电工照明及仪表模拟接线教具的制作

彭科宏　谭尚伟
广州市交通运输职业学校

[摘　要]本文主要是介绍在flash软件中，如何合理运用as3.0语言来制作具有较强交互性的模拟教具，本文以本人制作的课件《电工考证之照明及仪表》中的“模拟接线”部分为例，分析了制作模拟教具的流程和方法。

[关键词]ActionScript.3.0；课件；模拟；教具
【DOI】10.12252/j.issn.2096-627X.2021.09.696

本人根据上课的需求，制作了《电工考证之照明及仪表》课件，其中的“模拟接线”部分就是按照实物设备来制作的模拟教具，主要是可以让教师方便地讲解电工考证中的接线方法，也先让学生独立地操作示范进行模拟练习，大大提高了上课的效率。模拟教具制作开始前，需要做很多的准备，比如布局设计、颜色搭配等方面，特别是使用ActionScript.3.0编程时，可能需要花很长的时间来思考和多次试验才能实现想要的效果。

一、制作前的准备

（一）考查

在制作之前，先去电工实训室进行实际考查，记录“照明及仪表”工位上有哪些设备，比如有：灯泡、日光灯、镇流器、保险、单一开关和双联开关等。因为是要模拟制作，所以要记录好这些设备摆放的位置，跟实物一样，才方便学生理解，更容易掌握知识。

（二）绘图

为了后面的编程，需要把每个设备都要做成一个元件，包括有：1.按钮类：开关的两种状态，开和关的状态；圆形小孔，作为接线点；2.影片剪辑类：一般的设备，其中灯类有两个状态，亮和灭的状态；3.图形类：比较少用到。

（三）辅助

按照实物绘好元件后，还要再做几个辅助按钮，包括有：1.关闭按钮：关闭元件；2.原理图按钮：点击后弹出原理图；3.示范按钮：点击后显示接线示范；4.颜色按钮：用来连线时，选择导线的颜色；5.清除按钮：用来重置画板，清除所有的连线；

二、开始编程

我们编程使用的是ActionScript.3.0语言，简称AS3.0，是一种强大的面向对象编程语言，特别是其中文档类的使用，给我们的编程带来了许多的方便。

（一）建立文档类

在flash软件中，可以把代码写在时间轴上来进行编程，很方便，不过当工程比较浩大，代码较多，编程比较繁琐的时候，就会变得很混乱了，排错也会变得非常地困难，幸亏文档类可以帮我们解决这难题。

我们可以把文档类看成是“程序入口”，在某种意义上也可以当作“主类”。建立过程也很简单：1.在flash软件中新建一个“ActionScript　3.0类”文档；2.保存此文档到程序的根目录中；3.在新建程序的舞台属性中，在“发布”选项中找到“类”文本框，输入保存的类文件名。

当课件中存在多个不同的模拟教具，我们可以把这些模拟教具分别做成不同的元件，然后为每个元件都建一个文档类，这样就不会混乱了。元件中链接文档类也很简单：1.在库中显示元件；2.右键打开元件属性；3.在“ActionScript链接”的“类”文本框中输入类文件名；

（二）文档类的一般框架

编程总会有个开始的过程，一开始可能是无从下手的，不可能一步到位，需要我们把它“模块化”，就跟建房子一样，一砖一瓦地添加，房子就在不知不觉中建好了！一开始先把框架写下来，然后再慢慢地往里面添加内容。

（三）事件处理的侦听

交互性强的课件，是离不开事件的，比如点击按钮，就要调用“鼠标事件”；在课件中画图、连线，就要调用“载入帧事件”等。要想调用这些事件，就需要为事件对象添加侦听器，并且调用事件的处理函数，最后在自定义的处理函数中添加处理的方法，并且根据实际情况移除侦听器。可以说我们的程序就是由很多的侦听器和自定义函数组成的。

1.侦听器。侦听器通过调用“addEventListener”函数为某事件指定一个侦听器，并确定好要侦听的事件名称及要执行处理的函数名称，以课件中的“模拟接线”为例，在主界面点击“模拟接线”按钮后，实现弹出“模拟接线”元件，然后在这个元件中来进行模拟接线，那么我们就需要为主界面中的“模拟接线”按钮添加一个侦听器，侦听鼠标对此按钮的操作，并进行相应的处理，格式如下：

neirong1_mc.mnjx_btn.addEventListener(MouseEvent.MOUSE_DOWN,mnjx3_1);

