网页设计系列

Macromedia 大师群英会

Flash 经典作品解析

Flash Flash Flash Flash Flash

Eric E. Dolecki 等 著
精英科技 译

- 本书由美国多位顶级 Flash 艺术家联合编著，畅销全球。
- 全书分为 9 章，每章以经典的 Flash 原创作品为主线，用来演示并讨论一个相关的 Flash 高级主题，并指出实例制作中隐含的关键概念和技术。
- 本书引领新的创意思路，代表更新的、更高级的技术，是 Flash 爱好者必备用书。

光盘内含全书范例素材文件、最终动画效果、效果演示文件

macromedia PRESS
中国电力出版社
www.infopower.com.cn

Macromedia 大师群英会

Flash

Flash Flash Flash Flash Flash

经典作品解析

Eric E. Dolecki
Michael Grundvig
Klaus L. Hougesen
Allan Kennedy
Jobe Makar
Til Mauder
Torben Nielsen
Max Oshman
Robert(son)Ramirez
Oliver Shaw
Geoff Stearns
Michael Brandon Williams 著

精英科技 译

中国电力出版社

内 容 提 要

Flash 自其出现以来，已经风靡全球。目前市场上介绍 Flash 的书籍很多，但本书不是改头换面重谈大家早已熟知的陈旧话题。相反，本书着眼于高级 Flash 开发、设计技巧，通过多个经典实例展示了 Flash 编程和实现的新概念。本书涵盖的主题包括：Flash 三维动画、Flash 物理学、Flash 声音编程、组件系统结构、动态 Flash 页面、XML 以及 JavaScript 和 Flash 的交互。

本书适合希望在原有基础上，研究拓展新的高级技术的 Flash 高级开发人员阅读。

图书在版编目（CIP）数据

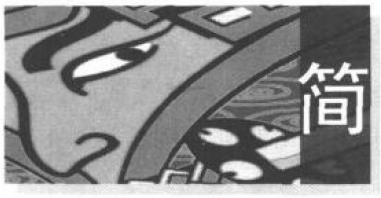
Flash 经典作品解析/（美）道莱克等著；精英科技译. —北京：中国电力出版社，2002.9
ISBN 7-5083-1280-5

Ⅰ.F… Ⅱ.①道…②精… Ⅲ. 动画-设计-图形软件，Flash Ⅳ.TP391.41

中国版本图书馆 CIP 数据核字（2002）第 064512 号

著作权合同登记号 图字：01-2002-6207 号

中国电力出版社出版、发行
（北京三里河路 6 号 100044 http://www.infopower.com.cn）
汇鑫印务有限公司印刷
各地新华书店经售
*
2003 年 1 月第一版 2003 年 1 月北京第一次印刷
787 毫米×1092 毫米 16 开本 20.25 印张 457 千字
定价 **38.00** 元



简　介

欢迎阅读本书

和很多 Flash 书不同，这本书不是改头换面重谈早已熟知的陈旧话题。相反，我们的目的是展现 Flash 编程和实现的新概念，帮助你拓展那些熟悉的概念。通过研究高级界面、XML、Flash 物理等主题，我们不仅希望能提供给你新的思路，更重要的是启发你的灵感和激情，探索开发属于你自己的技术。

本书的由来

创作本书的动机源自 Figleaf 软件公司的一个名为 Flashcoder（Flash 编码者）的高级 Flash 邮件组（http://chattyfig.figleaf.com）。（幕后人物 Branden Hall，出于好心做了这个邮件组，因为他喜欢用 ActionScript 编程。）该邮件组是高级 Flash 程序员交换有关 Flash 开发的信息，特别是 ActionScript 编码的公共场所。

这个邮件组建立了几个月后，我意识到应该收集整理一下讨论信息，留作为参考资料。起初，我想建一个数据库，但我觉得查询/返回的处理模式不能很好地展示信息。我们这群 Flash 程序员为何不一起创作一个集中的信息资源，特别是，一本书？

我的想法得到了很多人的积极响应。当我了解了这些人的技术专长时，吓了一跳。这么多天才，挑都挑不过来。

最后选好了团队，我们花了几个月的时间确定了内容提纲。我们在一个名为超级高手的在线消息板系统上工作（团队初始成员 Bob Clagget 创建）。

本书的适用范围

这本书适用于希望在原有基础上，研究拓展新的高级技术的 Flash 高级开发人员。本书假定读者应熟悉 ActionScript 编程、了解本书某些章节讨论的其他技术，如 XML 和 SQL。

