

Autodesk® Scaleform®

Scaleform CLIK用户指南

本文包括了Scaleform CLIK框架及所带内置组件执行方法的详细使用说明

作者: Prasad Silva, Matthew Doyle

版本: 2.0

最后修订: 2010年8月19号

Copyright Notice

Autodesk® Scaleform® 4.2

© 2012 Autodesk, Inc. All rights reserved. Except as otherwise permitted by Autodesk, Inc., this publication, or parts thereof, may not be reproduced in any form, by any method, for any purpose.

Certain materials included in this publication are reprinted with the permission of the copyright holder.

The following are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and other countries: 123D, 3ds Max, Algor, Alias, AliasStudio, ATC, AUGI, AutoCAD, AutoCAD Learning Assistance, AutoCAD LT, AutoCAD Simulator, AutoCAD SQL Extension, AutoCAD SQL Interface, Autodesk, Autodesk Homestyler, Autodesk Intent, Autodesk Inventor, Autodesk MapGuide, Autodesk Streamline, AutoLISP, AutoSketch, AutoSnap, AutoTrack, Backburner, Backdraft, Beast, Beast (design/logo) Built with ObjectARX (design/logo), Burn, Buzzsaw, CAiCE, CFdesign, Civil 3D, Cleaner, Cleaner Central, ClearScale, Colour Warper, Combustion, Communication Specification, Constructware, Content Explorer, Creative Bridge, Dancing Baby (image), DesignCenter, Design Doctor, Designer's Toolkit, DesignKids, DesignProf, DesignServer, DesignStudio, Design Web Format, Discreet, DWF, DWG, DWG (design/logo), DWG Extreme, DWG TrueConvert, DWG TrueView, DWFX, DXF, Ecotect, Evolver, Exposure, Extending the Design Team, Face Robot, FBX, Fempro, Fire, Flame, Flare, Flint, FMDesktop, Freewheel, GDX Driver, Green Building Studio, Heads-up Design, Heidi, Homestyler, HumanIK, i-drop, ImageModeler, iMOUT, Incinerator, Inferno, Instructables, Instructables (stylized robot design/logo), Inventor, Inventor LT, Kynapse, Kynogon, LandXplorer, Lustre, MatchMover, Maya, Mechanical Desktop, MIMI, Moldflow, Moldflow Plastics Advisers, Moldflow Plastics Insight, Moondust, MotionBuilder, Movimento, MPA, MPA (design/logo), MPI (design/logo), MPX, MPX (design/logo), Mudbox, Multi-Master Editing, Navisworks, ObjectARX, ObjectDBX, Opticore, Pipeplus, Pixlr, Pixlr-o-matic, PolarSnap, Powered with Autodesk Technology, Productstream, ProMaterials, RasterDWG, RealDWG, Real-time Roto, Recognize, Render Queue, Retimer, Reveal, Revit, RiverCAD, Robot, Scaleform, Scaleform GFx, Showcase, Show Me, ShowMotion, SketchBook, Smoke, Softimage, Sparks, SteeringWheels, Stitcher, Stone, StormNET, Tinkerbox, ToolClip, Topobase, Toxik, TrustedDWG, T-Splines, U-Vis, ViewCube, Visual, Visual LISP, Vtour, WaterNetworks, Wire, Wiretap, WiretapCentral, XSI.

All other brand names, product names or trademarks belong to their respective holders.

Disclaimer

THIS PUBLICATION AND THE INFORMATION CONTAINED HEREIN IS MADE AVAILABLE BY AUTODESK, INC. "AS IS." AUTODESK, INC. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE REGARDING THESE MATERIALS.

How to Contact Autodesk Scaleform:

Document	Scaleform 4.2 CLIK AS3 User Guide
Address	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
Website	www.scaleform.com
Email	info@scaleform.com
Direct	(301) 446-3200
Fax	(301) 446-3199

目 录

1 简介	i
1.1 概要	i
1.1.1 在Scaleform CLIK包含的内容	i
1.2 组件理解	ii
1.3 UI关注	ii
2 内置组件	iii
2.1 基本按钮和文本类型	iv
2.1.1 Button	v
2.1.2 CheckBox	x
2.1.3 Label	xiii
2.1.4 TextInput	xv
2.1.5 TextArea	xix
2.2 分组类型	xxii
2.2.1 RadioButton	xxii
2.2.2 ButtonGroup	xxvi
2.2.3 ButtonBar	xxviii
2.3 滚动类型	xxx
2.3.1 ScrollIndicator	xxx
2.3.2 ScrollBar	xxxii
2.3.3 Slider	xxxv
2.4 列表类型	xxxvi
2.4.1 NumericStepper	xxxviii
2.4.2 OptionStepper	xl
2.4.3 ListItemRenderer	xlii
2.4.4 ScrollingList	xlv
2.4.5 TileList	l
2.4.6 DropdownMenu	liv
2.5 进度类型	lx
2.5.1 StatusIndicator	lx
2.5.2 ProgressBar	lxii
2.6 其他类型	lxiv
2.6.1 Dialog	lxiv
2.6.2 UILoader	lxvi
2.6.3 ViewStack	lxvii

3 艺术细节	lxix
3.1 最优方法	lxix
3.1.1 像素图像	lxix
3.1.2 蒙版	lxx
3.1.3 动画	lxx
3.1.4 图层和绘制元素	lxx
3.1.5 复杂皮肤界面	lxxi
3.1.6 2的指数倍	lxxi
3.2 已知问题和推荐工作流程	lxxi
3.2.1 复制组件	lxxi
3.3 皮肤绘制实例	lxxiii
3.3.1 StatusIndicator皮肤绘制	lxxiv
3.4 字体和本地化	lxxvi
3.4.1 概要	lxxvi
3.4.2 内置字体	lxxvi
3.4.3 在textField嵌入字体	lxxvii
3.4.4 本地化系统	lxxvii
4 编程详述	lxxix
4.1 UI组件UIComponent	lxxx
4.1.1 初始化	lxxx
4.2 组件状态	lxxx
4.2.1 按钮组件	lxxxi
4.2.2 非按钮交互组件	lxxxii
4.2.3 非交互组件	lxxxiii
4.2.4 特殊案例	lxxxiii
4.3 事件模型	lxxxiii
4.3.1 最佳使用方法	lxxxiii
4.4 焦点处理	lxxxiv
4.4.1 最佳使用方法	lxxxiv
4.4.2 在复合附件捕获焦点	lxxxv
4.5 输入处理	lxxxvi
4.5.1 最佳使用方法	lxxxvi
4.5.2 多鼠标光标	lxxxviii
4.6 失效	lxxxix
4.6.1 最佳使用方法	lxxxix
4.7 组件缩放	lxxxix
4.7.1 Scale9Grid	xc

4.7.2	强制	xc
4.8	组件和数据设置	xcii
4.8.1	最佳使用方法	xcii
4.9	动态动画	xcii
4.9.1	最佳使用方法	xcii
4.10	弹出式支持	xciii
4.10.1	最佳使用方法	xciii
4.11	拖放	xciii
4.11.1	最佳使用方法	xciv
4.12	杂项	xcv
4.12.1	代理Delegate	xcv
4.12.2	本地Locale	xcv
5	实例	xcvi
5.1	基础	xcvi
5.1.1	包含两个textField的ListItem Renderer	xcvi
5.1.2	像素滚动视图	xcviii
5.2	合成	xcix
5.2.1	使用ScrollingList 的TreeView	xcix
5.2.2	可复用视窗	ci

1 简介

本文档提供了Scaleform® 通用精简接口工具（Common Lightweight Interface Kit，CLIK™）框架和组件的详细使用说明。在深入认识CLIK用户指南之前，希望开发者能够先阅读[CLIK入门](#)和[CLIK按钮入门](#)。这两个使用指南介绍了创建和运行Scaleform CLIK所需要的步骤，介绍了基本的概念并提供了创建和美化CLIK组件的详细指南。然而，专业级用户更喜欢在学习入门文档时直接查阅本文档作为参考。

1.1 概要

Scaleform CLIK为一组ActionScript™ 2.0 (AS2) 用户接口元素、库和工作流增强工具集， Scaleform 4.0用户为游戏控制器、PC和掌上游戏机应用快速实现丰富和高效的接口。框架的主要目标为构建一个精简（依据内存和CPU使用量）、易于绘制皮肤并高度个性化的架构。另外，对于一个基本的UI内核类和系统基本框架，CLIK包含了15个以上内置通用接口元素（例如，按钮、滚动条、文本输入框），将帮助开发者快速创建和重构用户接口界面。

1.1.1. 在Scaleform CLIK包含的内容

组件：简单扩展内置UI控制组件

Button	Slider
ButtonBar	StatusIndicator
CheckBox	ProgressBar
RadioButton	UILoader
Label	ScrollingList
TextInput	TileList
TextArea	DropdownMenu
ScrollIndicator	Dialog
ScrollBar	NumericStepper

类：系统核心 API函数

InputManager	Locale
FocusManager	Tween
DragManager	DataProvider
PopUpManager	IDataProvider
EventDispatcher	IList
Delegate	IListItemRenderer

文档和实例文件:

- CLIK入门
- CLIK按钮入门
- CLIK API 函数参考
- CLIK 用户指南
- CLIK Flash 实例
- CLIK 视频指南

1.2 组件理解

在开始之前，开发者需要理解Flash组件的确切技术细节。Flash的一系列默认接口创建工具和编译块我们称之为组件。但是，在本文档中“组件”指使用Scaleform CLIK框架创建的内置组件，这些组件由Scaleform 与世界知名的gskinner.com团队联合开发。

gskinner.com由Grant Skinner领导，Grant Skinner由Adobe任命负责为Flash Creative Suite® 4 (CS4)创建组件，为世界知名的Flash领先开发团队之一。关于gskinner.com的更多信息，请访问<http://gskinner.com/blog>。

为更深入理解内置CLIK组件，在Flash studio中开打默认的CLIK文件：

C:\Program Files\Scaleform\GFx 4.2\Resources\AS2\CLIK\components\CLIK_Components.fla

1.3 UI关注

创建一个美观的UI界面，第一步为在纸面或画图编辑器上如Microsoft Visio®上绘制草图。第二步为在Flash中开始原型化UI界面，其目的为在专门图形设计之前绘制出所有的界面项和流程图。Scaleform CLIK就是专门设计用于使用户快速原型化和重构以完成整个过程。

构建一个完整的UI界面有很多不同的方法。在Flash中，不存在页的概念，而在Visio或其他流程图绘制软件中有此概念，这里使用了关键帧和动画剪辑代替。如果所有的页面都在同一个Flash文件中，文件将占据更多内存，但是在页间切换更加快速且容易。如果每个页分别在单独的文件内存放，整体的内存使用量可以降低，但是导入时间更长。将每个页作为单独的文件也具有一些优势，可以使多个设计师和美工人员同时处理相同界面。而且作为技术和设计的需求，在决定UI项目结构时可以确保考虑到工作流程。

与传统的桌面或web应用不同，这里特别需要了解可以有很多种不同的方法来创建丰富的多媒体游戏界面。每个引擎，甚至不同的平台，实现方法也有所不同，有的方法使用起来更加高效，而有的则更加低效。例如，放置一个网页界面到单个Flash文件。这样管理起来更加方便，转换操作也更加容易，但是增加了内存的消耗，对于老的游戏控制器或移动终端则非常不利。如果一切都在单个Flash文件中进行，则不能使多个设计师同时工作在同一个界面的不同部分。如果项目为一个复杂的界面设计，需要一个庞大的设计团队，或者具有其他特殊的技术和设计需求，则最好将每个UI页面放置到不同的Flash文件。在很多情况下，这两种方案都是可行的，但是，在特定的项目中，其中一种可能比另外一种方案更具有优势。例如，屏幕可能需要根据需求导入和导出，或者在网络上动态更新和传输。衡量的底线为应该仔细评估UI界面资源的管理，从UI的逻辑规划到工作流实现以及性能都需要考虑完善。

虽然Scaleform强烈建议开发者使用Flash和ActionScript作为UI界面主要开发手段，但没有完美的答案，某些团队也许更喜欢用应用引擎脚本语言（如Lua或UnrealScript）来完成大多数繁重工作。在这些情况下，Flash应该被主要用作动画实现工具，只包含极少量的动态脚本ActionScript和Flash文件内部的交互内容。

在早期了解技术、设计和工作流程的需求非常重要，然后继续评估整个过程的整体方法，特别是在深入项目之前，确保顺利和最终的成功。

2 内置组件

初次观察， Scaleform

CLIK为一个基本的内置UI组件集，但是CLIK的真实意图是为创建丰富组件和界面提供一个框架。开发者可以自由 - 并可以按设想进行扩展 - 创建自定义组件适应自身需求并在CLIK框架上进行构建。

内置CLIK组件提供标准的UI功能，从基本的按钮和复选框到列表框、下拉菜单和模式对话框等复合组件。开发者可以方便得将这些标准组件进行扩展，增加更多特性或在从头创建自定义组件时作为简单的参考组件。

以下选项详细得描述了每个内置组件。他们按照复杂性和功能性进行分组。每个组件使用子章节列表进行描述。

- **User interaction** : 用户如何与组件进行交互。
- **Component setup**: 当在Flash授权环境中构造组件时需要的元素。
- **States**: 组件到函数所需要的不同可视状态（关键帧）。
- **Inspectable properties**: 在Flash授权环境中公布的属性，便于不使用代码配置特定组件特性。
- **Events**: 在UI界面中其他对象可以监听的组件所触发的事件列表。

- **Tips and tricks:** 完成各种与讨论组件相关的任务的实例代码。

2.1 基本按钮和文本类型

基本类型包括Button、CheckBox、Label、TextInput

和TextArea组件。按钮Button构成了多数用户界面的主干，CheckBox继承了Button的功能。Label为一个静态标签类型，而TextInput 和 TextArea分别可以表示单行文本和多行文本。



图1：来自*Free Realms*的主菜单按钮实例

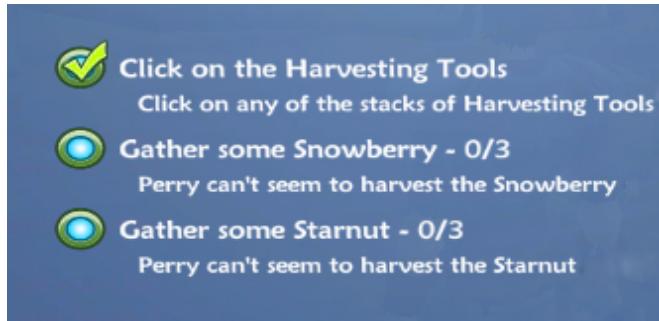


图2：来自*Free Realms*复选框（Check box）实例

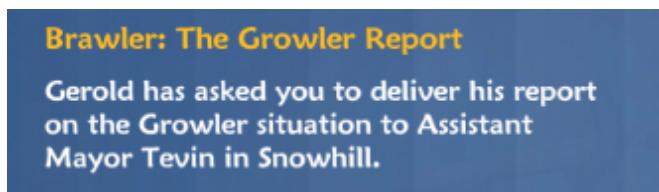


图3：来自*Free Realms*的标签（Label）实例



图4：来自*Crysis Warhead*的文本输入框（Text input）实例

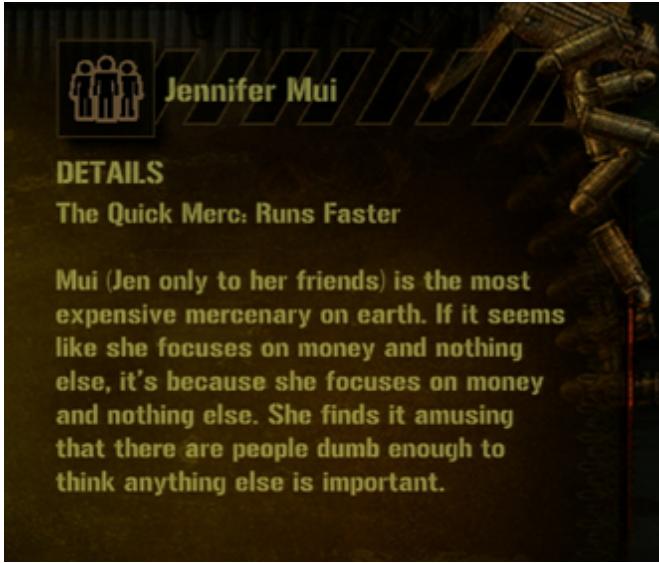


图5：来自*Mercenaries 2*的文本区域（Text area）实例

2.1.1 Button



图6：无皮肤按钮

按钮Button为CLIK框剪的基本组件，可以在任何地方使用需要一个能发出滴答声的界面控制。默认的Button类（gfx.controls.Button）支持一个textField来显示一个标签，并制定可视化用户互动。按钮可以单独使用，也可以作为合成组件的一部分，如作为ScrollBar的方向按钮或者Slider翻页按钮。大多数交互组件可以响应点击动作或为扩展按钮。

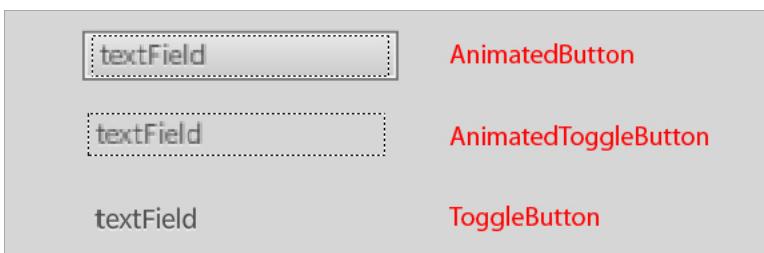


图7：AnimatedButton、AnimatedToggleButton和ToggleButton

CLIK

Button为一个通用按钮组件，支持鼠标交互，键盘交互，状态和其他功能，可以在多种用户界面中使用。同时也支持选中功能以及动画状态。**ToggleButton**、**AnimatedButton**和**AnimatedToggleButton**位于**Button.fla**组件源文件中，使用相同的基本组件类。

2.1.1.1 用户交互

按钮组件可以用鼠标或任何类似控制器点击。当获得焦点时也可以通过键盘按钮控制。

通常，只有一个获得焦点的组件接收键盘事件。设置一个组件的焦点有多种方法。其中一些方法在本文的[Programming](#)

