参数**：**上例参数（pa1,pa2）**一组用小括号括起来的一组数据,参数之间用逗号分隔.函数可有空参数.

**函数体**：**上例参数{trace(pa1+pa2);}**

是实现函数功能的代码,代码用大括号括起来,在使用函数时将执行代码.

那么这个例子中的函数就是定义一个名为fname的函数.函数体是输出pa1+pa2.

另在函数的里面是可以创建函数的.但是在函数内创建的函数不能与外面的函数重名.

那么我们来看下面这个例子：

|  |
| --- |
| 1. function dm(txt,Vx,Vy,Vfontsize,Valpha) 2. { 3. var c = $.createComment(txt,{ 4. x:Vx, 5. y:Vy, 6. fontsize:Vfontsize, 7. alpha:Valpha 8. }); 9. }; 10. dm("1",100,100,20,1); 11. dm("2",150,150,20,1); 12. dm("3",200,200,20,1); |

上面就完整的写明了函数的创建到使用.这里使用函数更加简便的使用了创建文本弹幕.方便多次使用.

这里需要注意的是在函数里面创建的变量是不能在函数外面使用的.更多详细的内容会在以后说到.

**2.3 播放器控制**

使用命令:

**Player.seek();**

**Player.jump(,);**

**2.3.1视频时间跳转至**

命令：Player.seek();

参数：Player.seek(offset);

参数说明：

**offset:**要在视频文件中移动到的时间近似值（以毫秒为单位）**.**[**详细请看**](http://web.archive.org/web/20121113204940/http://wiki.bilibili.tv/wiki/Player#seek)**;**

例：

|  |
| --- |
| 1. Player.seek(1000); |

**2.3.2跳转到指定av号指定页视频**

命令：Player.jump();

参数：Player.jump(av,page,.newwindow);

参数说明：

**av:**要跳转视频的av号码.

**page:**要跳转视频的第几页.

**newwindow:**是否打开新窗口跳转值为 flase为不打开.值为true打开.不填写默认值打开

|  |
| --- |
| 1. Player.jump(“av120040”,1,flase); |

高级弹幕的学习到这里已经告了一段落.  
在这里只说明了基础的运用.有时间我会写进阶运用方法.  
如果有什么不懂的可以把问题发送到我邮箱[301082782@qq.com](http://web.archive.org/web/20121113204940/mailto:301082782@qq.com) .  
其实说实话我也是一个半吊子.教程可能有很多地方有问题.所以欢迎大家积极的指出来.  
  
**那么~高级弹幕初阶教程到此结束**

**本文完**