如何使用这本书

这本书分为 9 章，每章讨论一个 Flash5 高级主题。各章相互独立，你可以按照任意顺

序阅读。除了第 2 章和第 7 章由两位作者合作完成，其他每一章的作者都不相同。

每一章的核心内容是开发实例——原创的 Flash 作品，用来演示本章的主题。每位作者都将带你学习他的例子，指出其中隐含的关键概念和技术。其中的一些更有魄力的作者创建了多个例子，还有些作者制作了许多小的演示程序，帮助导出示例。

所有的资源文件都包含在随书附赠的光盘上。希望你能研究这些文件，修改代码，看看结果如何。光盘上还包括了 Flash 的演示版以及一些没有在书中讨论的其他例子。

图标含义


光盘（CD-ROM） 图标出现在作者引用光盘上的文件时。


关键概念（Key concept） 该图标出现在某个概念对于这章介绍的主题特别重要时。


提示（Tip） 无论在什么地方看到这个图标，你都会得到好建议，从而使 Flash 变得更容易一些的。


注意（Warning） 这个图标提醒你注意潜在的问题。

每一章都有什么内容?

下面是每章内容纲要。

第 1 章：向 Flash 注入生命气息

这一章着眼于在 Flash 项目中编写逼真运动代码的方法。介绍了一些新的编码和设计技术，使生物爬行和移动。

第 2 章：Flash 三维动画

越来越多的 Flash 开发者开始理解 3D 的威力。这一章分为 2 部分，分别讨论 2 种差别很大的创建三维动画和图形的方法：ActionScript 3D（脚本 3D）和 Rendered 3D（绘制 3D）。

第 3 章：Flash 物理学

本章向你展示如何运用 Flash 物理学编写 ActionScript 脚本，创建现实运动。包括直线运动、引力加速度运动、简单物体碰撞及碰撞反应。

第 4 章：Flash 声音编程

有效的利用声音，可以使动画序列、交互式游戏以及其他 Flash 作品变得更加丰富生动。本章重点讨论使用 Flash 声音对象的全新编程技术。

第 5 章：组件系统结构

按照组件系统结构创建的网站页面加载更快、更新更容易。本章介绍如何计划和建设一个可快速载入的基于组件的系统，以及如何预防影响 Flash 网站性能的可用性问题。

第 6 章：创建动态 Flash 页面

介绍如何结合服务器端脚本语言和 ActionScript 脚本语言，创建动态 Flash 网站。本章解释了所需的相关编码技术，并向你展示应用这些技术的 3 个实际应用程序的脚本，你可以将它们用在任何网站上。

第 7 章：Flash 界面设计

介绍如何使用窗口常用界面元素，如下拉菜单和超级链接，让 Flash 站点更协调、更有效。本章向你展示如何制作和使用窗口实现精细的界面效果，从而使观众留下令人难忘的体验。

第 8 章：XML 和 Flash

XML 和 Flash 的组合可以设计出意想不到的交互式的、自适应的前端应用程序。本章提供了运用 XML 创建可重用、可理解代码的方法，简化和加速程序过程，甚至使错误最少。

第 9 章：Flash 和 JavaScript 交互

运用 JavaScript 的基本知识，你可以极大地拓展 Flash 工具、插件和函数功能。本章提供了现实的例子和专家意见，通过 Flash 和 JavaScript 集成，改变浏览器的效果。

我希望这些描述能够激发你的阅读欲望。有一点可以肯定：即使你只学到了本书三分之一的内容，你就能制作令人惊异的 Flash 网站。

后续

当这本书上架的时候，我们将建成自己的网站（www.supersamuraiflash.com）。在那儿，你将找到更多实用的 Flash 电影、书的更新、已经发现的错误更正（不是全部），也可以发表你的观点。

希望你能喜欢这本书。

Flash 用户和数据库顾问

Robertson Ramirez

于 Queens, New York

目 录

第1章

向 Flash 注入生命气息

Geoff Stearns，来自亚利桑那州图森市，是一名 Flash 自由开发者和培训者。业余时间，他维护了一个网站 www.deconcept.com，用于探索实践 Flash 的极限和新兴网络技术。

在这一章，我们将采用不同的方法，在你的 flash 设计中加入逼真的动作。我们也将探索关于可编程动作（programmatic movement）的一些基本概念。使用可编程动作，你就可以随时随地移动 flash 电影里的物体。