[设计细节](#)中有所描述。大多数**CLIK**组件当交互时，特别是按下鼠标左键或类似控制器在上面按下（点击）时也可以接收焦点。[\(Tab\)](#)

和[\(Shift+Tab\)](#)键（或对应导航控制）可以在显示的焦点组件上移动焦点。这种特性也是大多数桌面应用软件和网页上所提供的。注意非**CLIK**元素使用的焦点也可以被**CLIK**组件使用。这意味着一个**Flash**开发者能够将**CLIK**元素和非**CLIK**元素在场景相互混合和匹配并使焦点行为按意图进行，特使在使用[\(Tab\)](#)和[\(Shift+Tab\)](#)键时可以在场景中移动焦点。

默认情况下按钮响应（[Enter](#)）键和空格键。将鼠标箭头移动到按钮上面然后移开也会使组件产生动作，拖动鼠标光标移入和移出也一样。

在游戏控制器或掌上游戏机，开发者能简单用对应游戏杆来控制键盘和鼠标控制事件。例如，（[Enter](#)）键在Xbox360或

PS3控制器上通常映射为（A）或（X）按钮。此映射使UI界面中使用**CLIK**应用到多种类型的平台之上。

2.1.1.2 组件设置

一个使用**CLIK Button**类的**MovieClip**必须具备下面所列的子单元。也列出了可选元素。

- **textField:** (可选) **TextField** 类型。按钮标签
- **focusIndicator:** (可选) **MovieClip**类型。一个单独的**MovieClip**

用来显示焦点状态。如果被使用，必须有两个命名帧：“show” 和 “hide”。默认情况下，**over**状态用来表示一个获得焦点的**Button**组件。但是在有些类中，这个行为限制了使用，设计师可能要将焦点状态和鼠标**over**状态分开使用。

2.1.1.3 状态

CLIK 按钮组件支持基于用户交互的不同的状态。这些状态包括：

- **up** 或默认状态；

- 当鼠标箭头在组件上方或者获得焦点时**over**状态；
- 当按钮按下时**down**状态；
- **disabled**状态；

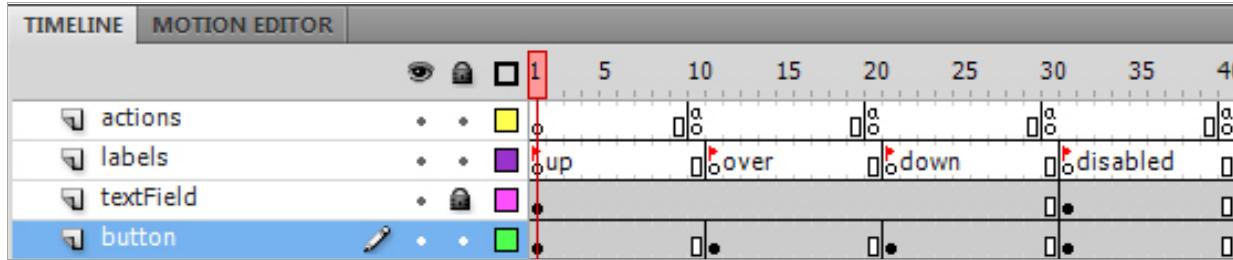


图8: Button时间轴

这些状态在Flash时间轴中用关键帧来表示，为按钮组件所需要的最少关键帧设置的正确操作。同时还有其余状态扩展组件功能以支持复杂用户交互和动画转换，这些信息在文档[Getting Started with CLIK Buttons](#)中有所描述。

2.1.1.4 属性检查

组件的重要属性可以通过Flash IDE的Component Inspector面板或者Parameters标签进行设置。需要在CS4中打开该面板，在上方工具条中选择Window下拉菜单，点击启动Component Inspector窗口，或者按下(Shift+F7)键。这将打开Component Inspector面板。这些被称之为“检查属性”。这为不熟悉AS编程美工和设计师配置组件行为和功能提供了一种便利的方法。

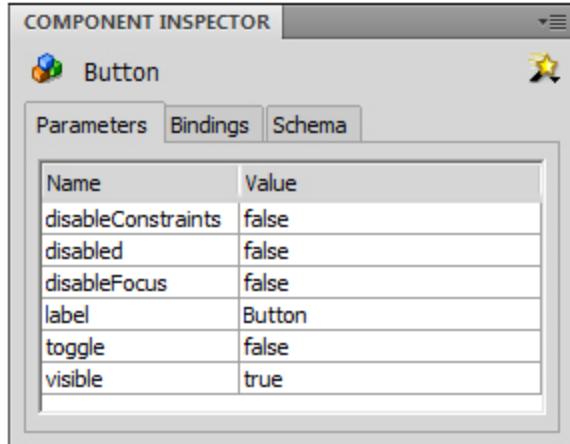


图9: CS4组件观察中的按钮组件属性检查窗口

按钮组件的检查属性为：

disableConstraints	按钮Button组件包含了在组件调整大小时按钮中的textField的定位和缩
---------------------------	---

放参数。设置该属性为true则禁止强制对象。这在需要通过时间轴动画调整按钮、 <code>textField</code> 大小或者重新定位特别有用，按钮大小不会发生变化。如果为disabled、 <code>textField</code> 在整个生命周期都将使用其默认参数值，这将禁止在按钮时间轴上创建的任何 <code>textField</code> 转换/缩放动作。	
disabled	如果设置为true则禁止按钮。一旦为disabled，则按钮不接收焦点。
disableFocus	默认情况下接受焦点用于用户交互。设置该属性为ture将禁止焦点获取。
Label	设置按钮中的文本显示。
Toggle	设置按钮套索模式。如果设置为true，按钮将作为一个套索按钮使用。
Visible	如果设置为false则隐藏按钮。

改变属性只能在发布包含该组件的SWF文件后才有效。Flash

IDE在设计阶段场景中不显示任何改变，因为CLIK组件不属于编译剪辑。这是用来确保组件易于使用和绘制皮肤。

2.1.1.5 事件

大多数组件产生事件用户交互、状态改变和焦点管理。这些事件在使用CLIK组件创建一个功能丰富的用户界面时非常重要。

所有事件调用接收一个对象参数，包含事件相关信息。以下为所有事件的常用属性。

- **type:** 事件类型。
- **target:** 产生事件的目标。

按钮组件产生事件列表如下。事件旁边的属性列表为通用属性的补充内容。

Show	运行时可视属性已设置为true
Hide	运行时可视属性已设置为false
focusIn	按钮获得焦点
focusOut	按钮失去焦点
Select	所选属性已改变。 Selected: 按钮的选择状态，true为被选择，为布尔类型。
stateChange	按钮状态已改变。 State: 按钮新状态。String类型。值为“up”、“over”、“down”等。 参考 CLIK按钮入门 文档获取详细的状态列表信息。
rollOver	鼠标箭头在按钮上方滚动。 mouseIndex: 用来产生事件的鼠标光标索引（只应用在多鼠标-

	光标环境）。数值类型。值为0-3。
rollOut	鼠标光标从按钮移开。 <i>mouseIndex</i> : 用户产生事件的鼠标光标索引（只应用在多鼠标光标环境） 数值类型。值为0-3。
Press	按钮被点击。 <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标-箭头环境）。数值类型。值为0-3。
doubleClick	按钮被双击。只在 <i>doubleClickEnabled</i> 属性为true时被触发。 <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标-光标环境）。数值类型。值为0-3。
Click	按钮被点击。 <i>mouseIndex</i> : 用来产生事件的鼠标光标索引（只应用在多鼠标-光标环境）。数值类型。值为0-3。
dragOver	鼠标光标拖动到按钮上方（鼠标左键被按下） <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标-箭头环境）。数值类型。值为0-3。
dragOut	鼠标箭头从按钮拖开（鼠标左键被按下）。 <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标-箭头环境）。数值类型。值为0-3。
releaseOutside	鼠标箭头从按钮拖开鼠标左键松开。 <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标-箭头环境）。数值类型。值为0-3。

ActionScript代码片段用来捕获或处理这些事件。下例中展示了如何处理按钮点击事件。

```
myButton.addEventListener("click", this, "onButtonPress");
function onButtonPress(event:Object) {
    // 执行操作
}
```

第一行代码为“click”事件安装事件监听器，按钮名称为‘myButton’，当事件触发时使其调用onButtonPress函数。相同的代码也可以用来处理其他的事件。事件句柄中返回的Object参数（例子中名为‘event’）包含了事件的相关信息。这些信息作为Object参数属性返回并与多数事件有所不同。

2.1.1.6 提示和技巧

运行过程中用自定义属性创建按钮组件：

```
import gfix.controls.Button;
attachMovie("Button", "btnInstanceName", someDepth, {_x:33, _y:262, ...});
```

设置CLIK按钮实例可以在选中和未选中之间切换。这里不需要通过属性检查窗口进行Toggle属性的设置。

```
btn.toggle = true;
```

使能鼠标左键双击：

```
btn.doubleClickEnabled = true;
btn.addEventListener("doubleClick", this, "onDoubleClick");
function onDoubleClick(e:Object):Void {
    // 鼠标被双击!
}
```

在按钮被按下时使能点击事件的重复触发：

```
btn.autoRepeat = true;
```

2.1.2 CheckBox



图 10：无皮肤复选框CheckBox

复选框CheckBox (gfx.controls.CheckBox)为一个按钮组件当被点击时设置为选中状态。复选框用来显示true/false（布尔值）的变化。与ToggleButton功能类似，但是隐藏设置Toggle属性。

2.1.2.1 用户交互

使用鼠标或者任何相关键盘控制器点击CheckBox组件可以使其为选中或未选中。在其他方面，复选框行为与按钮相同。

2.1.2.2 组件设置

使用CLIK CheckBox类的动画剪辑MovieClip必须具备下面所列的子单元。也列出了可选元素：

- **textField:** (可选) TextField 类型，按钮标签。
- **focusIndicator:** (可选) MovieClip 类型，一个独立的 MovieClip 用来显示焦点状态。如果被使用，该 MovieClip 必须有两个名字为“show”和“hide”的帧。

2.1.2.3 状态

根据 Toggle 属性，复选框 CheckBox 需要另外的关键帧集来表示选择状态。这些状态包括：

- **up** 或默认状态；
- 当鼠标箭头在组件上方或者获得焦点时为 **over** 状态；
- 当按钮被点击时候为 **down** 状态；
- **disabled** 状态；
- **selected_up** 或者默认状态；
- 当鼠标箭头位于组件上方或获得焦点时为 **selected_over** 状态；
- 当按钮被按下时为 **selected_down** 状态；
- **selected_disabled** 状态；

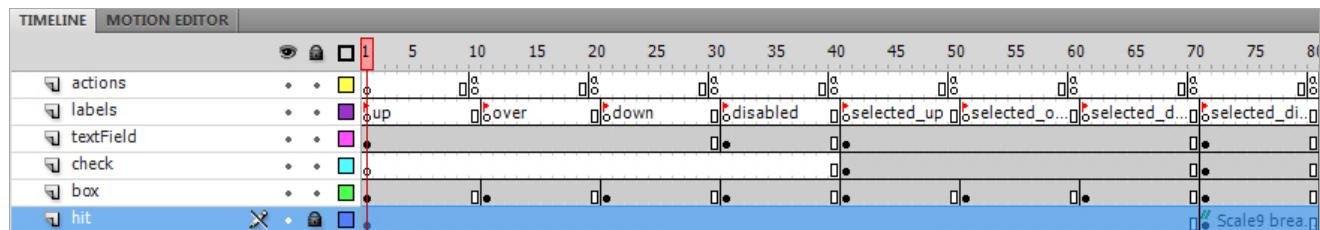


图 11: CheckBox 时间轴

这里为复选框 CheckBox 所需要的最少关键帧设置。按钮 Button 组件支持状态和关键帧的扩展设置，与复选框组件相同，在 [CLIK 按钮入门](#) 文档中有详细描述。

2.1.2.4 属性检查

由于从控制按钮继承而来，复选框 CheckBox 包含了与按钮相同的检查属性，具有 **disableFocus** 属性和 **Toggle** 属性。

data	自定义字符串可以配合组件作为独立数据使用，而不同于复选框 CheckBox 上的标签。
disableConstraints	按钮组件包含一个在组件调整大小时按钮内部 textField 的定位和缩放对象。设置该属性为 true 将禁止该强制对象。这在通过时间轴动画调整或重定位 textField 时特别有用，而按钮组件不需改变尺寸。如果为 disabled ， textField 在整个生命周期都将使用其默认参数值，这将禁止在按钮时间轴上创建的任何 textField 转换/缩放动作。

disabled	如果设置为true禁止按钮
label	设置按钮标签
selected	当设置为true使得复选框CheckBox有效（被选中）。
visible	如果设置为false隐藏按钮。

2.1.2.5 事件

所有的事件调用都接收一个对象Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 产生事件的目标对象。

复选框CheckBox组件产生的事件列表如下。事件旁边的属性列表为通用属性的补充内容。

show	运行时可视属性已设置为true
hide	运行时可视属性已设置为false
focusIn	组件获得焦点
focusOut	组件失去焦点
select	所选属性已改变。 <i>Selected:</i> 按钮的选择状态，true为被选择，为布尔类型。
stateChange	按钮状态已改变。 <i>State:</i> 按钮新状态。String类型。值为“up”、“over”、“down”等。 参考 CLIK 按钮入门 文档获取详细的状态列表信息。
rollOver	鼠标箭头在按钮上方滚动。 <i>mouseIndex:</i> 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。
rollOut	鼠标箭头从按钮移开。 <i>mouseIndex:</i> 用户产生事件的鼠标箭头索引（只应用在多鼠标箭头环境） 数值类型。值为0-3。
press	按钮被点击。 <i>mouseIndex:</i> 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。
doubleClick	按钮被双击。只在 doubleClickEnabled 属性为true时被触发。 <i>mouseIndex:</i> 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。
click	按钮被点击。 <i>mouseIndex:</i> 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。

dragOver	鼠标箭头拖动到按钮上方（鼠标左键被按下） <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标-箭头环境）。数值类型。值为0-3。
dragOut	鼠标箭头从按钮拖开（鼠标左键被按下）。 <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标-箭头环境）。数值类型。值为0-3。
releaseOutside	鼠标箭头从按钮拖开鼠标左键松开。 <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标-箭头环境）。数值类型。值为0-3。

下例中代码展示了如何处理复选框CheckBox的Toggle动作：

```
myCheckBox.addEventListener("select", this, "onCheckBoxToggle");
function onCheckBoxToggle(event:Object) {
    // 执行操作
}
```

2.1.3 Label



图 12: 未绘制皮肤的标签

CLIK

标签Label组件(gfx.controls.Label)为一个不可编辑的标准textField组件，由MovieClip符号进行围绕，具有一些附加的便利特性。在本质上，标签Label支持与标准textField相同的属性和行为，但是，有一些常用的特性为该组件本身所特有。如果用户需要直接改变其属性，支持访问标签实际上的textField区域。在某些情况下，如一些下面内容将会描述的内容，开发者可能会使用textField替代标签组件。

由于标签Label为一个MovieClip符号，可以用图形元素进行修饰，而这在标准的textField无法做到。作为一个符号，不需要如textField实例一样对每一个进行配置。标签Label还提供了disabled状态在时间轴可以被定义。然而，用标准的textField来模拟这些功能需要很多复杂的AS2代码。

标签Label组件默认强制使用，这意味着在运行时在场景中改变一个标签Label实例大小不会显示出效果，开发者应该在大多数情况下使用textField来提到Label标签。通常，文本元素不需要经常被重用，则textField比起标签Label使用更加简单。

2.1.3.1 用户交互

在标签Label内无用户交互。

2.1.3.2 组件设置

使用CLIK Label类的MovieClip必须拥有下列命名的子元素。标注了对应的可选元素：

- **textField**: TextField 类型， Label 标签文本

2.1.3.3 状态

CLIK Label组件支持基于标准属性的两种状态：

- **default** 或者enabled 状态；
- **disabled** 状态

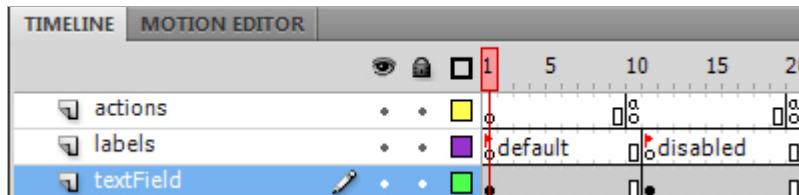


图 13: Label 时间轴

2.1.3.4 检查属性

标签Label的检查属性如下：

text	设置标签文本
visible	如果设置为false隐藏标签
disabled	如果设置为true禁止标签

2.1.3.5 事件

所有事件调用接收单个Object参数包含事件相关信息。以下为所有事件的通用属性：

- **type**: 事件类型

- **target:** 产生事件的目标

由标签Label组件产生的事件列表如下。事件旁边的属性列表为通用属性的补充内容。

show	运行时组件可视属性已设置为true
hide	运行时组件可视属性已设置为false
stateChange	Label标签状态已改变。 State: Label新状态。String类型。值为“default”或“disabled”。

下面例子为如何监听标签Label状态改变：

```
myLabel.addEventListener("stateChange", this, "onStateChange");
function onStateChange(event:Object) {
    if (event.state == "default") {
        // 执行操作
    }
}
```

2.1.3.6 提示和技巧

在标签Label组件中显示HTML文本。关于在标准textField 中支持HTML标签请参考Flash 8文档：

```
lbl.htmlText = "<b>foo</b>bar";
```

2.1.4 TextInput

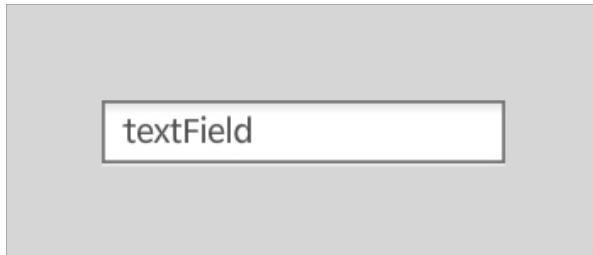


图 14: 无皮肤TextInput.