“mnjx_btn”是在主界面中影片剪辑元件“neirong1_mc”中的一个按钮元件，这里添加了对这个按钮的“MouseEvent.MOUSE_DOWN”即“按下鼠标”事件，并且这个事件的处理函数是“mnjx3_1”，然后在此函数中添加实现的方法。

2.自定义函数。添加侦听器后，就要新建侦听器中自定义的事件处理函数，格式如下：

privatefunctionmnjx3_1(event:MouseEvent){　}

其中“event:MouseEvent”代表着此函数接收的事件是“鼠标事件”，最后在此函数中添加处理事件的方法。

三、带弹性效果的等比例放大显示

课件中一些动态效果和一些交互，简单点分析，其实就是一种运动，一种二维的运动，当然也可以令二维运动体现出三维的效果。我们可以把现实中的一些运动，在flash中编写代码把它模拟出来，比如说在重力作用下的运动、有摩擦力的运动、缓动、弹性的运动等等。

把那些运动变成代码，无非就是按一定的规律来改变对象的坐标。坐标的变化，也就是X、Y、Z轴的偏移，对象就会根据坐标轴的偏移量来重新确定自己的坐标，也就使对象的位置产生了变化，对象也就在我们的眼中“动”起来了。所以实现运动的最根本的要素，就是坐标轴的偏移量，也就是向量。通过把对象的坐标加上按一定规律计算出来的向量，就可以实现对象的上下左右运动了。

在案例课件中，通过点击按钮把“模拟接线”元件调出来非常地简单，可以直接用元件“visible”属性来实现，但是太过于死板。在案例中，对“模拟接线”的显示进行了一个弹性效果的处理，看起来炫了很多，其中用到的方法如下：

（一）先在主函数“init”函数中添加一个侦听器

this.addEventListener(Event.ENTER_FRAME,mnjxEnterframe);

这里用到了“ENTER_FRAME”也就是“载入帧事件”，元件的属性不断地产生变化，就需要不断地刷新屏幕，才能显示出具有最新属性的元件。

（二）在事件处理函数“mnjxEnterframe”中加入计算的方法

关键方法如下：

```
varwidth_dx:Number=target_width-this.width;
varwidth_ax:Number=width_dx*spring;
width_vx+=width_ax;
width_vx*=friction;
this.width+=width_vx;
```

这里主要是计算出元件宽度变化的速度向量“width_vx”，其中“target_width”是元件最终的宽度大小，也就是目标宽度；“spring”定义为弹性系数，值是0到1的范围，示例课件中的值为0.1；“friction”定义为摩擦力系数，值也是0到1的范围，示例课件中的值为0.75；通过分别计算“width_ax”和“width_dx”而得出最终的向量“width_vx”，把元件的宽度不断地加上这个向量，就动态地实现了宽度的放大。这里还需要添加缩放的停止代码，移除掉侦听器，要不然缩放运动会一直地继续下去。

上面只是实现把元件的宽度带有弹性效果地放大，接着把元件的高度也这样处理，就可以实现带有弹性效果的等比例放大效果了。至此我们的“模拟接线”元件被调了出来，剩下的就是要在这个元件上绘制直线，实现模拟接线了。

四、绘图板的设计思路

模拟接线中的接线部分，是课件中的最关键环节。我们在“模拟接线”这个元件上，为里面的每个设备都配了相应的接线点，其实就是一个按钮元件，为不同的接线点取不同的实例名称。并且为这些接线点添加了侦听器，侦听鼠标的点击事件，设计思路如下：

1.鼠标点击了第一个接线点后，这个点我们称之为原点，就从原点到鼠标之间画一条直线，并会跟随着鼠标位置的变化而重新绘制，并且移除对此元件的侦听，不能进行二次点击，效果如图1：