为了解释本章的概念，我和你一起浏览一下我的网站 www.deconcept.com 上的两个实例。我将解释我是怎样设计和编程实现它们的，更重要的是，我将解释为什么要选择这样做，而不是采用其他的方法。

当你正在学习一门新的程序语言时，编码技术可能是最难学的事情之一。Flash，作为这样一种广泛的程序语言，只是增加了挑战性——大多数目标都可以通过多种技术途径实现。通过解释我的代码布局和讨论为何选取某些特定技术，我希望能教会你如何更有效地应用 ActionScript 脚本语言。

创建一只交互式的蜘蛛

我的第一个设计是一只交互式的蜘蛛。通过测试源文件，你将对如何使用 ActionScript 创建逼真的行为和动作，如何更有效地使用剪辑事件（clip events）有一个大概的认识。

打开光盘上的 spider.swf。

蜘蛛身体上有个透明的按钮，你可以拖着它在屏幕上移动。试验一下，看看如何工作（图 1.1）。

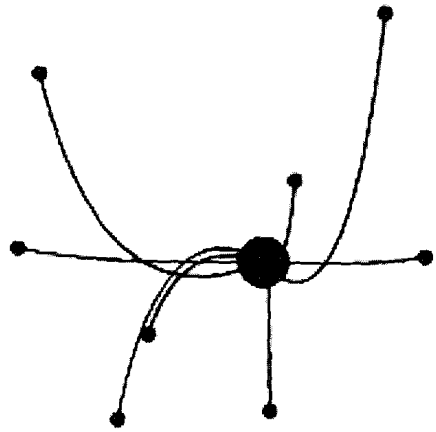


图 1.1　抓住蜘蛛的身体并拖动它

建立蜘蛛源文件

打开光盘上的 spider.fla。

我建立的这个文件，大小为 500×300 像素，帧速率为 20 帧/秒。（如果设定更高的帧速率，蜘蛛腿将移动得更快，看上去会有点奇怪。）

为了使模块清晰，我把蜘蛛的所有部分都放在另一个电影剪辑里。这样无论我怎么拖动蜘蛛，它都会按照自己的方式运动。并且由于该电影没有中间过程(tween)，主时间线(main timeline）只包含一帧和两层，一层用于蜘蛛本身，一层用于所有的动作。把所有的动作都放在一个层里，搜索和编辑更容易一些。

在动作层（Actions layer）的第一帧，我设置了如下的代码：

```
fscommand ("allowscale",false);
stop();
```

第一行代码，避免在测试期间，缩放和拉伸电影画面，在大多数我的电影里都用了这行代码；Stop()动作阻止电影循环播放。

我编程时采用专家模式（Expert mode），因为这种模式书写更快更方便。我希望你也用这种模式，即使你不太适应在 ActionScript 窗口上使用它，或者你记不住所有的句法。如果你忘记了一个命令或者句法，那么切换到正常模式（Normal mode），用 ActionScript 菜单把命令写到窗口上，然后返回专家模式，填写变量。把 Flash 5 ActionScript 参考指南放在手边，以便查找不常用的命令。

创建蜘蛛身体各部分

蜘蛛身体各部分非常简单：圆圆的身体被 8 条腿围绕着，每条腿的末端有脚趾。当蜘蛛的基本形状——圆和线条画好后，还需要做更多幕后工作。在这一小节，我们简要讨论蜘蛛身体的每个部分，并指出创建它们的乐趣所在。

身体

为了制作蜘蛛的身体，我创建了一个名为 m-body 的电影剪辑。这个电影剪辑有 2 层：一幅身体的图像和一个用来使蜘蛛可交互的透明按钮。

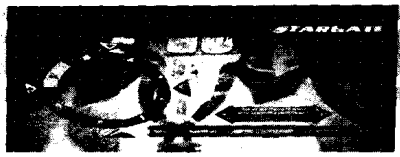
看一下库（library），注意蜘蛛身体的图像称为 g-body，按钮称为 b-body。Flash 不允许在库中重名，因此把一个字母放在单词“body”前面，可以重复使用这个单词。同时，这种命名方法也能快速直观地确定一个符号的用处。

脚趾

我使用剪辑事件让脚趾运动。Flash 5 只能把剪辑事件加入到舞台（stage）上的电影剪辑中。因此我创建了一个电影剪辑容器，把 m-toe 电影剪辑放在里面，然后把剪辑事件放在 m-toe 电影剪辑上。建好一个脚趾后，用 attachMovie()方法从库中调用该脚趾 8 次。