文本输入TextInput

(gfx.controls.TextInput)为一个可编辑的textField组件，用来捕获用户文本输入。与标签Label类似，该组件仅仅为一个标准textField的外框，因此支持textField的功能，如密码模式、最大字符数和HTML文本。只有一部分属性为该组件本身所有，其他的属性可以直接进入TextInput的textField实例进行更改。

textField组件应该用于输入，因为无需编辑文本可以使用**Label**标签来显示。与**Label**标签类似，开发者可能会根据自身需求用标准的**textFields**代替**TextInput**组件使用。但是，当开发复杂UI界面时，特别是在PC应用中，**TextInput**组件提供了基于标准**textField**的有价值的功能扩展。

作为特殊属性，**TextInput**支持焦点和**disabled**状态，这在标准**textField**中很难实现。根据独立的焦点状态，**TextInput**支持自定义焦点指示器，这在标准**TextInput**也不包含。复杂的AS2代码需要改变标准**TextInput**的外观类型，而**TextInput**的外观类型可以在时间轴上简单得进行配置。**TextInput**检查属性为不熟悉Flash

Studio的设计师和编程人员提供了一种简单的工作流程。开发者可以简便得监听**TextInput**触发的事件以创建自定义行为。

TextInput同时也支持**textField**，提供的标准选择和剪切、拷贝和粘贴功能，包括了多行HTML格式文本。默认情况下，快捷键命令为选择 (Shift+Arrows)、剪切 (Shift+Delete)、拷贝 (Ctrl+Insert)、和粘贴(Shift+Insert)。

“文本输入”可支持鼠标滚动和滑出事件。可通过对这一特殊的**actAsButton**属性进行设置，来提供能够执行两种鼠标事件的两个额外关键帧。这些帧被命名为“over”和“out”，分别代表滚动和滑出状态。如果设置了**actAsButton**模式，并且“over”/“out”帧不存在，那么“文本输入”将会按照“默认值”关键帧执行这两种事件。请注意，这些帧不会通过预设的“文本输入”组件而出现。开发商会根据具体要求对他们进行添加。

当用户未设定或输入值时，该组件还支持默认显示的文本。可将默认文本属性设置为任何字符串。默认文本的主题（颜色和样式）为浅灰(0xAAAAAA)，斜体。可通过向“默认文本格式”属性分配一个新的“文本格式”对象的方法对样式进行自定义。

2.1.4.1 用户交互

点击**TextInput**使其获得焦点，则在**textField**中出现一个箭头。当箭头显现时，用户能够通过键盘或类似控制设备输入字符。按下坐右方向键可以移动箭头。当箭头已经位于**textField**的左边缘，使用左方向键则焦点将转移到左边的控制部件。使用右方向键也如此。

2.1.4.2 组件设置

使用CLIK **TextInput**类的MovieClip必须具备下面所列的子单元。也列出了可选元素：

- **textField**: **textField** 类型

2.1.4.3 States状态

CLIK TextInput组件支持三种状态，均基于焦点和disabled属性：

- **default** 或enabled 状态；
- **focused** 状态，通常为textField周围突出显示的边框；
- **disabled** 状态。

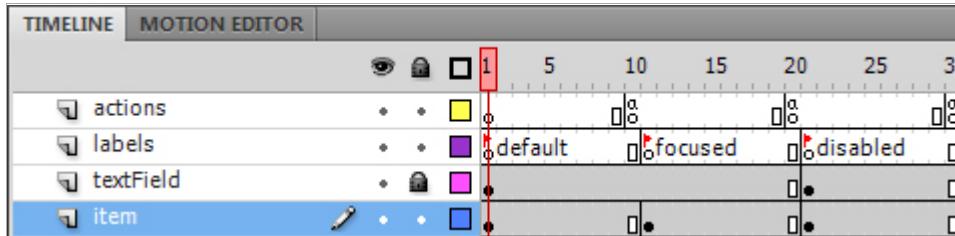


图 15: TextInput 时间轴

2.1.4.4 检查属性

文本输入TextInput组件的检查属性如下所示：

text	设置textField文本
visible	如果设置为false则隐藏组件
disabled	如果设置为true则禁止组件
editable	如果设置为false则使TextInput不可编辑
maxChars	一个大于零的数字，作为textField中可以输入的最多字符数。
password	如果为true，设置textField显示“*”字符代替实际字符。而textField接收的值为输入字符的实际值，返回对应文本。
defaultText	当“文本字段”为空时所显示的文本。该文本由“默认文本格式”对象控制，该对象的默认设置为浅灰和斜体。
actAsButton	如果为“真”，则“文本输入”的行为在未选中状态下类似一个按钮，并支持滚动和滑出状态。一旦按下鼠标或tab键，“文本输入”将会转为正常模式，直至退出选中状态。

2.1.4.5 事件

所有调用接受一个Object参数，包含了事件的相关信息。以下为事件的通用属性。

- **type:** 事件类型。
- **target:** 产生事件的目标。

文本输入TextInput组件产生的事件列表如下。事件旁边的属性列表为通用属性的补充内容。

show	运行时可视属性已设置为true
hide	运行时可视属性已设置为false
focusIn	组件获得焦点
focusOut	组件失去焦点
textChange	textField 内容发生改变
rollover	鼠标光标在组件上滚动。只有当设置了“动作按钮”后才会启动。 <i>mouseIndex:</i> 用于生成事件的鼠标索引（仅适用于多鼠标光标环境）。数字类型值为0到3.
rollout	鼠标光标在组件上滑出。只有当设置了“动作按钮”后才会启动。 <i>mouseIndex:</i> 用于生成事件的鼠标索引（仅适用于多鼠标光标环境）。数字类型值为0到3.

以下代码为揭示如何监听textField内容变化的实例：

```
myTextInput.addEventListener("textChange", this, "onTextChange");
function onTextChange(event:Object) {
    // 执行操作
}
```

2.1.4.6 提示和技巧

当获得焦点时停止自动选择文本：

```
_global.gfxExtensions = true;
textinput.textField.noAutoSelection = true;
```

在TextInput组件中显示HTML文本，关于在标准textField 中支持HTML标签请参考Flash 8文档：

```
textinput.htmlText = "<b>foo</b>bar";
```

2.1.5 TextArea

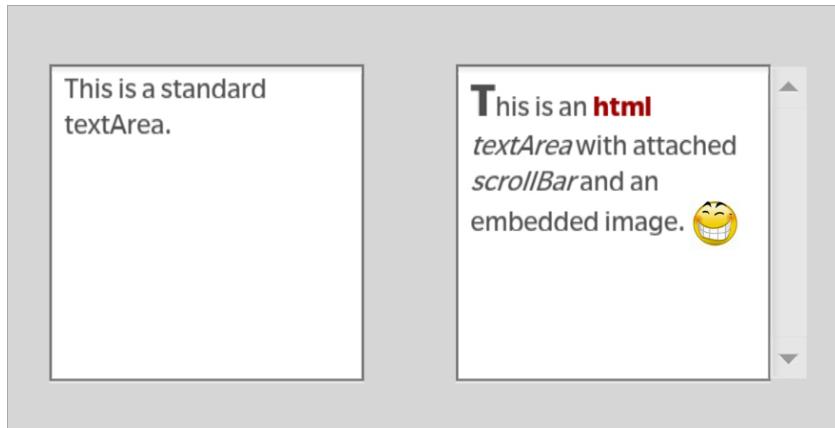


图 16: 无皮肤TextArea。

文本区域TextArea (`gfx.controls.TextArea`)从CLIK

`TextInput`继承而来，共享相同功能、属性和状态，但是具有一个可选滚动条`ScrollBar`用于多行可编辑滚动文本输入。请参考“文本输入”描述，学习更多的有关“文本输入”和“文本区域”共享的特殊功能。

类似于Label和`TextInput`，`TextArea`也为一个标准的多行`textField`的外框，因此支持`textField`的属性和行为，如HTML文本、文字边框、选择、剪切、拷贝、粘贴。开发者可以简单得用一个标准的`textField`替代`TextArea`，但是，强烈建议使用该组件，因为其具有扩展功能、状态、检查属性和事件。

尽管标准`textField`能够用于`ScrollIndicator`或者`ScrollBar`，`TextArea`提供了与这些组件紧凑的组合功能。与标准的`textField`不同，`TextArea`能够在使用键盘或类似控制器使其获得焦点时进行滚动，甚至在不可编辑时也如此。由于滚动组件不能获得焦点，`TextArea`能够展现更多有趣的焦点图型状态，能够在获得焦点的时候装饰自身和滚动组件。

2.1.5.1 用户交互

点击`TextArea`使其获得焦点，则在`textField`中出现一个箭头。当箭头显现时，用户能够通过键盘或类似控制设备输入字符。按下左右方向键可以移动箭头。当箭头已经位于`textField`的边缘，使用方向键则焦点将转移到相邻的控制部件。

2.1.5.2 组件设置

一个使用CLIK `TextArea`类的MovieClip必须具备下面所列的子单元。也列出了可选元素：

- **textField:** `TextField` 类型

2.1.5.3 状态

与上级组件TextInput类似，TextArea组件支持三种状态，以焦点和disabled属性为基础。

- **default** 或enabled 状态；
- **focused** 状态，通常为textField周围突出显示的边框；
- **disabled** 状态；

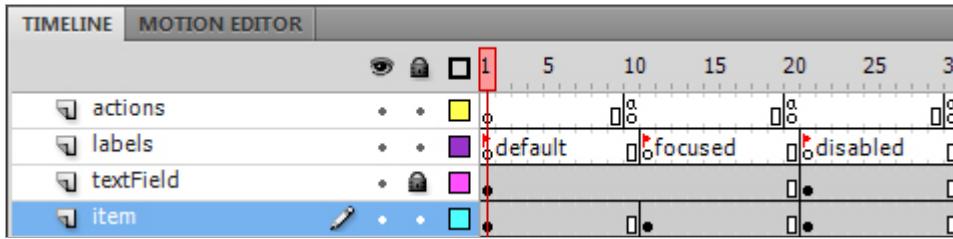


图 17: TextArea 时间轴

2.1.5.4 检查属性

TextArea组件检查属性与TextInput类似，具有一对辅助和冗长的密码特性。辅助特性与CLIK ScrollBar组件有关，将在2.4节描述：

text	设置 textField文本
visible	如果设置为false则隐藏组件
disabled	如果设置为false则禁止组件
editable	如果设置为false则使TextInput不可编辑
maxChars	一个大于零的数字，作为textField中可以输入的最多字符数。
scrollBar	CLIK ScrollBar组件使用的实例名，或者一个到ScrollBar符号的链接ID（本例中由TextArea创建一个实例）。
scrollPolicy	当设置为“auto”时，滚动条将只显示是否有足够的文本可以需要滚动。如果设置为“on”则ScrollBar将长期显示，如果设置为“off”则不显示，该属性只影响分配一个ScrollBar的组件（参考ScrollBar特性）
defaultText	当“文本字段”为空时所显示的文本。该文本由“默认文本格式”对象控制，该对象的默认设置为浅灰和斜体。
actAsButton	如果为“真”，则“文本输入”的行为在未选中状态下类似一个按钮，并支持滚动和滑出状态。一旦按下鼠标或tab键，“文本输入”将会转为正常模式，直至退出选中状态。

2.1.5.5 事件

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

文本区域TextArea组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false
focusIn	组件获得焦点
focusOut	组件失去焦点
textChange	textField 内容已改变
scroll	在文本区域滚动
rollover	鼠标光标在组件上滚动。只有当设置了“动作按钮”后才会启动。 <i>mouseIndex:</i> 用于生成事件的鼠标索引（仅适用于多鼠标光标环境）。数字类型值为0到3.
rollout	鼠标光标在组件上滑出。只有当设置了“动作按钮”后才会启动。 <i>mouseIndex:</i> 用于生成事件的鼠标索引（仅适用于多鼠标光标环境）。数字类型值为0到3.

以下例子展示了如何监听TextArea滚动事件：

```
myTextArea.addEventListener("scroll", this, "onTextScroll");
function onTextScroll(event:Object) {
    // 执行操作
}
```

2.1.5.6 插入

当获得焦点时停止自动选择文本：

```
_global.gfxExtensions = true;
textInput.textField.noAutoSelection = true;
```

在TextArea组件中显示HTML文本。在标准textField中支持HTML标签请参考Flash 8 文档。

```
textInput.htmlText = "<b>foo</b>bar";
```

2.2 分组类型

分组类型包括RadioButton、ButtonGroup和

ButtonBar组件。ButtonGroup为一个管理类型具有特别的逻辑来维护按钮Button的分组。不具有可视外观不存在于场景中。但是，ButtonBar存在于场景中也用来维护按钮分组。RadioButton为一个特殊的按钮可以自动与同类组件分组到ButtonGroup中。

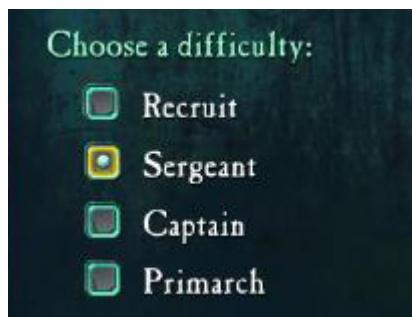


图 18:来自*Dawn of War II*的选择按钮组

2.2.1 RadioButton

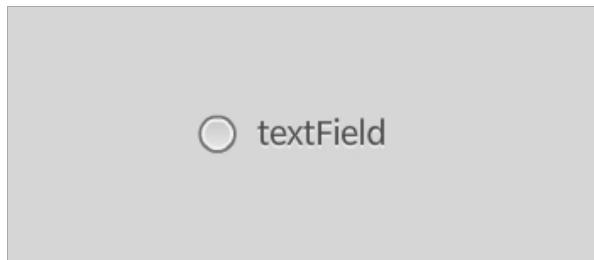


图 19:无皮肤的选择按钮RadioButton

选择按钮RadioButton

(gfx.controls.RadioButton)为一个按钮组件，通常属于一个集用来显示和改变一个值。在这个集中只能选择一个选择按钮，点击集中另外一个选择按钮将选中一个新的组件，之前被选中的组件将失去被选择状态。

CLIK选择按钮与复选框CheckBox组件非常类似，共享功能、状态和行为。主要的区别为选择按钮支持分组属性，可以指派一个自定义按钮组ButtonGroup（见下节）。选择按钮不需要固定设置为选中属性，因为选中属性由按钮组实例进行管理。

2.2.1.1 交互

使用鼠标或类似控制器点击未选中的选择按钮组件将其选中。如果选择按钮为被选中状态，另外一个同属一个按钮组的选择按钮被点击，则先前选中的选择按钮将失去选中状态。其他方面，选择按钮行为与按钮相同。

2.2.1.2 编辑

使用CLIK RadioButton类的MovieClip必须具备下面所列的子单元。也列出了可选元素：

- **textField:** (可选) TextField 类型，按钮标签。
- **focusIndicator :** (可选MovieClip 类型，一个独立的 MovieClip 用来显示焦点状态。如果被使用，该 MovieClip 必须具有两个帧名为：“show” 和 “hide”。

2.2.1.3 状态

由于选择按钮可以在选中和未选中状态间转换，与复选框CheckBox类似，需要至少以下几种状态：

- **up** 或默认状态；
- 当鼠标箭头在组件上方或者获得焦点时为**over** 状态；
- 当按钮被点击时候为**down** 状态；
- **disabled** 状态；
- **selected_up** 或者默认状态；
- 当鼠标箭头位于组件上方或获得焦点时为**selected_over** 状态；
- 当按钮被按下时为 **selected_down** 状态；
- **selected_disabled** 状态；

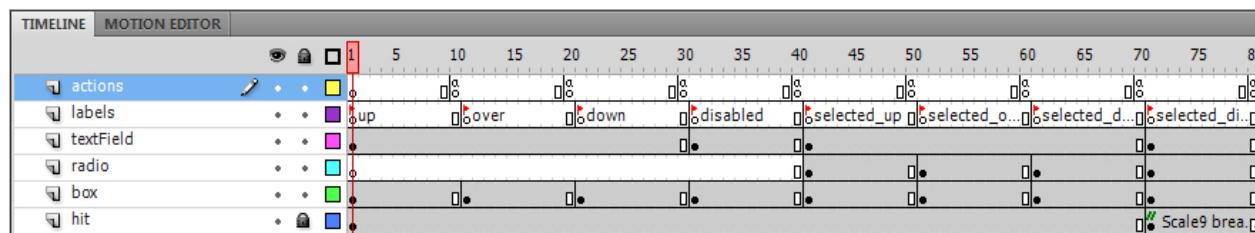


图 20:选择按钮RadioButton时间轴

这里为选择按钮RadioButton所需要的最少关键帧设置。按钮Button组件支持状态和关键帧的扩展设置，与选择按钮RadioButton组件相同，在[CLIK 按钮入门](#)文档中有详细描述。

2.2.1.4 检查属性

由于从按钮Button控制按钮继承而来，选择按钮RadioButton包含了与按钮相同的检查属性，具有disabled、Focus属性和Toggle属性。

label	设置按钮Button的标签
visible	如果设置为false则隐藏按钮
disabled	如果设置为true则禁止按钮
disableConstraints	Button组件包含了在组件调整大小时按钮中的textField的定位和缩放参数。设置该属性为true则禁止强制对象。这在需要通过时间轴动画调整按钮textField大小或者重新定位特别有用，按钮大小不会发生变化。如果为disabled，textField在整个生命周期都将使用其默认参数值，这将禁止在按钮时间轴上创建的任何textField转换/缩放动作。
selected	当设置为true时确认（或选中）选择按钮RadioButton
data	自定义字符串可以配合组件作为独立数据使用，而不同于选择按钮RadioButton上的标签。
group	一个名为按钮组ButtonGroup的实例应该被使用或由选择按钮RadioButton自动创建。如果由选择按钮创建，新的ButtonGroup将在选择按钮容器内存在。例如，如果选择按钮存在于_root，则选择按钮对象将在_root下创建。所有的选择按钮使用相同的组属于同一个集

2.2.1.5 事件

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

选择按钮RadioButton组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false
focusIn	组件获得焦点
focusOut	组件失去焦点
select	所选属性已改变。 Selected: 按钮的选择状态，true为被选择，为布尔类型。
stateChange	按钮状态已改变。 State: 按钮新状态。String类型。值为“up”、“over”、“down”等。参考 CLIK按钮入门 文档获取详细的状态列表信息。

rollOver	鼠标箭头在按钮上方滚动。 <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。
rollOut	鼠标箭头从按钮移开。 <i>mouseIndex</i> : 用户产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）数值类型。值为0-3。
press	按钮被点击。 <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。
doubleClick	按钮被双击。只在 <i>doubleClickEnabled</i> 属性为true时被触发。 <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。
click	按钮被点击。 <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。
dragOver	鼠标箭头拖动到按钮上方（鼠标左键被按下） <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。
dragOut	鼠标箭头从按钮拖开（鼠标左键被按下）。 <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。
releaseOutside	鼠标箭头从按钮拖开鼠标左键松开。 <i>mouseIndex</i> : 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。

以下例子显示如何处理选择按钮RadioButton的选中状态：

```
myRadio.addEventListener("select", this, "onRadioToggle");
function onRadioToggle(event:Object) {
    // 执行操作
}
```

2.2.1.6 按钮

选择按钮RadioButton组不在相同的层：

```
import gfx.controls.ButtonGroup;
var myGroup:ButtonGroup = new ButtonGroup("groupName");
radio1.group = myGroup;
radio2.group = myGroup;
```

```
someMovie.radio1.group = myGroup;  
someMovie.radio2.group = myGroup;
```

2.2.2 ButtonGroup

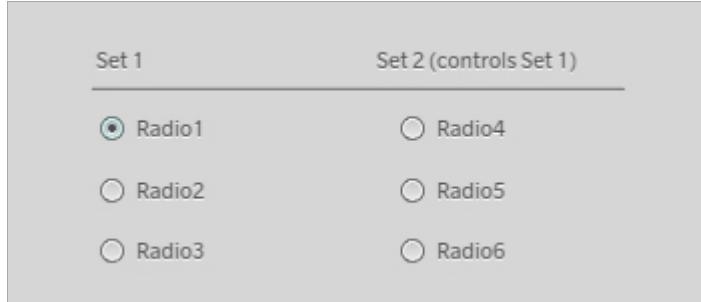


图 21: 无皮肤按钮分组 **ButtonGroup**.