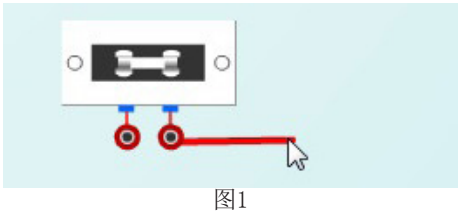

图1

2.如果期间在除接线点外的其他地方单击了鼠标后，原点就变成了鼠标单击的坐标点，并在新原点和旧原点之间，画一条直线，此操作可以进行多次，并把画的直线实例化，转换成影片剪辑元件，并为之添加侦听器，方便以后进行删除，效果如图2：

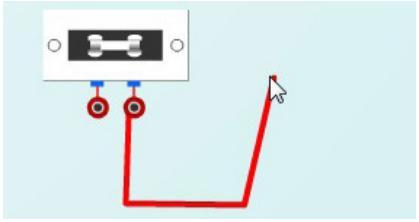

图2

3.直到点击了第二个接线点后，从新原点到第二个接线点间，画一条直线，并放入之前的实例中，至此连线结束，效果如图3：

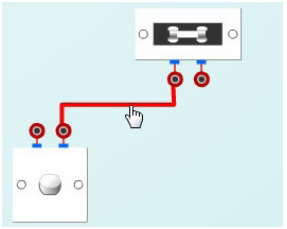

图3

五、绘图板的编程

（一）编程重点

直线的绘制是重点，此处主要调用了两个绘制函数，第一个函数是事件处理函数，用来绘制动态的直线，直线的长度、位置会跟着鼠标的变化而改变，如下：

```
privatefunctionmnjx3_2_linefun（event:Event）:void{……}
```

第二个函数是自定义函数，用来绘制固定的直线，直线的长度、位置不会产生变化，并且直线是画在了新建的元件“line”中，每一条连续的线都是一个元件，方便之后进行删除，如下：

```
privatefunctionmnjx3_2_linefun2(…相应的参数…){……}
```

在绘制直线过程中，舞台侦听了鼠标的点击事件，当点击了除接线点外的地方后，调用事件处理函数“mnjx3mousedown”，并在此函数中调用第二个绘制直线的函数“mnjx3_2_linefun2”，而且把鼠标点击的位置赋给原点，变成一个新的原点，最后继续调用第一个绘制直线的函数“mnjx3_2_linefun”，用来绘制动态的跟随鼠标的直线，代码如下：

```
privatefunctionmnjx3mousedown(event:Event):void{
            mnjx3_2_linefun2(mnjx3ballx,mnjx3bally
,mouseX,mouseY,mnjx3line);
    mnjx3ballx=mouseX;
    mnjx3bally=mouseY;
}
```

（二）编程关键

还有一个关键的问题，就是应该什么时候停止绘制直线呢？按照我们的思路，应该是在点击了两个接线点后，就要停止绘制了，变成代码的话，就要设置一个开关变量“mnjx3kg”，默认值为0，当点击一个接线点后就会加1：“mnjx3kg+=1”，然后我们再来一个判断，对变量“mnjx3kg”进行一个取模运算“mnjx3kg%2”，如果值不等于0，那么代表着“mnjx3kg”值为1，说明鼠标只点击了一次接线点；如果值等于0，那么就代表着“mnjx3kg”值为2，说明鼠标点击了两次接线点，应该要停止绘制了。这个开关变量应该在接线点的鼠标点击事件的处理函数中加入，接线点的点击事件处理函数关键代码如下：

```
privatefunctionmnjx3ballDown(event:MouseEvent){
mnjx3kg+=1;
if(mnjx3kg%2!=0){
…….
}else{
……
}
}
```

六、总结

编写代码时，思路必须得清晰，我们制作时是按照“先想、再试、后写”的思路进行的，制作前先想一想，有了方法后再来试验，方法不行，再来换其他的方法，最后实现了，再来完成我们的制作。当编写出现困难时，应该适当地停止制作，隔一段时间再继续，在这段时间内可以慢慢地想想原因，或者想一下有没有其他的方法来实现。总之课件的制作是不可能一步到位地，需要我们有足够的耐心和细心，特别是代码的编写，要经过反复的试验、尝试，才能最终实现我们所要求的效果。

参考文献：

[1]KeithPeters. 翻译.王汝义. FlashActionScript3.0动画教程[M]. 人民邮电. 2008-04

[2]陈冰. Flash第一步——ActionScript编程篇[M]. 清华大学出版社. 2006