腿

我需要创建 8 条腿，所以用 attachMovie()方法更容易，我只要从库里抓出腿，把它们放到舞台上任何我想要放的地方就可以了。如果用了 duplicateMovieClip()方法，则必须把一条腿放在舞台上计划创建它的地方。



因为 attachMovie()只能从库中抓取电影剪辑，所以蜘蛛腿必须是电影剪辑。然而实践中，赋予所有原始图片实际的图片符号，比直接把它们拖曳到电影剪辑中更好。

为了绘制表示蜘蛛脚的线条，我创建了一条 45 度角的线条，其尺寸是 100×100 像素，然后在两点之间拉伸线条。（线条的左上角应该位于电影剪辑片段所在的坐标系的原点。）我给线条做了一个小小的弯曲度，这样它看上去就更好看了。

让蜘蛛动起来

现在我们已经完成了蜘蛛身体的各部分，让我们回顾一下使蜘蛛运动的代码。

脚趾运动

当用户拖着蜘蛛的身体到屏幕上的新位置时，腿和脚趾必须跟着动。让脚趾运动相当简单，我用剪辑事件和一些运动的代码让脚趾滑到新的位置，同时还增加了一些代码来保存每个脚趾相对于身体的位移，使得每个脚趾所处的区域，相对于初始位置不变：身体下面的脚趾仍然在身体下面，身体上方的或者左边的脚趾仍然分别在身体上方或者左边。

一个很好的经验是把代码写在各自对应的地方，而不是都写在一起。这样如果出了什么问题，更容易找到出错的部分。

下面是 m-toe 电影剪辑中的代码：

```
onClipEvent (load) {
 // -- set up variables
 // -- "a" controls how fast the toes move, a lower number
 →will make them slide faster
 var a = 2;
 var newX = _parent._x;
 var newY = _parent._y;
 var xSpeed = 0;
 var ySpeed = 0;
 // -- offset is the distance of this movie clip to
 →the body
 var xOffset = _parent._x -_parent._parent.body._x;
 var yOffset = _parent._y-_parent._parent.body._y;
}
 onClipEvent (enterFrame) {
 // -- Calculate the speed each frame
 xSpeed = (_parent._x-(newX+xOffset))/a;
 ySpeed = (_parent._y-(newY+yOffset))/a;
 // -- adjust parent movie clip position
```

```
_parent._x -= xSpeed;
_parent._y -= ySpeed;
}
```

现在让我们回顾一下告诉脚趾何时以及向何地运动的代码。

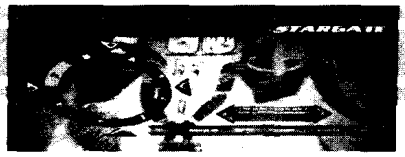
操纵脚趾

用户不拖动蜘蛛，蜘蛛是不会动的。一旦用户拖动了蜘蛛，脚趾应该紧跟着身体移动。为此，在身体上设立一个剪辑事件，检查每个脚趾到身体的距离。基于变量 WalkDist 随机生成一个距离，如果超过了该距离，脚趾就会移动到靠近身体的新的随机位置。

同样，希望各个脚趾相对于身体的位置不变。换句话说，蜘蛛移动后，身体下面的脚趾还在身体下面（或者至少很接近），身体最上面的脚趾仍然在最上方附近。这样蜘蛛能够保持相对平衡，不至于所有的脚趾都跑到身体一侧去。

m-body 剪辑的代码如下：

```
onClipEvent (load) {
 // -- first swap the depths of the body to place it above
 →the leg lines (This is only a cosmetic change)
 this.swapDepths(1000);
 // -- set up the variables
 // -- walkDist is used as a base number to keep the toes
 →close to thebody
 // -- the higher this number is, the further the legs will
 →be allowed to travel
 // -- from the body
 var walkDist = 60;
 var maxDist = 0;
 // -- this tells the body how many legs we have
 var toes = 8;
 // -- attach the leg lines from the library and name them
 for (i=1; i<=toes; i++) {
  _parent.attachMovie("leg", "leg"+i, i+100);
 }
}
onClipEvent (enterFrame) {
 // -- loop through each toe and check its distance from
 →the body
 for (i=1; i<=toes; i++) {
  myToe = _parent["toe"+i];
  xDist = Math.abs(myToe.toe.newX-this._x);
  yDist = Math.abs(myToe.toe.newY-this._y);
```




```
maxDist = Math.floor(Math.random()*walkDist)+walkDist;
// -- if the distance is farther than a certain amount,
→tell that toe to move
if (xDist>maxDist || yDist>maxDist) {
 myToe.toe.newX =
 →this._x+Math.floor(Math.random()*walkDist)-walkDist/2;
 myToe.toe.newY =
 →this._y+Math.floor(Math.random()*walkDist)-walkDist/2;
}
// -- adjust the "legs" each cycle (the lines)
myLeg = _parent["leg"+i];
myLeg._x = this._x;
myLeg._y = this._y;
myLeg._xscale = myToe._x-this._x;
myLeg._yscale = myToe._y-this._y;
}
}
```