CLIK ButtonGroup

(gfx.controls.ButtonGroup)本身不是一个组件，但是有重要作用用于管理按钮集。可以使在集中的一个按钮被选中，确保其余的未选中。如果用户选择集中的另一个按钮，则当前选中按钮将变为未选中。任何从CLIK按钮组件继承的组件（如复选框CheckBox和选择按钮RadioButton）能够使用按钮分组ButtonGroup实例。

2.2.2.1 用户交互

按钮组不具有用户交互功能，因为不是可视组件。但是，在下属的选择按钮RadioButton被点击时起到间接的作用。

2.2.2.2 组件设置

使用CLIK按钮组ButtonGroup类的MovieClip不需要任何子单元，因为不具有可视化外观。

2.2.2.3 状态

按钮组ButtonGroup在场景中不具有可视外观。因此无关联状态。

2.2.2.4 检查属性

按钮组ButtonGroup在场景中不具有可视外观。因此无需检查属性。

2.2.2.5 事件

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

ButtonGroup组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

change 在组中选择一个新的按钮

- *item* : 选择按钮，CLIK按钮Button类型
- *data*: 选择按钮的数值，AS2对象类型

itemClick 组中的按钮被点击

- *item* : 选择按钮，CLIK按钮Button类型

下例展示了如何判断按钮组ButtonGroup中哪个按钮被选中：

```
myGroup.addEventListener("change", this, "onNewSelection");
function onNewSelection(event:Object) {
    if (event.item == myRadio) {
        // 执行操作
    }
}
```

2.2.2.6 按钮

从组中找出当前选中的选择按钮：

```
var selectedRadio:Button = group.selectedButton;
```

在场景中为按钮组ButtonGroup中所属的默认选择按钮安装监听器

```
radioBtn1.group.addEventListener("itemClick", this, "onClick");
```

2.2.3 ButtonBar

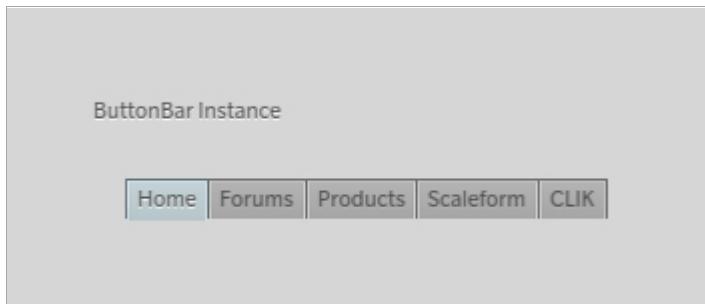


图 22: 无皮肤按钮栏 **ButtonBar**.

按钮栏ButtonBar

(gfx.controls.ButtonBar)与按钮组ButtonGroup类似，尽管具有可视化外观。也可以基于数据源dataProvider（参见[编程细节](#)描述dataProvider章节）动态创建按钮实例。按钮栏可用与创建动态tab栏UI元素。

```
buttonBar.dataProvider = [ "item1", "item2", "item3", "item4", "item5" ];
```

2.2.3.1 用交互

按钮栏与按钮组ButtonGroup具有相同的行为，也不能与用户直接交互，按钮栏也由按钮点击事件简介影响。

2.2.3.2 组件设置

使用CLIK按钮栏ButtonBar类的视频剪辑MovieClip不需要任何子单元，因为不具有可视化外观。

2.2.3.3 样

CLIK按钮栏不具有任何可视状态，因为其管理按钮组件只用来显示组的状态。

2.2.3.4 检查属性

尽管按钮栏组件没有内容（只为Flash

IDE场景中的一个简单小圆圈），但包含几个检查属性。主要由按钮栏ButtonBar创建的按钮实例分布设置决定。

visible	如果设置为false则隐藏ButtonBar
disabled	如果设置为true则禁止buttonBar
itemRenderer	按钮组件符号的链接ID，该符号将根据按钮栏的数据分配需求进行实例化

direction	按钮位置，水平并排放置按钮实例或垂直叠放
spacing	按钮实例间隔，只影响当前方位（见 direction 属性）
autoSize	如果设置为true，重新调整按钮实例以适应显示标签
buttonWidth	设置所有的按钮实例为常用宽度，如果autoSize设置为true忽略该属性

2.2.3.5 事件

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

ButtonBar组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false
focusIn	组件获得焦点
focusOut	组件失去焦点
change	在组中选择一个新的按钮 <ul style="list-style-type: none"> • <i>index</i>: 按钮栏的选择序列，数值类型，值为0到最小为1的按钮数 • <i>renderer</i> : 选择的按钮，CLIK按钮Button类型 • <i>item</i> : 数据源dataProvider选择项，AS2 Object类型 • <i>data</i>: 选择数据源dataProvider的数值，AS2 Object类型
itemClick	在组中点击按钮 <ul style="list-style-type: none"> • <i>index</i>: 点击按钮的按钮栏序列，数值类型，值为0到最小为1的按钮数 • <i>item</i> : 数据源dataProvider选择项，AS2 Object类型 • <i>data</i>: 选择数据源dataProvider的数值，AS2 Object类型 • <i>mouseIndex</i>: 用来产生事件的鼠标光标序号（只在多鼠标光标环境中使用），数值类型，值为0-3

以下例子展示了当在按钮栏中的按钮实例被点击后如何进行事件监听。

```
myBar.addEventListener("itemClick", this, "onItemClick");
function onItemClick(event:Object) {
    processData(event.data);
    // 执行操作
}
```

2.3 滚动类型

滚动类型包括ScrollIndicator、ScrollBar和

Slider。滚动指示器ScrollIndicator无需交互用来显示目标对象组件的滚动位置，而滚动条ScrollBar支持用户交互来改变滚动位置。滚动条由四个按钮组成：翻页按钮、轨道、向上方向箭和向下方向键。滑动条Slider与滚动条类似，但是只包含一个交互的翻页按钮和轨道，不根据目标组件的单元数来改变翻页按钮大小。

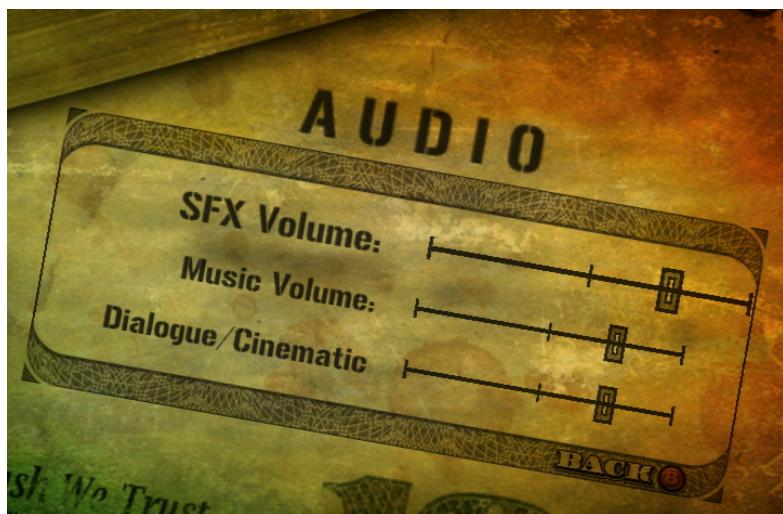


图 23: 来自**Mercenaries 2**的音频菜单滑动条实例

2.3.1 ScrollIndicator



图 24: 无皮肤 ScrollIndicator.

CLIK滚动指示器ScrollIndicator

(gfx.controls.ScrollIndicator)显示了其它组件的滚动位置，如多行文本区域TextField。可以在文本区域用来自动显示滚动位置。所有基于列表的组件，包括文本区TextArea，具有滚动条属性可以由滚动指示器或滚动条（见下节）实例或链接ID进行显示。

2.3.1.1 用户交互

滚动指示器与具有用户交互功能，因为只用来显示。

2.3.1.2 组件设置

一个使用CLIK ScrollIndicator类的MovieClip必须具备下面所列的子单元。也列出了可选元素：

- **thumb:** CLIK按钮Button类型，滚动指示块。
- **track:** 动画剪辑MovieClip类型，滚动指示轨道，轨道的边界决定了滚动块可以移动的位置。

2.3.1.3 状态

滚动指示器没有显性状态，使用子单元的状态：滚动块和轨道按钮组件。

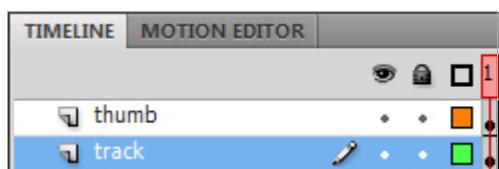


图 25:滚动指示器 ScrollIndicator 时间轴

2.3.1.4 检查属性

滚动指示器的检查属性如下所示：

scrollTarget	设置一个文本区域TextArea或者常用多行文本域textField作为滚动目标自动响应滚动事件。非文本域textField类型需要手动更新滚动指示器ScrollIndicator属性。
visible	如果设置为false则隐藏组件
disabled	如果设置为false则禁止组件
offsetTop	拖动点顶部偏移。正值将使拖动点向顶部位置的更高处移动。
offsetBottom	拖动点底部偏移。正值将使拖动点向底部位置的更低处移动。

2.3.1.5 事件

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

ScrollIndicator组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false
scroll	滚动位置发生改变。 <ul style="list-style-type: none">• <i>position:</i> 新滚动位置，数值类型，值为最小位置到最大位置

下例显示如何监听滚动事件：

```
mySI.addEventListener("scroll", this, "onTextScroll");
function onTextScroll(event:Object) {
    trace("scroll position: " + event.position);
    // 执行操作
}
```

2.3.1.6 提示和技巧

设置一个独立的手动滚动指示实例：

```
scrollInd.setScrollProperties(pageSize, minPos, maxPos, pageScrollSz);
scrollInd.positon = currPos;
```

设置滚动方向，使用`_rotation`属性在场景中创建组件时自动设置。如果滚动指示器组件不可旋转默认为水平方向，则需要明确的设置该值。

```
scrollInd.direction = "horizontal";
```

2.3.2 ScrollBar

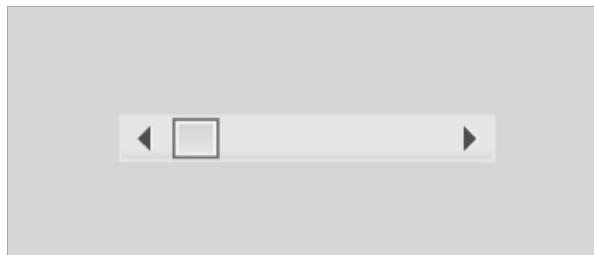


图 26: 无皮肤滚动条ScrollBar.

CLIK滚动条ScrollBar(gfx.controls.ScrollBar)显示和控制其它组件的滚动位置。通过拖动滚动块按钮增加交互功能到滚动指示器，同时还包括向上和向下方向键和一个可以点击的轨道。

2.3.2.1 用交互

滚动条ScrollBar上的滚动块可以由鼠标或类似控制器进行控制，可以在滚动条轨迹边界之内拖动。当鼠标光标位于滚动条上方时移动鼠标滚轮可以进行滚动操作。点击向上方向键可以向上滚动，点击向下方键可以向下滚动。点击轨道有两种行为：滚动块继续沿点击点位置滚动，或者立即跳转到点击点位置并设为拖动状态。轨道模式由*trackMode*检查属性决定（见检查属性小节）。忽略轨道模式*trackMode*设置，按下(Shift)键并点击轨道将立即将滚动快移动到鼠标光标处并设为拖动状。

2.3.2.2 组件设置

一个使用CLIK ScrollBar类的MovieClip必须具备下面所列的子单元。也列出了可选元素：

- **upArrow**: CLIK 按钮类型，向上滚动按钮；通常位于滚动条顶部。
- **downArrow**: CLIK 按钮类型，向下滚动按钮；通常位于滚动条底部。
- **thumb**: CLIK按钮类型，滚动条上的滚动块。
- **track**: CLIK按钮类型，滚动条轨道，其边界决定了滚动块的活动方位。

2.3.2.3 状态

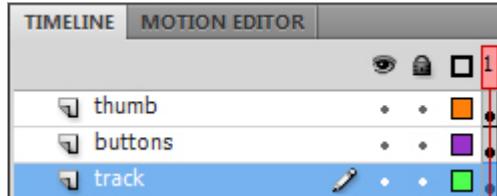


图 27: 滚动条ScrollBar 时间轴

滚动条ScrollBar，与滚动指示器ScrollIndicator类似，不需要显性状态。使用子单元的状态包括：滚动块、向上按钮、向下按钮和轨道按钮组件。

2.3.2.4 检查属性

滚动条检查属性和滚动指示器类似，只增加一条：

scrollTarget	响应滚动事件时设置文本区TextArea或常用多汗文本域textField的滚动目标位置
trackMode	当用户用鼠标点击滚动条，滚动页scrollPage设置将导致滚动块翻页知道释放师表。scrollToCursor设置可以使滚动块立即跳转到鼠标光标所在位置并转入拖动模式直到释放鼠标。
visible	如果设置为false则隐藏组件
disabled	如果设置为false则禁止组件
offsetTop	拖动点顶部偏移。正值将使拖动点向顶部位置的更高处移动。
offsetBottom	拖动点底部偏移。正值将使拖动点向底部位置的更低处移动。

2.3.2.5 事件

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

ScrollBar组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false
scroll	滚动位置已改变。 <ul style="list-style-type: none">• position: 新滚动位置，数值类型，值为最小位置到最大位置之间。

下列代码指示如何监听滚动事件：

```
mySB.addEventListener("scroll", this, "onListScroll");
function onListScroll(event:Object) {
    trace("scroll position: " + event.position);
    // 打印操作
}
```

2.3.2.6 指示器

手动设置一个独立的滚动指示器：

```
scrollBar.setScrollProperties(pageSize, minPos, maxPos, pageScrollSz);
scrollBar.positon = currPos;
```

设置滚动方向，使用`_rotation`属性在场景中创建组件时自动设置。如果滚动指示器组件不可旋转默认为水平方向，则需要明确的设置该值。

```
scrollBar.direction = "horizontal";
```

设置轨道在`scrollPage`模式下被点击时的滚动位置值。默认值为1：

```
scrollBar.trackScrollPageSize = pageSize;
```

2.3.3 Slider



图 28: 无皮肤滑动条Slider.

滑动条Slider(gfx.controls.Slider)显示了一个范围内的数值，用滚动块来表示值，通过拖动来改变值。

2.3.3.1 交互

滑动块可以被鼠标或类似控制器在滑动轨道边界内拖动。在轨道上点击可以立即移动滚动块到鼠标光标所在位置并设置为拖动状。当获得焦点时，左右方向键在对应的方向移动滚动块，而home和end键可以将滚动块移动到滚动条的开始和末尾处。

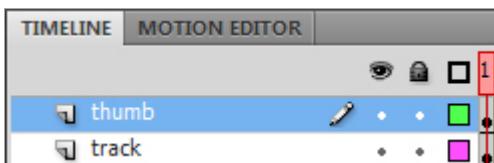
2.3.3.2 组件设置

使用CLIK Slider类的MovieClip必须具备下面所列的子单元。也列出了可选元素：

- **thumb:** CLIK 按钮类型，滑动快。
- **Track:** CLIK 按钮类型，滑动轨道边界决定了滑动块可以移动位置。

2.3.3.3 状态

与滚动指示器和滚动条类似，滑动条没有显性状态，使用子单元的状态包括：滚动块和轨道按钮组件。



□ 29:滑动条 Slider 时间轴

2.3.3.4 属性

滑动条Slider组件的检查属性如下所示：

visible	如果设置为false则隐藏组件
----------------	-----------------

| **disabled** | 如果设置为false则禁止组件 |

value	滑动条Slider显示的数值
minimum	滑动条范围内的最小值
maximum	滑动条范围内的最大值
snapping	如果设置为true, 滚动快将按照多被间隔进行移动
snapInterval	滚动间隔决定滚动块跳转间隔函数, 设置为false时该值无效
liveDragging	如果设置为true, 拖动滚动块时则滑动条Slider将产生一个改变事件, 滑动条只在拖动结束后产生事件改变
offsetLeft	拖动点向左偏移。正值将会向内挤压拖动点。
offsetRight	拖动点向右偏移。正值将会向内挤压拖动点。

2.3.3.5 事

所有的事件调用都接收一个Object参数, 包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

Slider组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false
focusIn	组件获得焦点
focusOut	组件失去焦点
change	改变滑动条Slider值

下例显示了如何监听滑动条Slider值的改变:

```
mySlider.addEventListener("change", this, "onValueChange");
function onValueChange(event:Object) {
    trace("slider value: " + event.target.value);
    // 执行操作
}
```

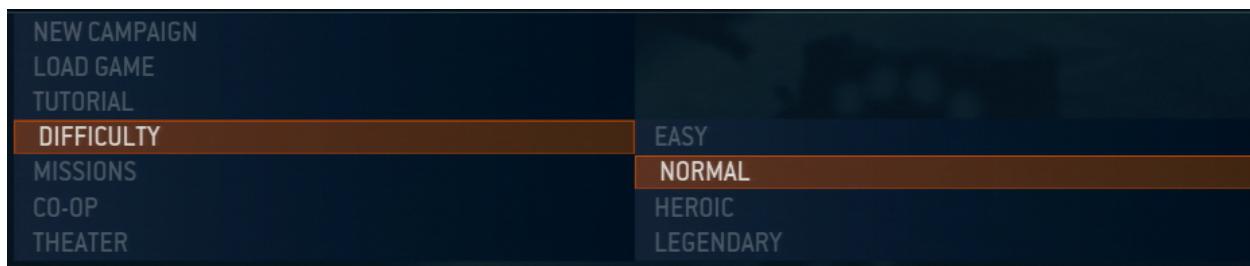
2.4 列表类型

CLIK列表类型包括NumericStepper、 OptionStepper、 ScrollingList、 TileList 和DropdownMenu组件。所有组件，除了NumericStepper，都用DataProvider来管理信息并显示。ListItemRenderer组件也包括在此目录中，因为ScrollingList 和TileList组件需要用来显示列表项目。

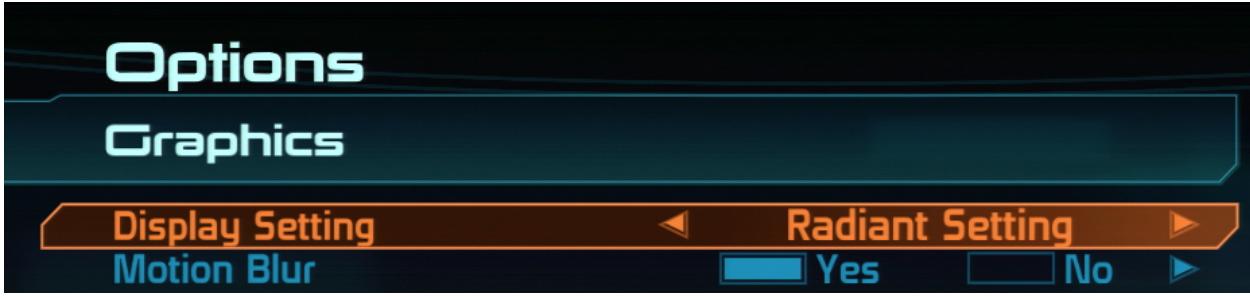
NumericStepper和 OptionStepper组件只用一次显示一个单元，而ScrollingList 和TileList能显示多个单元。最后两个组件可以支持ScrollIndicator或 ScrollBar组件。DropdownMenu组件在静止状态显示一个单元，但是在打开时使用ScrollingList 或TileList可以显示多个单元。



□ 30:来自 *Mercenaries 2*的滚动指示器滚动列表实例



□ 31:来自 *Halo Wars*的'Difficulty Settings'下拉菜单实例



□ 32:来自 *Mass Effect*的'Display Setting'选项实例

2.4.1 NumericStepper



□ 33: 无皮肤Numeric Stepper.

NumericStepper

(gfx.controls.NumericStepper)显示在分配范围内一个数字，支持任意步长值的增加和减少。

2.4.1.1 用户交互

NumericStepper包括两个方向键可以通过鼠标或类似控制器点击以改变数字值。当获得焦点时，数字值可以通过键盘左右方向键或类似控制器改变。这些键可以根据步长增加和减少当前值。按下(Home)和(End)键或类似控制器将在对应的最大值和最小值之间改变数字值。

2.4.1.2 组件设置

使用CLIK NumericStepper类的MovieClip必须具备下面所列的子单元。也列出了可选元素：

- **textField:** TextField 类型，用来显示当前值。
- **nextBtn:** CLIK 按钮Button类型，点击可以根据步长增加当前值。
- **prevBtn:** CLIK 按钮Button类型，点击可以根据步长减少当前值。

2.4.1.3 状态

NumericStepper组件支持三种状态，基于焦点和禁止属性：

- **default**或 enabled 状态；

- **focused** 状态，突出显示textField 区域；
- **disabled** 状态；

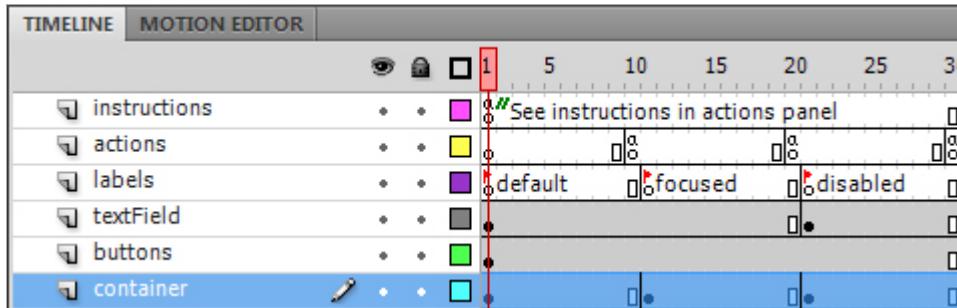


图34: NumericStepper 时间轴

2.4.1.4 检查属性

从数字步长NumericStepper组件继承来的动画剪辑MovieClip将有以下检查属性：

visible	如果设置为false则隐藏组件
disabled	如果设置为false则禁止组件
value	NumericStepper.显示的数值
minimum	NumericStepper数值范围内的最小值
maximum	NumericStepper数值范围内的最大值

2.4.1.5 事

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

NumericStepper组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false
change	NumericStepper数值发生改变
stateChange	NumericStepper焦点或禁止属性发生改变。 <ul style="list-style-type: none"> • state: 新状态名，String类型，值为“default”、“focused” 或 “disabled”。

下例显示了对NumericStepper值改变的监听：

```
myNS.addEventListener("change", this, "onValueChange");
function onValueChange(event:Object) {
    trace("ns value: " + event.target.value);
    // □行操作
}
```

2.4.1.6 按钮

改变增加/减少间隔:

```
ns.stepSize = 0.5;
```

增加数值后缀，例如:

```
ns.labelFunction = function(value:Number) {
    switch(value) {
        case 1:
            return value + "st";
        default:
            return value + "th";
    }
}
```

2.4.2 OptionStepper



□ 35: 无皮肤OptionStepper.

OptionStepper

(`gfx.controls.OptionStepper`),与`NumericStepper`类似，用来显示一个值，但可以显示更多信息。使用数据源`dataProvider`实例查询当前值；因此支持不同类型元素的任意数值。应用`OptionStepper`的选择项索引属性通过代码对显示值进行设置，该索引是附在数据提供者中的一个从零开始的索引。数据源`dataProvider`通过代码进行指派，如下例所示：

```
optionStepper.dataProvider = ["item1", "item2", "item3", "item4"];
```

2.4.2.1 用交互

与NumericStepper类似，OptionStepper包括两个方向键，可以通过鼠标或类似控制器点击改变当前值。当获得焦点时，当前值可以通过左右方向键或类似控制器进行改变。这些键按照顺序向前或者向后改变当前值。按下(Home)和(End)键或类似控制器可以将当前值改变为数据源dataProvider的第一个和最后一个单元。

2.4.2.2 组件设置

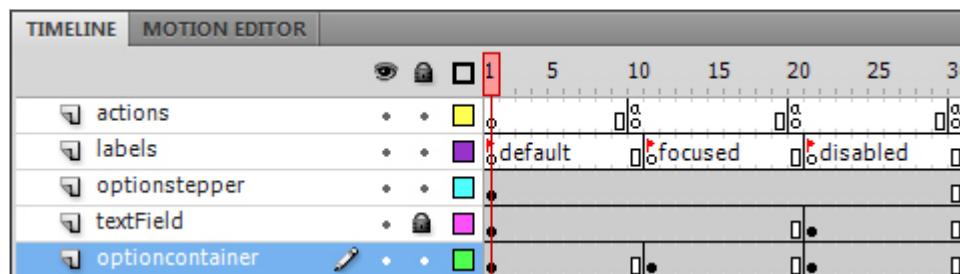
使用CLIK NumericStepper类的动画剪辑MovieClip必须具备下面所列的子单元。也列出了可选元素：

- **textField:** TextField类型，用来显示当前值。
- **nextBtn:** CLIK按钮 Button类型，改变当前值为数据源dataProvider中的下一个值。
- **prevBtn:** CLIK 按钮 Button类型，当前值为数据源dataProvider中的上一个值。

2.4.2.3 状态

OptionStepper组件支持三种状态，基于焦点和禁止属性：

- **default**或 enabled 状态；
- **focused** 状态，突出显示textField 区域；
- **disabled** 状态；



□ 36: OptionStepper 时间轴

2.4.2.4 检查属性

来自选项步长OptionStepper组件的视频剪辑MovieClip具有以下检查属性。

visible	如果设置为false则隐藏组件
disabled	如果设置为false则禁止组件

2.4.2.5 事件

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

OptionStepper组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false
change	OptionStepper值发生改变
stateChange	OptionStepper焦点或禁止属性发生改变。 <ul style="list-style-type: none"> • state: 新状态名称, String类型, 值为“default”、“focused”或“disabled”。

下列显示了OptionStepper值改变的监听:

```
myOS.addEventListener( "change" , this , "onValueChange" );
function onValueChange( event:Object ) {
    trace( "os value: " + event.target.selectedItem );
    // 执行操作
}
```

2.4.3 ListItemRenderer



图 37:无皮肤 ListItemRenderer.

列表项渲染器ListItemRenderer

(gfx.controls.ListItemRenderer)从CLIK按钮Button类继承而来，扩展了列表相关特性，在容器组件中需要用到。但是，不是作为一个单独组建设计，只与滚动列表ScrollingList、标题列表TitleList和下拉菜单DropdownMenu组件共同使用。

2.4.3.1 交互

由于ListItemRenderer从按钮Button组件继承而来，与按钮具有相同的用户交互如使用鼠标点击。滚动鼠标光标到ListItemRenderer或将光标移开都将对组件产生影响，拖动鼠标光标效果也一样。ListItemRenderer的容器组件定义了键盘或类似控制器的交互。

2.4.3.2 组件设置

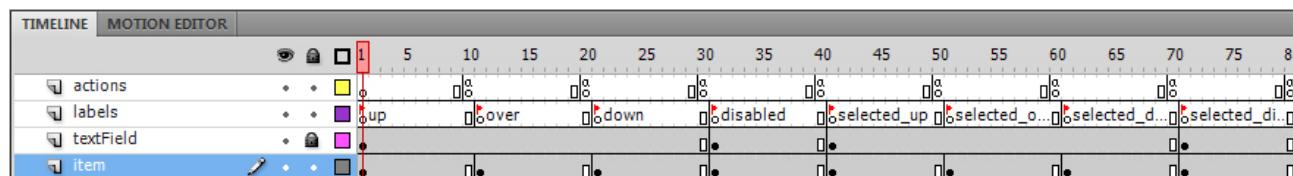
使用CLIK ListItemRenderert类的动画剪辑MovieClip必须具备下面所列的子单元。也列出了可选元素：

- **textField:** (可选) TextField 类型，列表项标签。
- **focusIndicator:** (可选) MovieClip类型，一个独立的动画剪辑
MovieClip用来显示焦点状态。如果被使用，该动画剪辑必须拥有两个帧分别命名为：“show”和“hide”。

2.4.3.3 状态

由于可以在一个容器组件内部被选择，列表项渲染器ListItemRenderer需要关键帧为**selected**设置来表示其选择状态。组件状态包括：

- **up** 或默认状态；
- 当鼠标箭头在组件上方或者获得焦点时为**over** 状态；
- 当按钮被点击时候为**down** 状态；
- **disabled** 状态；
- **selected_up** 或者默认状态；
- 当鼠标箭头位于组件上方或获得焦点时为**selected_over** 状态；
- 当按钮被按下时为 **selected_down** 状态；
- **selected_disabled** 状态；



□ 38: ListItemRenderer 时间轴

这里为选择按钮ListItemRenderer所需要的最少关键帧设置。按钮Button组件支持状态和关键帧的扩展设置，与ListItemRenderer组件相同，在[CLIK 按钮入门](#)文档中有详细描述。

2.4.3.4 检查属性

由于列表项渲染器ListItemRenderer由一个容器组件按控制，用户无需手动配置，值包含了按钮的一小部分检查属性。

label	ListItemRenderer标签设置
visible	如果设置为false则隐藏按钮
disabled	如果设置为true则禁止按钮

2.4.3.5 事件

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

列表项渲染器ListItemRenderer组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false
focusIn	组件获得焦点
focusOut	组件失去焦点
select	所选属性已改变。 <ul style="list-style-type: none">• Selected: 按钮的选择状态，true为被选择，为布尔类型。
stateChange	按钮状态已改变 State：按钮新状态。String类型。值为“up”、“over”、“down”等。 参考 CLIK按钮入门 文档获取详细的状态列表信息。
rollOver	鼠标箭头在按钮上方滚动。 <ul style="list-style-type: none">• mouseIndex: 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。
rollOut	鼠标箭头从按钮移开。 <ul style="list-style-type: none">• mouseIndex: 用户产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）数值类型。值为0-3。
press	按钮被点击。 <ul style="list-style-type: none">• mouseIndex: 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。数值类型。值为0-3。
doubleClick	按钮被双击。只在 doubleClickEnabled 属性为true时被触发。 <ul style="list-style-type: none">• mouseIndex: 用来产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）。

	箭头环境）。数值类型。值为0-3。
click	按钮被点击。 <ul style="list-style-type: none"> • <i>mouseIndex</i>: 用来产生事件的鼠标箭头索引（只应用在多鼠标-箭头环境）。数值类型。值为0-3。
dragOver	鼠标箭头拖动到按钮上方（鼠标左键被按下） <ul style="list-style-type: none"> • <i>mouseIndex</i>: 用来产生事件的鼠标箭头索引（只应用在多鼠标-箭头环境）。数值类型。值为0-3。
dragOut	鼠标箭头从按钮拖开（鼠标左键被按下）。 <ul style="list-style-type: none"> • <i>mouseIndex</i>: 用来产生事件的鼠标箭头索引（只应用在多鼠标-箭头环境）。数值类型。值为0-3。
releaseOutside	鼠标箭头从按钮拖开鼠标左键松开。 <ul style="list-style-type: none"> • <i>mouseIndex</i>: 用来产生事件的鼠标箭头索引（只应用在多鼠标-箭头环境）。数值类型。值为0-3。

2.4.4 ScrollingList



图 39: 无皮肤滚动列表ScrollingList.