这段代码工作原理如下：

当电影开始时，身体代码连续 8 次从库中取出腿的电影剪辑。无论何时开始进入一帧，蜘蛛都循环检测每个脚趾离身体的距离。如果该距离大于变量 maxDist，代码为每个脚趾选择随机_x 和_y 位置。然后代码更新每条腿的位置。

让蜘蛛可交互

为了能够拖动蜘蛛，我给蜘蛛身体上的按钮添加了动作。下面是 m-body 电影剪辑中按钮的代码：

```
on (press) {
 this.startDrag();
}
on (release) {
 stopDrag();
}
```

这就完成了蜘蛛的运动。现在让我们看一下行走的昆虫。

创建行走的昆虫

我的第二个作品是一只正在行走的昆虫。测验源文件，你将对可编程运动、通过

ActionScript 控制多个物体和编写一般的代码有更进一步的认识。

打开光盘上的 walking_insect.swf 文件，看看动画如何工作（图 1.2）。尽管电影中有 3 只昆虫，我只讨论如何创建它们中的一只。

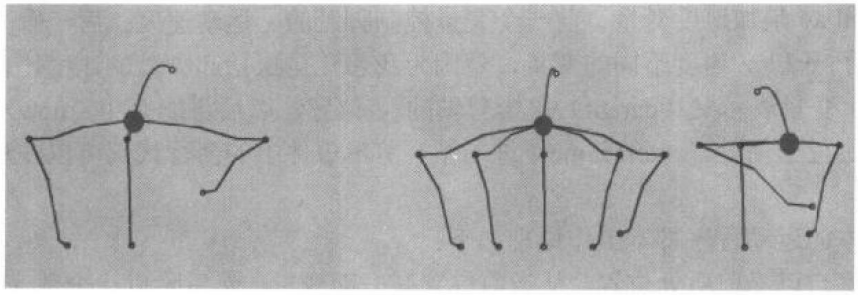


图 1.2　动作中的行走的昆虫

当昆虫第一次登上舞台时，随机选取昆虫腿的条数。然后产生脚，并开始四处走动，昆虫有 3 种状态：向前走，向后走，静止不动。

创建昆虫身体各部分

打开光盘上的 walking_insect.fla 文件。

该文件设置为 750×285 像素，帧速率为 20 帧/秒。

昆虫头上有触角，上面有一个小圆点。昆虫腿由线条构成，从头一直延伸到膝盖，再从膝盖一直到脚趾。膝盖和脚趾都用小圆点表示。

和创建蜘蛛一样，我把昆虫的代码分别放在它的各相关身体部位上。这样，如果昆虫不能正确工作，容易精确定位错误代码。

脚趾

在库中，找到 m-toe holder（持有）电影剪辑（在名为 toes-knees-head dot 的文件夹内）。这里面有一个名为 m-small dot 的小圆点图片，该图片本身是一个内含 g-small dot 图片的电影剪辑。我用 m-small dot 电影剪辑控制昆虫的膝盖、脚趾和触角。每次我把它放入 container（容器）电影剪辑，并根据需要，给它加上不同的代码。

下面是 m-toe holder 电影剪辑的代码：

```
onClipEvent (load) {
 var a = 3;
 var ay = 2;
 oYpos = _parent._y;
 newX = _parent._x;
 newY = _parent._y;
}
```

```
onClipEvent (enterFrame) {
 _parent._x -= (_parent._x-newX)/a;
 _parent._y -= (_parent._y-newY)/ay;
}
```