滚动列表ScrollingList

(gfx.controls.ScrollingList)为一个组件可以滚动其元素。可以自身展示列表项或者使用现存的场景中的列表项。可以附加滚动指示器ScrollIndicator

或滚动条ScrollBar组件提供滚动反馈和控制功能。该组件可与数据源dataProvider一起使用。数据源通过代码指派，如下面实例所示：

```
scrollingList.dataProvider = ["item1", "item2", "item3", "item4"];
```

默认情况下滚动列表crollingList使用列表项渲染器ListItemRenderer组件作为内容。因此列表项渲染器必须在FLA文件库中存在，除非列表项渲染器检查属性改变成另外一个组件。见检查属性小节获得更多信息。
。

2.4.4.1 用交互

点击一个列表项或者一个附属的滚动条ScrollBar实例将焦点传递到滚动列表ScrollingList组件。当获得焦点，按下向上向下方向键或类似的控制器控制列表选项的滚动选择一个元素。如果没有选中元素，则默认选中顶部元素。如果光标位于滚动列表ScrollingList顶部则鼠标滚轮能够在列表中滚动。

在边界上的滚动行为由滚动列表ScrollingList的*wrapping*属性决定，无检查项。如果*wrapping*设置为“*normal*”，当选项达到列表开始处或者结束处则焦点将离开组件。如果*wrapping*设置为“*wrap*”，则选项将围绕开始处或结束处。如果*wrapping*设置为“*stick*”，则选项在达到数据结尾时停止，焦点不转移到相邻组件。

按下键盘(Page Up) 和 (Page

down)或类似控制器将按页滚动选项，例如，列表中的可视选项数。按下(Home) 和 (End)键，或者类似控制器，将滚动列表到相应元素的开始和末尾处。与附属滚动条ScrollBar组件交互将按预期影响滚动列表ScrollingList。见滚动条ScrollBar小节了解其用户交互功能。

开发者可以简单得将游戏控制杆映射到键盘和鼠标控制键。例如，键盘方向键通常引导控制器上的D-Pad。这个对应关系可以使CLIK构建的UI界面工作在更加广泛的平台之上。

2.4.4.2 组件设置

滚动列表ScrollingList不需要任何子单元，但是，在场景上的滚动列表ScrollingList组件上放置一个组件或者调整组件大小时一个可视化背景就能起到辅助作用。

2.4.4.3 样

滚动列表ScrollingList组件支持三种状态，基于焦点和禁止属性：

- **default** 或enable状态；
- **focused**状态，通常突出显示组件边框区域；
- **disabled** 状态。

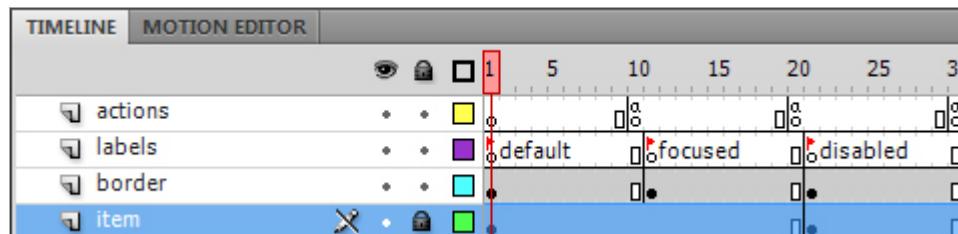


图 40:滚动列表 ScrollingList 时间轴

2.4.4.4 检查属性

来自滚动列表ScrollingList 组件的视频剪辑MovieClip具有以下检查属性。

visible	如果设置为false则隐藏组件。不隐藏附属滚动条或其他任何扩展列表项图形
disabled	如果设置为true则禁止组件，不禁止附属滚动条和列表项（包括内部创建和外部渲染）
itemRenderer	列表项渲染器ListItemRenderer符号名称。用于创建内部列表项实例，如果设置了 <i>rendererInstanceName</i> 属性则不产生影响
rendererInstanceName	列表项渲染器扩展前缀，与滚动列表ScrollingList组件一起使用。场景中的列表项实例必须用该属性值作为前缀。如果属性设置为‘r’，则所有本组件使用的列表项实例必须具有以下值：‘r1’、‘r2’、‘r3’……第一个项应该为数值1。
scrollBar	场景中的滚动条ScrollBar组件实例名称或符号名。如果指定了实例名称，则滚动列表ScrollingList将作为实例的钩，如果符号名未指定，将由滚动列表ScrollingList创建一个符号实例。
margin	内部创建的列表组件和列表项组件的页面边缘。如果设置了 <i>rendererInstanceName</i> 属性该值不产生影响。
rowHeight	内部创建的列表项实例高度，如果设置了 <i>rendererInstanceName</i> 属性该值不产生影响。
paddingTop	在列表项的顶部进行额外填充。如果设置了 <i>rendererInstanceName</i> 属性，则这一数值无效。不影响自动生成的滚动条。
paddingBottom	在列表项的底部进行额外填充。如果设置了 <i>rendererInstanceName</i> 属性，则这一数值无效。不影响自动生成的滚动条。
paddingLeft	在列表项的左侧进行额外填充。如果设置了 <i>rendererInstanceName</i> 属性，则这一数值无效。不影响自动生成的滚动条。
paddingRight	在列表项的右侧进行额外填充。如果设置了 <i>rendererInstanceName</i> 属性，则这一数值不会产生影响。不影响自动生成的滚动条。
thumbOffsetTop	滚动条拖动点顶部偏移。如果列表未能自动创建一个滚动条示例，则该属性不会产生影响。
thumbOffsetBottom	滚动条拖动点底部偏移。如果列表未能自动创建一个滚动条示例，则该属性不会产生影响。
thumbSizeFactor	滚动条拖动点的页面尺寸因素。如果某一指定因素的数值大于1.0，则拖动点的尺寸将会增加。小于1.0的正值将会缩小拖动点的尺寸。如果滚动条未附在列表中，则这一数值不会产生影响。

2.4.4.5 事

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

滚动列表ScrollingList组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行中设置为false
focusIn	组件获得焦点
focusOut	组件失去焦点
change	选择序号发生变化。 <ul style="list-style-type: none">• <i>index:</i> 新选择序号，数值类型，值为0到列表项数量最小为1。
itemPress	一个列表项被按下。 <ul style="list-style-type: none">• <i>renderer:</i> 按下的列表项，CLIK按钮类型。• <i>item:</i> 与列表项关联的数据，该值从列表数据源dataProvider重新得到。AS2对象类型。• <i>index:</i> 与数据源dataProvider 关联的列表项序号。数值类型，值为0到列表项数量最小为1。• <i>mouseIndex:</i> 用来产生事件的鼠标光标序号（只引用在多鼠标光标环境），数值类型，值为0到3。
itemClick	一个列表项被点击。 <ul style="list-style-type: none">• <i>renderer:</i> 按下的列表项，CLIK按钮类型。• <i>item:</i> 与列表项关联的数据，该值从列表数据源dataProvider重新得到。AS2对象类型。• <i>index:</i> 与数据源dataProvider 关联的列表项序号。数值类型，值为0到列表项数量最小为1。• <i>mouseIndex:</i> 用来产生事件的鼠标光标序号（只引用在多鼠标光标环境），数值类型，值为0到3。
itemDoubleClick	一个列表项被双击。 <ul style="list-style-type: none">• <i>renderer:</i> 按下的列表项，CLIK按钮类型。• <i>item:</i> 与列表项关联的数据，该值从列表数据源dataProvider重新得到。AS2对象类型。• <i>index:</i> 与数据源dataProvider 关联的列表项序号。数值类型，值为0到列表项数量最小为1。• <i>mouseIndex:</i> 用来产生事件的鼠标光标序号（只引用在多鼠标光标环境），数值类型，值为0到3。

	环境）， 数值类型， 值为0到3。
itemRollOver	鼠标光标在列表项上方滚动。 <ul style="list-style-type: none"> • <i>renderer</i>: 按下的列表项， CLIK按钮类型。 • <i>item</i>: 与列表项关联的数据， 该值从列表数据源dataProvider重新得到。 AS2对象类型。 • <i>index</i>: 与数据源dataProvider 关联的列表项序号。 数值类型， 值为0到列表项数量最小为1。 • <i>mouseIndex</i>: 用来产生事件的鼠标光标序号（只引用在多鼠标光标环境）， 数值类型， 值为0到3。
itemRollOut	鼠标光标滚动出列表项。 <ul style="list-style-type: none"> • <i>renderer</i>: 按下的列表项， CLIK按钮类型。 • <i>item</i>: 与列表项关联的数据， 该值从列表数据源dataProvider重新得到。 AS2对象类型。 • <i>index</i>: 与数据源dataProvider 关联的列表项序号。 数值类型， 值为0到列表项数量最小为1。 • <i>mouseIndex</i>: 用来产生事件的鼠标光标序号（只引用在多鼠标光标环境）， 数值类型， 值为0到3。

下例显示了如何监听列表项的点击动作：

```
myList.addEventListener("itemClick", this, "onItemClicked");
function onItemClicked(event:Object) {
    trace("list item was clicked: " + event.renderer);
    // 执行操作
}
```

2.4.4.6 擦除

显示一个来自复杂对象集合的标签：

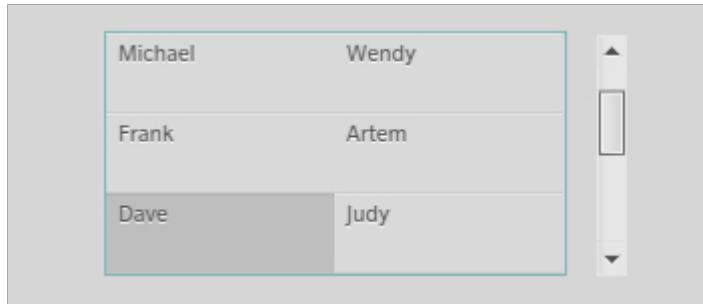
```
// ScrollingList将自动使用属性名称 'label'
// 如果在项目对象中找到：
list.dataProvider = [{label: "one", data:1}, {label: "two", data:2}];

// 但是如果项目对象具有不同的标签属性， 如下所示，
// 则列表可以配置成使用该属性来代替：
list.labelField = "name";
list.dataProvider = [{name: "one", data:1}, {name: "two", data:2}];

// 如果从项目对象上构建一个标签逻辑上比较复杂并需要一个函数，
```

```
//则设置labelFunction属性
list.labelFunction = function(itemObj:Object):String {
    // 逻辑上构建一个标签
}
list.dataProvider = [{p1: "foo", p2: 1}, {p1: "bar", p2: 2}];
```

2.4.5 TileList



□ 41: 无皮肤标题列表TitleList.

标题列表TitleList

(gfx.controls.TileList)与滚动列表ScrollingList类似，为一个可以滚动其元素的组件。可以自身展示列表项或者使用现存的场景中的列表项。可以附加滚动指示器ScrollIndicator或滚动条ScrollBar组件提供滚动反馈和控制功能。标题列表TileList和滚动列表ScrollingList的区别为标题列表同时支持多个行和列，列表项选择可以向四个主要方向移动。该组件可与数据源dataProvider一起使用。数据源通过代码指派，如下面实例所示：

```
tileList.dataProvider = ["item1", "item2", "item3", "item4", "item5"];
```

默认情况下标题列表TitleList使用列表项渲染器ListItemRenderer组件作为内容。因此列表项渲染器必须在FLA文件库中存在，除非列表项渲染器检查属性改变成另外一个组件。见检查属性小节获得更多信息。

2.4.5.1 用交互

点击一个列表项或者一个附属的滚动条ScrollBar实例将焦点传递到标题列表TileList组件。当获得焦点，按下向上向下方向键或者类似的控制器，如果包含多行逐个单元垂直滚动列表选项。也可以用向左向右方向键或类似控制器，如果包含多列逐个单元水平滚动列表选项。如果标题列表TileList包含多行和多列，四个方向键或类似控制器可以用来导航列表项。如果没有选中元素，顶部元素自动作为默认选项。如果光标位于标题列表边界上鼠标滚轮可以滚动列表。

按下键盘(Page Up) 和 (Page

down) 或类似控制器将按页滚动选项，例如，列表中的可视选项数。按下(Home) 和

(End)键，或者类似控制器，将滚动列表到相应元素的开始和末尾处。与附属滚动条ScrollBar组件交互将按预期影响标题列表TileList。见滚动条ScrollBar小节了解其用户交互功能。

2.4.5.2 组件设置

标题列表TileList不需要任何子单元，但是，在场景上的标题列表TileList组件上放置一个组件或者调整组件大小时一个可视化背景就能起到帮助作用。

2.4.5.3 样式

标题列表TileList组件支持三种状态，基于焦点和禁止属性：

- **default** 或enable状态；
- **focused**状态，通常突出显示组件边框区域；
- **disabled** 状态。

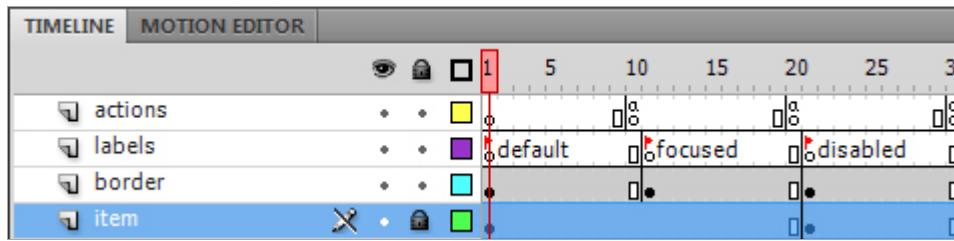


图 42: 标题列表TileList 时间轴

2.4.5.4 检查属性

来自标题列表TileList组件的视频剪辑MovieClip具有以下检查属性：

Visible	如果设置为false则隐藏组件。不隐藏附属滚动条或其他任何扩展列表项目图型
Disabled	如果设置为true则禁止组件，不禁止附属滚动条和列表项（包括内部创建和外部渲染）
itemRenderer	列表项渲染器ListItemRenderer符号名称。用于创建内部列表项实例，如果设置了rendererInstanceName属性则不产生影响
rendererInstanceName	列表项渲染器扩展前缀，与标题列表TileList组件一起使用。场景中的列表项实例必须用该属性值作为前缀。如果属性设置为‘r’，则所有本组件使用的列表项实例必须具有以下值：‘r1’、‘r2’、‘r3’

第一个项应该为数值1。
scrollbar	场景中的滚动条ScrollBar组件实例名称或符号名。如果指定了实例名称，则标题列表TileList将作为实例的钩，如果符号名未指定，将由标题列表TileList创建一个符号实例。
margin	内部创建的列表组件和列表项组件的页面边缘。如果设置了 <i>rendererInstanceName</i> 属性该值不产生影响。
rowHeight	内部创建的列表项实例高度，如果设置了 <i>rendererInstanceName</i> 属性该值不产生影响。
columnWidth	内部创建的列表项实例宽度，如果设置了 <i>rendererInstanceName</i> 属性该值不产生影响。
externalColumnCount	当设置了 <i>rendererInstanceName</i> 属性，该值用来指示外部渲染器使用的列数的标题列表TileList
Direction	滚动方向，语义上行和列不根据该值进行改变。

2.4.5.5 事件

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

标题列表TileList组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

Show	组件可视属性在运行时已设置为true
Hide	组件可视属性在运行时已设置为false
focusIn	组件获得焦点
focusOut	组件失去焦点
Change	选择序号发生变化。 <ul style="list-style-type: none"> • <i>index:</i> 新选择序号，数值类型，值为0到列表项数量最小为1.
itemPress	列表口被按下。 <ul style="list-style-type: none"> • <i>renderer:</i> 按下的列表项，CLIK按钮类型。 • <i>item:</i> 与列表项关联的数据，该值从列表数据源dataProvider重新得到。AS2对象类型。 • <i>index:</i> 与数据源dataProvider关联的列表项序号。数值类型，值为0到列表项数量最小为1。 • <i>mouseIndex:</i> 用来产生事件的鼠标光标序号（只引用在多鼠标光标环境），数值类型，值为0到3。

itemClick	列表项被点击。
	<ul style="list-style-type: none"> • <i>renderer</i>: 按下的列表项, CLIK按钮类型。 • <i>item</i>: 与列表项关联的数据, 该值从列表数据源dataProvider重新得到。AS2对象类型。 • <i>index</i>: 与数据源dataProvider 关联的列表项序号。数值类型, 值为0到列表项数量最小为1。 • <i>mouseIndex</i>: 用来产生事件的鼠标光标序号(只引用在多鼠标光标环境), 数值类型, 值为0到3。
itemDoubleClick	列表项被双击。
	<ul style="list-style-type: none"> • <i>renderer</i>: 按下的列表项, CLIK按钮类型。 • <i>item</i>: 与列表项关联的数据, 该值从列表数据源dataProvider重新得到。AS2对象类型。 • <i>index</i>: 与数据源dataProvider 关联的列表项序号。数值类型, 值为0到列表项数量最小为1。 • <i>mouseIndex</i>: 用来产生事件的鼠标光标序号(只引用在多鼠标光标环境), 数值类型, 值为0到3。
itemRollOver	鼠标光标在列表项上方滚动。
	<ul style="list-style-type: none"> • <i>renderer</i>: 按下的列表项, CLIK按钮类型。 • <i>item</i>: 与列表项关联的数据, 该值从列表数据源dataProvider重新得到。AS2对象类型。 • <i>index</i>: 与数据源dataProvider 关联的列表项序号。数值类型, 值为0到列表项数量最小为1。 • <i>mouseIndex</i>: 用来产生事件的鼠标光标序号(只引用在多鼠标光标环境), 数值类型, 值为0到3。
itemRollOut	鼠标光标移开列表项。
	<ul style="list-style-type: none"> • <i>renderer</i>: 按下的列表项, CLIK按钮类型。 • <i>item</i>: 与列表项关联的数据, 该值从列表数据源dataProvider重新得到。AS2对象类型。 • <i>index</i>: 与数据源dataProvider 关联的列表项序号。数值类型, 值为0到列表项数量最小为1。 • <i>mouseIndex</i>: 用来产生事件的鼠标光标序号(只引用在多鼠标光标环境), 数值类型, 值为0到3。

下例显示如何判断标题列表TileList接收焦点:

```
myList.addEventListener("focusIn", this, "onListFocused");
function onListFocused(event:Object) {
    trace("tile list was focused!");
}
```

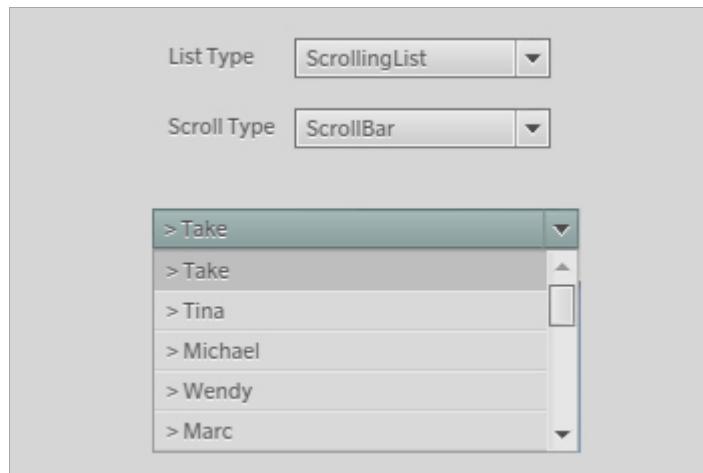
```
// □行操作  
}
```

2.4.5.6 换行

显示一个来自复杂对象集合的标签:

```
// ScrollingList将自动使用属性名称 'label'  
//如果在项目对象中找到  
list.dataProvider = [{label: "one", data:1}, {label: "two", data:2}];  
  
// 但是如果项目对象具有不同的标签属性, 如下所示,  
// 则列表可以配置成使用该属性来代替:  
list.labelField = "name";  
list.dataProvider = [{name: "one", data:1}, {name: "two", data:2}];  
  
// 如果从项目对象上构建一个标签逻辑上比较复杂并需要一个函数,  
// 这设置labelFunction属性  
list.labelFunction = function(itemObj:Object):String {  
    // 逻辑上构建一个标签Logic to construct a label  
}  
list.dataProvider = [{p1: "foo", p2: 1}, {p1: "bar", p2: 2}];
```

2.4.6 DropdownMenu



□ 43: 无皮肤下拉菜单DropdownMenu.

下拉菜单DropdownMenu

(gfx.controls.DropdownMenu)包含了按钮和列表的行为。点击组件打开列表包含了选择元素。下拉菜单

DropdownMenu

只在静止状态显示选择元素。可以配置成使用滚动列表`ScrollingList`或标题列表`TileList`，并可以配上滚动条`ScrollBar`或滚动指示器`ScrollIndicator`。列表与数据源`dataProvider`相关联，下拉菜单`DropdownMenu`列表元素与数据源`dataProvide`进行连接。数据源`dataProvide`通过代码指派，如下例所示：

```
dropdownMenu.dataProvider = ["item1", "item2", "item3", "item4"];
```

默认情况下下拉菜单`DropdownMenu`使用列表项渲染器`ListItemRenderer`组件作为内容。因此下拉菜单`DropdownMenu`和列表项渲染器`ListItemRenderer`必须在FLA文件库中存在，除非下拉属性改变成另外一个组件。参考检查属性小节获得更多信息。

也需注意默认情况下下拉菜单`DropdownMenu`在列表元素中不附带一个滚动条，滚动条`ScrollBar`或滚动指示器`ScrollIndicator`必须通过代码附加到下拉菜单`DropdownMenu`列表元素。见提示和技巧小节获得更多信息。

2.4.6.1 用交互

点击下拉菜单`DropdownMenu`实例或者按下(`Enter`)键或类似控制器将打开可选元素列表。当打开时焦点转移到该列表，如在用户交互小节所描述用户能够与列表中`ScrollingList`、`TileList`和`ScrollBar`组件进行交互。点击一个列表项将其选中，关闭列表并在下拉菜单`DropdownMenu`组件中显示选择项。在列表边界外用鼠标点击将自动关闭列表，焦点将传递回下拉菜单组件。

2.4.6.2 组件设置

下拉菜单`DropdownMenu`继承了按钮组件的绝大多数功能。从而视频剪辑`MovieClip`使用下拉菜单类必须有以下名称的子单元。列表和滚动条动态创建，注出了对应的可选单元：

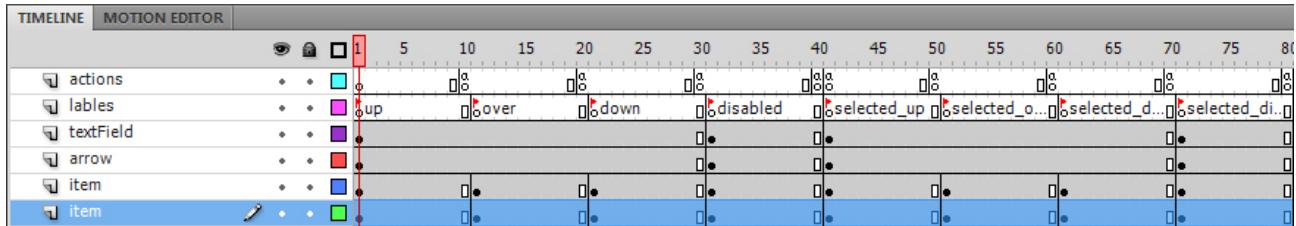
- **textField:** (可选) `TextField` 类型，按钮标签。
- **focusIndicator:** (可选)
`MovieClip`类型，一个单独的视频剪辑用来显示焦点状态，如果被使用，则视频剪辑必须拥有两个帧名为：“show” 和 “hide”。

2.4.6.3 状态

下拉菜单`DropdownMenu`打开时为选中状态，因此需要与`ToggleButton`和`CheckBox`具有相同的状态来指示选中状态，这些状态包括：

- **up** 或默认状态：
- 当鼠标箭头在组件上方或者获得焦点时为**over** 状态；
- 当按钮被点击时候为**down** 状态；

- **disabled** 状态;
- **selected_up** 或者默认状态;
- 当鼠标箭头位于组件上方或获得焦点时为**selected_over** 状态;
- 当按钮被按下时为 **selected_down** 状态;
- **selected_disabled** 状态;



□ 44: 下拉菜单DropdownMenu 时间轴

这里为选择按钮下拉菜单DropdownMenu所需要的最少关键帧设置。按钮Button组件支持状态和关键帧的扩展设置，与DropdownMenu组件相同，在[CLIK 按钮入门](#)文档中有详细描述。

2.4.6.4 检查属性

下拉菜单DropdownMenu组件的检查属性为：

Visible	如果设置为false则隐藏组件
Disabled	如果设置为false则禁止组件
Dropdown	列表组件（ScrollingList 或 TileList）的符号名称，与下拉菜单DropdownMenu组件一起使用
dropdownWidth	下拉列表宽度，如果该值为-1，则下拉菜单将依据组件宽度决定列表宽度
itemRenderer	下拉列表的项目渲染符号名称，由下拉列表实例创建
scrollbar	下拉列表滚动条的符号名，由下拉列表实例创建。如果值为空，则下拉列表无滚动条
Margin	列表部件与内部创建的列表项之间的边距。该边距还会对自动生成的滚动条产生影响。
paddingTop	在列表项的顶部进行额外填充。不影响自动生成的滚动条。
paddingBottom	在列表项的底部进行额外填充。不影响自动生成的滚动条。
paddingLeft	在列表项的左侧进行额外填充。不影响自动生成的滚动条。
paddingRight	在列表项的右侧进行额外填充。不影响自动生成的滚动条。
thumbOffsetTop	滚动条拖动点顶部偏移。如果没有自动生成滚动条示例，则这一属性不会产生影响。
thumbOffsetBottom	滚动条拖动点底部偏移。如果没有自动生成滚动条示例，则这一属性不会产生影响。
thumbSizeFactor	滚动条thumb页面尺寸因素。如果某一指定因素的数值大于1.0，则thumb

	的尺寸将会增加。小于1.0的正值将会缩小thumb的尺寸。如果滚动条未附在列表中，则这一数值不会产生影响。
offset	下拉列表从下拉按钮位置进行水平偏移。正值将列表移动到下拉按钮水平位置右侧。
offset	下拉列表从下拉按钮发生的垂直偏移。正值将列表从按钮中移除。
Extent	用于连接 offsetX 的偏移宽度。如果 dropdownWidth 属性的设置值不等于1，则该数值不会产生影响。
Direction	列表的开启方向。有效值为“上”和“下”。

2.4.6.5 事件

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

DropdownMenu组件产生的事件列表如下所示。除change事件意外，与Button组件类似，事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
Hide	组件可视属性在运行时已设置为false
focusIn	组件获得焦点
focusOut	组件失去焦点
change	选择序号发生变化。 <ul style="list-style-type: none"> • <i>index</i>: 新选择序号，数值类型，值为0到列表项数量最小为1. • <i>data</i> : 选择序号中与列表项关联的数据，AS2对象类型
select	所选属性已改变。 <ul style="list-style-type: none"> • <i>Selected</i>: 按钮的选择状态，true为被选择，为布尔类型。
stateChange	按钮状态已改变 State : 按钮新状态。String类型。值为“up”、“over”、“down”等。 参考 CLIK按钮入门 文档获取详细的状态列表信息。
rollOver	鼠标箭头在按钮上方滚动。 <ul style="list-style-type: none"> • <i>mouseIndex</i>: 用来产生事件的鼠标箭头索引（只应用在多鼠标光标环境）。数值类型。值为0-3。
rollOut	鼠标箭头从按钮移开。 <ul style="list-style-type: none"> • <i>mouseIndex</i>: 用户产生事件的鼠标箭头索引（只应用在多鼠标箭头环境）数值类型。值为0-3。
press	按钮被点击。

	<ul style="list-style-type: none"> • <i>mouseIndex</i>: 用来产生事件的鼠标箭头索引（只应用在多鼠标光标环境）。数值类型。值为0-3。
doubleClick	按钮被双击。只在 <i>doubleClickEnabled</i> 属性为true时被触发。 <ul style="list-style-type: none"> • <i>mouseIndex</i>: 用来产生事件的鼠标箭头索引（只应用在多鼠标光标环境）。数值类型。值为0-3。
click	按钮被点击。 <ul style="list-style-type: none"> • <i>mouseIndex</i>: 用来产生事件的鼠标箭头索引（只应用在多鼠标光标环境）。数值类型。值为0-3。
dragOver	鼠标箭头拖动到按钮上方（鼠标左键被按下） <ul style="list-style-type: none"> • <i>mouseIndex</i>: 用来产生事件的鼠标箭头索引（只应用在多鼠标光标环境）。数值类型。值为0-3。
dragOut	鼠标箭头从按钮拖开（鼠标左键被按下）。 <ul style="list-style-type: none"> • <i>mouseIndex</i>: 用来产生事件的鼠标箭头索引（只应用在多鼠标光标环境）。数值类型。值为0-3。
releaseOutside	鼠标箭头从按钮拖开鼠标左键松开。 <ul style="list-style-type: none"> • <i>mouseIndex</i>: 用来产生事件的鼠标箭头索引（只应用在多鼠标-箭头环境）。数值类型。值为0-3。

2.4.6.6 捕获

判断下拉菜单开打和关闭:

```
dropdown.addEventListener("click", this, "onClick");
function onClick(e:Object) {
    if (e.target.isOpen) {
        //当open时执行操作
    }
}
```

在运行时动态创建下拉菜单DropdownMenu:

```
//确保代码中具有链接;
//意味着设计的符号/组件存在与FLA库中。
//例如: 下列中代码需要用到DropdownMenu、ScrollingList
//和 ScrollBar组件。
attachMovie("DropdownMenu", "dd", this.getNextHighestDepth(),
            {dropdown: "ScrollingList"});
dd.dataProvider = ["one", "two", "three", "four", "five", "six"];

// 安装一个滚动条或滚动指示器到下拉列表比较复杂,
//因为需要一个延时设置下面代码中有所描述。
```

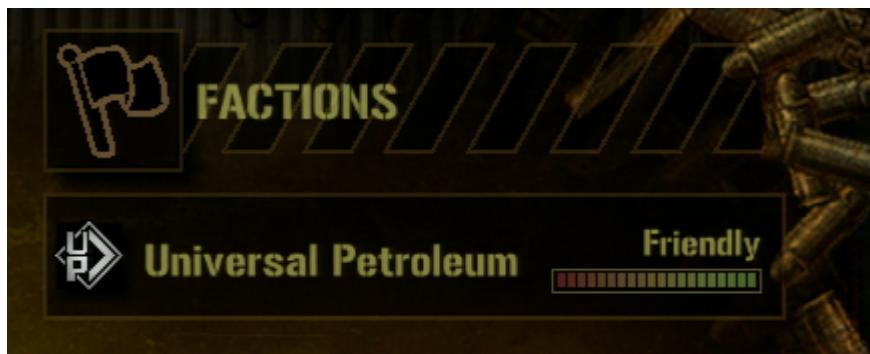
```
//使下拉实例在附加滚动条到列表前先显示列表，这里就需要一个延时：  
onEnterFrame = function() {  
    dd.dropdown.scrollBar = "ScrollBar";  
    onEnterFrame = null;  
}
```

显示一个复杂对象集中的标签：

```
// ScrollingList将自动使用属性名称 'label'  
// 如果在项目对象中找到：  
list.dataProvider = [{label: "one", data:1}, {label: "two", data:2}];  
  
// 但是如果项目对象具有不同的标签属性，如下所示，  
// 则列表可以配置成使用该属性来代替：  
  
list.labelField = "name";  
list.dataProvider = [{name: "one", data:1}, {name: "two", data:2}];  
// 如果从项目对象上构建一个标签逻辑上比较复杂并需要一个函数，  
// 则设置labelFunction属性  
list.labelFunction = function(itemObj:Object):String {  
    // 逻辑上构建一个标签  
}  
list.dataProvider = [{p1: "foo", p2: 1}, {p1: "bar", p2: 2}];
```

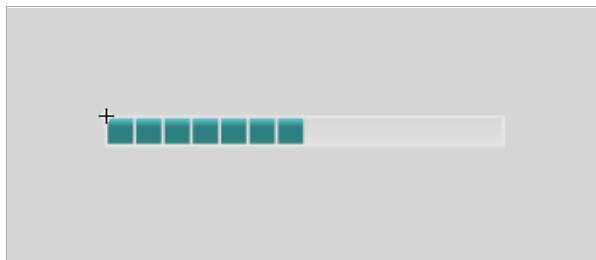
2.5 进度类型

进度类型用来显示状态或事件或动作的进度。The Scaleform CLIK框架包含了两个组件属于此分类，状态指示器StatusIndicator 和进度条ProgressBar。状态指示器StatusIndicator组件用来显示事件或动作的状态。而进度条组件ProgressBar与状态指示器拥有相同的语义，但是包含了一些附加功能，可以监听其他产生进度事件的组件或动作。



□ 45:来自*Mercenaries 2*的Fraction状态指示器实例

2.5.1 StatusIndicator



□ 46: 无皮肤状态指示器StatusIndicator.

状态指示器StatusIndicator component

(gfx.controls.StatusIndicator)显示事件或动作状态，使用时间轴作为视觉指示器。状态指示器的值将为一个最大和最小值之间的值用来产生一个帧的序号，以在组件时间轴上播放。由于组件时间轴用来显示状态，为创建创新的视觉指示器提供了绝对自由的发挥空间。

2.5.1.1 手交互

状态指示器StatusIndicator无用户交互。

2.5.1.2 组件设置

使用CLIK

状态指示器StatusIndicator的视频剪辑MovieClip不需要任何子单元。但是状态指示器需要至少两个帧以正确操作。确保在第一帧插入stop()命令避免播放该帧。状态指示器组件在值属性产生的对应帧执行gotoAndStop()命令。

2.5.1.3 标

状态指示器StatusIndicator组件无状态，组件帧用来显示事件或动作

2.5.1.4 检查属性

来自状态指示器StatusIndicator组件的视频剪辑MovieClip具有以下检查属性：

visible	如果设置为false则隐藏组件
disabled	如果设置为false则禁止组件
value	一个事件或动作的状态值，为一个最大值到最小值之间的播放的帧序号。
minimum	插入目标帧的最小值
maximum	插入目标帧的最大值

2.5.1.5 事

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

StatusIndicator组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false

2.5.2 ProgressBar

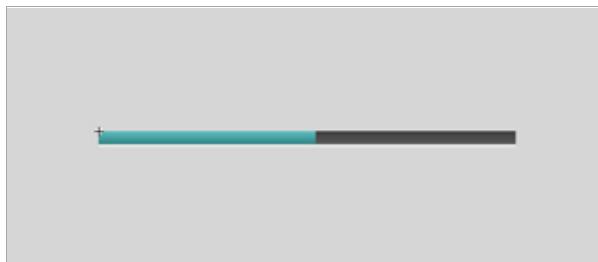


图 47: 无皮肤进度条 ProgressBar.

进度条ProgressBar

(gfx.controls.ProgressBar)与状态指示器StatusIndicator类似，也用来通过时间轴显示事件或动作状态，但是，与其他组件或产生的进度事件协作。正确指派目标并设置其模式，进度条组件将根据目标导入值（bytesLoaded 和 bytesTotal）自动改变视觉状态。

2.5.2.1 交互

进度条ProgressBar无任何用户交互。

2.5.2.2 组件设置

与状态指示器StatusIndicator类似，使用CLIK进度条StatusIndicator类的视频剪辑MovieClip不需要任何子单元。但是进度条ProgressBar需要至少两个帧以正确操作。确保在第一帧插入stop()命令避免播放该帧。进度条ProgressBar将在值属性产生的相关帧上执行gotoAndStop()。

2.5.2.3 状态

进度条ProgressBar组件无状态，组件帧用来显示事件或动作状态。

2.5.2.4 检查属性

来自进度条ProgressBar组件的视频剪辑MovieClip具有以下检查属性：

visible	如果设置为false则隐藏组件
disabled	如果设置为false则禁止组件
target	进度条ProgressBar将进行“监听”的目标对象，判断bytesLoaded 和 bytesTotal值
mode	properties.进度条ProgressBar监听模式。在“manual”模式，进度条值必须使用setProgress方法进行设置，在“

“polled”模式，目标必须为bytesLoaded和bytesTotal属性，而在“event”模式，目标必须分派“progress”事件，包括bytesLoaded和bytesTotal属性。

2.5.2.5 事

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

ProgressBar组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false
progress	当进度条ProgressBar值改变时产生
complete	当进度条值达到最大值时产生

下例中显示了进度条事件的监听：

```
myProgress.addEventListener("progress", this, "onProgress");
myProgress.addEventListener("complete", this, "onProgress");
function onProgress(event:Object) {
    if (event.type == "progress") {
        // 执行操作
    } else {
        trace("Loading complete!");
    }
}
```

2.6 其他类型

Scaleform

CLIK框架页包括若干个组件且不能简单区分，但是为UI开发提供了宝贵的功能。它们为Dialog、UILoader和ViewStack组件。对话框Dialog组件可以为模式和非模式对话框，UI导入器UILoader提供一个方便的接口以导入内容，视图堆栈ViewStack能用来管理窗体，

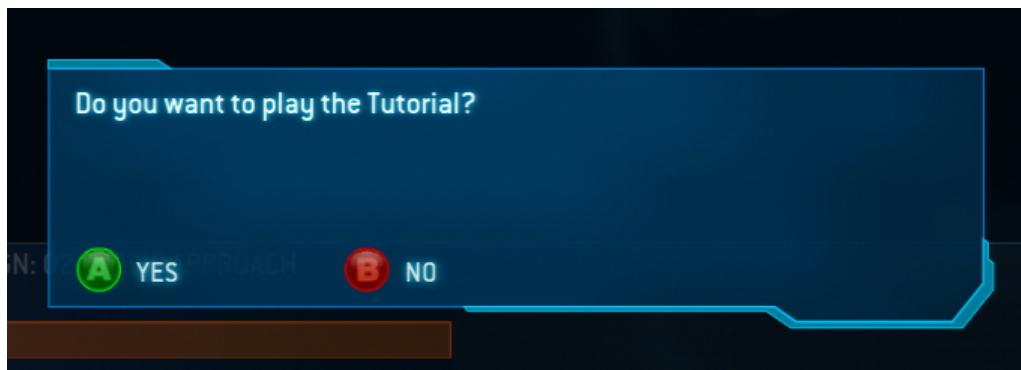
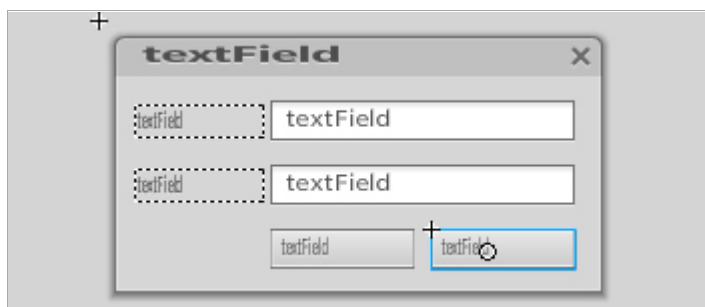