变量 a 和 ay 是加速度变量。前一个变量控制脚趾的 x 运动速度；后一个变量控制 y 运动速度。赋予 x 和 y 速度不同的变量，是因为我想让昆虫抬起脚趾的速度比左右移动的速度快。oYpos 变量表示父（parent）电影剪辑的_y 位置，以后可能有用。newX 和 newY 变量告诉脚趾定位于何处。enterFrame（进入帧）剪辑事件中的两行代码可以在舞台上四处移动剪辑。

这段基本的运动程序脚本工作原理如下：

脚趾观察自己当前所处位置，从当前位置减去新变量，然后除以一个数 a，以此来更新位置。因为在每帧都要执行这个过程，因此如果我想把脚趾移到一个新位置，我所要做的一切就是修改 newX 和 newY 变量，这样脚趾就会滑到它的新位置。

膝盖

m-knee holder 电影剪辑的代码如下：

```
onClipEvent (load) {
 a = 1.5;
 d = 1.7;
 newX = _parent._x;
}
onClipEvent (enterFrame) {
 newX = (_parent._parent["toe"+_parent.num]._x-
 →_parent._parent.head._x)*1.5+_parent._parent.head._x;
 xSpeed = ((_parent._x - newX)/a+xSpeed)/d;
 _parent._x -= xSpeed;
}
```

你可以看出，我用了大体相同，只是细微处稍有区别的运动代码来实现不同的行为。为了给膝盖的运动增加一点逼真的弹跳，我在代码中增加了变量 d。同样，请注意我只更改了膝盖的 x 位置。当膝盖加入昆虫体时，头部代码设置它们的 y 位置。因为 y 位置总是相同的，所以没有必要更新。

enterFrame 剪辑事件第一行让每个膝盖和其相关的脚趾保持一定的偏移距离。当把每个脚趾和膝盖（toe1、toe2、knee1、knee2 等等）加到头上时，给它们分配一个数 num。所以，在每一帧里，每个膝盖都检查和它具有相同 num 值的脚趾的 x 位置，并根据该脚趾的位置相应更新自己的位置。

触角

现在我们来看 m-head dot holder 电影剪辑，它是最后一个调用 m-small dot 剪辑的电影

剪辑。它的代码如下：

```
onClipEvent (load) {
 a = 6;
 d = 1.04;
 _parent._parent.attachMovie("headline", "headline", 5);
}
onClipEvent (enterFrame) {
 newX = _parent._parent.head._x;
 newY = _parent._parent.head._y-30;
 myXSpeed = ((_parent._x-newX)/a+myXSpeed)/d;
 myYSpeed = ((_parent._y-newY)/a+myYSpeed)/d;
 _parent._x -= myXSpeed;
 _parent._y -= myYSpeed;
}
```

这里，我再次便用了和前面例子相同的描述运动的代码，只有少量变动。通过改变变量 d 的值，让触角圆点在一个特定的点附近摇摆。该点的 newX 和 newY 变量使其在 x 方向上和昆虫的头对齐，y 方向在头上方 30 个像素。运动代码考虑到了昆虫的休息——我们没必要着急更新圆点的位置，因为在每一帧，圆点自己都会做这项工作。

腿

进入库中的 lines（线条）文件夹。注意，有两种不同的 line（线条）电影剪辑：一个用于头部圆点（g-headline），一个用于腿（g-line）。把这两种线条都输出，这样，当想在电影中使用线条的时候，可以随时调用 attachMovie()方法。

让昆虫行走

找到库中的 m-walker 电影剪辑，观察头部电影剪辑（head movie clip）的动作。这儿有不少代码，所以我将逐段检查这些代码。

这些代码的基本思想是：加载电影时，主时间线第一帧的动作将 m-walker 电影剪辑加入舞台。该剪辑加入后，将触发剪辑事件中的动作。代码选取 3 到 5 之间的随机数以确定昆虫腿的条数。然后代码加入脚趾、膝盖和头，以及腿，并均匀地把脚趾分放在昆虫体的下方，用同样的方法来处理膝盖和头。

下面是该通用代码的一个示例，昆虫没有预设的腿数——腿数是在昆虫第一次载入的时候随机选取的。这种技术可以用于任何项目。如果创建昆虫时没有使用这种通用代码——例如我设定昆虫为 5 条腿——而客户需要 3 条腿，那么我必须手动从文件删除 2 条腿，然后重新导入。而如果使用了通用代码，我只要修改一个变量值就可以了。

加入剪辑后，enterFrame 剪辑事件开始工作。如果昆虫站立不动，该事件不做任何事情。