图 48:来自 *Halo Wars*的对话框 Dialog 实例

2.6.1 Dialog



□ 49:无皮肤对话框Dialog实例

CLIK对话框 Dialog组件

(gfx.controls.Dialog)显示一个对话框视图，如一个警告对话框，在应用程序的顶部。提供示例的静态接口，通过对话框显示和隐藏任何视频剪辑MovieClip，也为一个基本类可以作为实际对话框视频剪辑MovieClip使用（扩展）。为确保一次只打开一个对话框，新的Dialog.show()调用将关闭当前开打的对话框。

注意内置组件没有任何内容，因为设计成完全由用户定义。但是，通过简单的编辑组件符号可以添加内容。

2.6.1.1 用户交互

对话框Dialog的用户交互在创建的对话框视图里定义。

2.6.1.2 组件设置

使用CLIK Dialog类的MovieClip必须具备下面所列的子单元。也列出了可选元素：

- **closeBtn:** (可选) CLIK按钮 Button类型，与视窗关闭按钮类似。
- **cancelBtn:** (可选) CLIK按钮 Button类型，取消cancel按钮用来关闭对话框无确认。
- **submitBtn:** (可选) CLIK按钮 Button类型，OK按钮在确认后关闭对话框。
- **dragBar:** (可选)视频剪辑MovieClip 类型，对话框的拖动标题栏。

2.6.1.3 状态

对话框Dialog组件无状态。视频剪辑MovieClip作为对话框视图显示，或许有或许没有自身状态。

2.6.1.4 检查属性

来自对话框Dialog组件的视频剪辑MovieClip具有以下检查属性：

visible	如果设置为false则隐藏组件
disabled	如果设置为false则禁止组件

2.6.1.5 事件

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

Dialog组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false
submit	对话框Dialog确认按钮被点击。

- **data:** 对话框Dialog提交数据，对话框组件的getSubmitData方法调用于此方法，应该被自定义对话框覆盖。默认返回值为true（布尔型）。**getSub**

`mitData`的返回值为AS2对象。

close 点击了Dialog的闭关按钮

以下例子显示如何处理对话框提交事件：

```
myDialog.addEventListener("submit", this, "onLoginSubmit");
function onLoginSubmit(event:Object) {
    trace("Received data from dialog: " + event.data);
}
```

2.6.2 UILoader

CLIK UILoader

(`gfx.controlsUILoader`)通过唯一路径导入一个扩展的SWF/GFX或者图像。UILoaders支持导入资源自动缩放以适合边框。如果Scaleform和平台支持线程运行则异步导入资源。

2.6.2.1 用交互

UILoader组件无用户交互，如果一个SWF/GFX文件导入到UILoader，则可能有自身用户交互。

2.6.2.2 组件设置

使用CLIK UILoader类的MovieClip必须具备下面所列的子单元。也列出了可选元素：

- **bg:** (可选)视频剪辑 MovieClip

类型，UILoader背景，无函数支持，不同于场景中的上级组件的可是外观，背景可以在运行时取消。

2.6.2.3 状态

UILoader组件无状态，如果SWF/GFX文件导入到UILoader，可能有自身状态。

2.6.2.4 检查属性

来自UI导入器UILoader组件的视频剪辑MovieClip具有以下检查属性：

Visible	如果设置为false则隐藏组件
autoSize	如果设置为true，导入内容自动缩放以适合UILoader边框
maintainAspectRatio	如果为true，导入内容将在UILoader边距内纵横比进行合适缩放。如果为false，内容伸展铺满UILoader边框。
Source	导入的SWF/GFX或图像文件名
Timeout	导入等待超时时间为毫秒级，如果达到了超时时间内容还未导入完毕，UILoader将产生一个‘ioError’事件

2.6.2.5 事件

所有的事件调用都接收一个Object参数，包含事件相关信息。以下为通用事件的属性。

- **type:** 事件类型。
- **target:** 事件产生的目标。

UIloader组件产生的事件列表如下所示。事件旁列出的属性为通用属性的补充。

show	组件可视属性在运行时已设置为true
hide	组件可视属性在运行时已设置为false
progress	导入过程不考虑是否可以被导入，事件在以下情况下触发a) 内容导入或者b) 导入超时
loaded	导入数据比例，属性值在0和100之间
complete	内容导入完毕
ioError	指定内容不能被导入

下例展示了如何通知导入错误：

```
myUILoader.addEventListener("ioError", this, "onLoadingError");
function onLoadingError(event:Object) {
    displayErrorMessage("Could not load" + event.target.source);
}
```

2.6.3 ViewStack

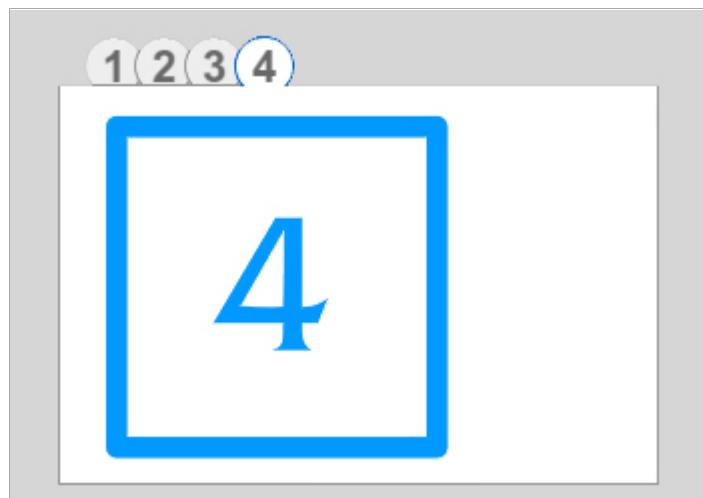


图 50: 无皮肤视图堆栈ViewStack

CLIK视图堆栈ViewStack

(gfx.controls.ViewStack)显示一个视图，来自一个导入到缓存中的集。该组件可以用于多视图组件如Tab Box或

Accordion（这些例子的样本都可以在演示文件夹里找到）。一个视图堆栈ViewStack能够指向另外一个组件如选择按钮RadioButton组以当组件改变时自动改变视图。

2.6.3.1 用交互

视图堆栈ViewStack组件无用户交互，通过ViewStack导入的视图并显示可能有自身用户交互。

2.6.3.2 组件设置

视频剪辑MovieClip使用CLIK视图堆栈ViewStack类不需要任何命名子单元。但必须根据导入内容缩放尺寸。

2.6.3.3 状态

视图堆栈ViewStack组件无状态，由ViewStack导入并显示的视图可能有自身状态。

2.6.3.4 检查属性

来自视图堆栈ViewStack组件的视频剪辑MovieClip具有以下检查属性：

visible	如果设置为false则隐藏组件
cache	如果为true，导入视图将存放在缓存中，这样节省创建视图的处理时间，但是需要一个固定的ViewStack目标组targetGroup（见下面内容）。
targetGroup	有效组对象名称，如ButtonGroup，产生‘change’事件，组对象中的当前元素必须具有一个数据属性包含一个导入并显示的视图的链接ID。例如，选择按钮RadioButton，拥有一个数据属性可以通过Flash IDE组件检查其指派一个链接ID。

2.6.3.5 事件

视图堆栈ViewStack不产生任何事件。

3 艺术细节

本章将帮助美工设计师开发Scaleform

CLIK组件的皮肤，包括为小的组件实例绘制皮肤的详细过程和最优方法，以及动画和内置字体。

3.1 最优方法

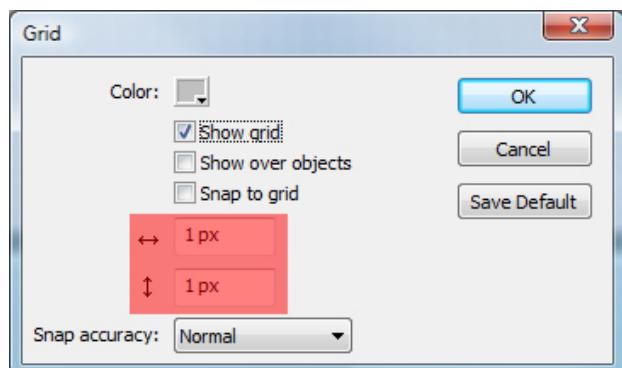
本章包括了为Scaleform Clik组件创建皮肤的最优方法。

3.1.1 像素图像

当开发基于CLIK组件绘制矢量皮肤，建议所有资源都用像素图像。一个完美的像素资源在Flash栅格化中完美排列。

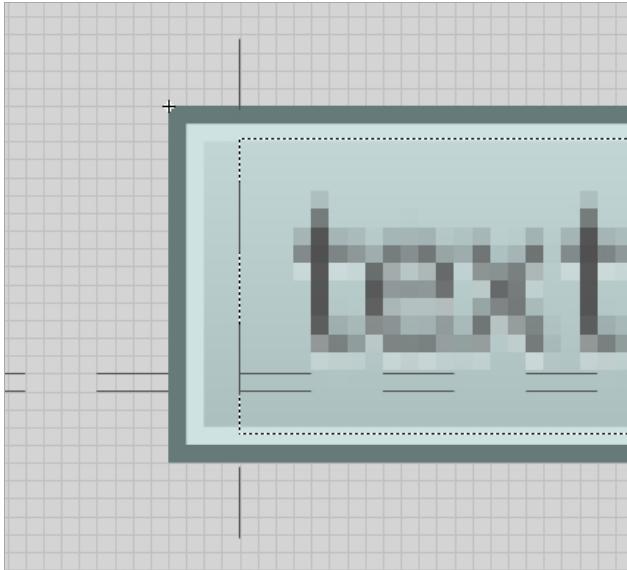
为使得能够改变栅格：

1. 从Flash顶部菜单选择视图；
2. 选择栅格，点击使栅格显示；
3. 从Flash顶部菜单再次选择视图；
4. 选择栅格然后点击编辑栅格；
5. 在垂直和水平位置输入1px；
6. 点击OK。



□ 51: 编辑栅格 Grid视窗

现在应该可以看到覆盖在场景上的一个栅格。每个栅格表示一个像素。确保当创建美术资源时对齐栅格。这回确保最终的SWF中图像不会模糊。**注意：**保持所有图像尺寸为2的指数倍；否则可能会导致模糊，见下面的2的指数倍标题。



□ 52: 一个像素的矢量图像

3.1.2 蒙版

在Flash中蒙板可以使设计师在运行时隐藏部分图像。蒙板经常在动画效果中使用，但是，蒙板消耗大量资源。Scaleform建议尽量避免蒙板的使用。如果蒙板必须使用，保持在个位数之内并测试添加蒙板前后动画性能。

在Flash中使用蒙板的一个可能选择为在Photoshop®中创建PNG，用透明混合处理需要遮盖的区域。但是，这只能用于非动画图像的透明区域。

3.1.3 动画

最好避免动画中矢量图形的转换，如将一个正方形转换成圆形。这些动画类型开销非常大，因为这些形状在每一个帧都需要重新进行评估。

如果可以避免在矢量图形中缩放动画，额外的栅格化影响内存和性能 –

将矢量图形转换为三角元素图像的过程。开销最小的动画为使用平移和旋转，不需要额外的栅格化。

避免使用可编程映射动画，选择以映射动画为基础的时间轴选择，因为时间轴映射动画在性能上更加有优势。使用时间轴映射动画，保持在既定帧速率下实现一个平滑的动画所需要的最少帧数量。

3.1.4 图层和绘制元素

在Flash中创建皮肤使用尽可能少的图层。每使用一个图层至少增加一个绘制元素。绘制元素使用越多，内存需求就越大，性能也将受到影响。

3.1.5 复杂皮肤界面

在复杂皮肤中最好使用位图；但是，在简单皮肤中使用矢量图可以节省更多内存并可以在任何分辨率下缩放而不失真（模糊）。

3.1.6 2的指数倍

确保位图在尺寸上为2的指数倍，2的指数倍位图尺寸如下所示：

- 16x16
- 32x32
- 64x64
- 128x128
- 128x256
- 256x256
- 512x256
- 512x512

3.2 已知问题和推荐工作流程

本节包括了已知艺术相关Flash问题列表和在Scaleform CLIK下的推荐工作流程。

3.2.1 复制组件

在Flash中复制组件存在几个问题，复制组件导致原来组件的链接信息和组件定义信息不拷贝到新的组件中去。同样的，有些组件没有这些链接信息将无法工作。本节描述了两种方法来解决这个问题。

3.2.1.1 复制组件（方法1）

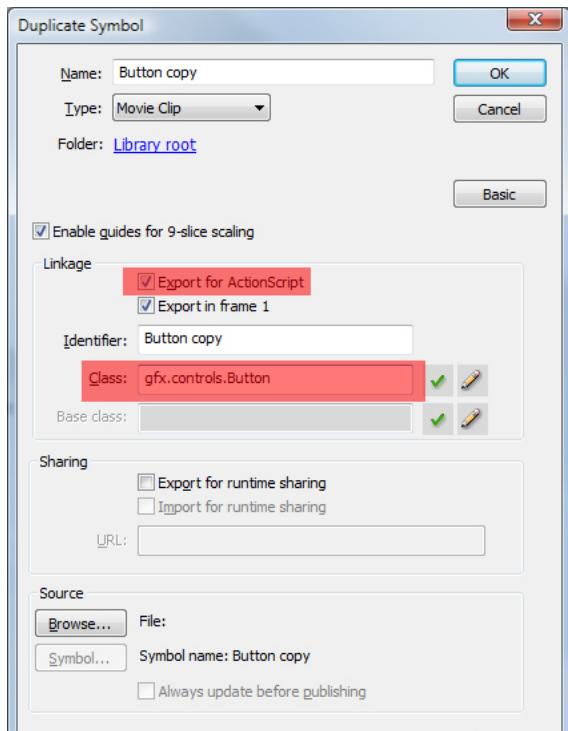
从外部FLA文件复制一个无更改（无皮肤）CLIK组件，如按钮，到目标FLA文件最快的方法如下：

1. 打开CLIK_Components.fla文件。
2. 从文件的库中拷贝组件（例如，按钮）到目标FLA文件，在源文件库中点击并选择Copy，然后点击目标FLA中的库并选择Paste。
3. 点击目标FLA库中的组件并选择Rename对其重新命名，不要与‘Button’重名。
4. 再次点击目标FLA库中的组件，选择Properties。
5. 改变Identifier区域匹配步骤3中选择组件的新名称。
6. 点击库视窗中的空白区域并选择Paste，一个新的原始组件拷贝将粘贴进来包括所有的完整的链接信息。

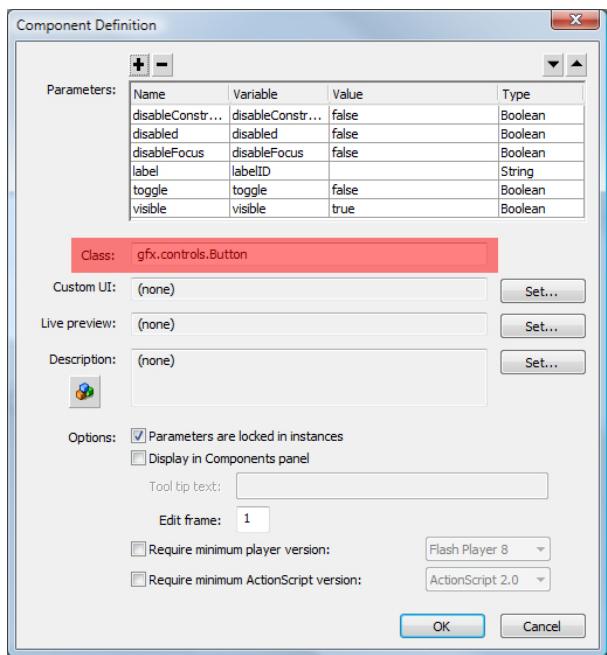
3.2.1.2 复制组件（方法2）

当在相同的库中复制符号（组件），链接信息将不会被拷贝到新的复制符号中去。组件功能执行需要这些信息，实现的方法为：

1. 点击library面板中需要复制的组件并选择Properties。
2. 在Properties视窗，双击文本（例如gfx.controlsButton）突出显示Class文本区域并按下(CTRL+C)键进行拷贝。
3. 按下Cancel。
4. 再次点击需要拷贝的组件，并选择Duplicate。
5. 在Duplicate Symbol视窗中，点击Export使ActionScript复选框使能。
6. 点击Class区域并按下(CTRL+V)来粘贴链接信息到该文本区域textField。
7. 按下OK。
8. 点击library面板中新拷贝的组件并选择Component Definition。
9. 点击Class textField空白区域并按下(CTRL+V)来粘贴链接信息到文本区域textField。
10. 按下OK。



□ 53: 复制符号链接（必须填充红色显亮区域）



□ 54: 组件定义（必须填入类）

3.3 皮肤绘制实例

Scaleform

CLIK的主要优势为视觉和函数层的分离。分离能够使大部分部件中编程人员和艺术设计师可以各自独立工作。轻松自定义外观和风格为这种分离的主要优势之一。

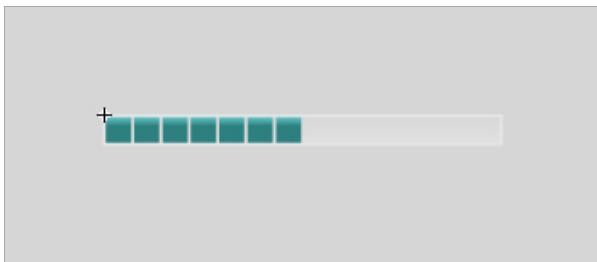
尽管内置CLIK组件具有一个标准的外观和风格，但有时候需要自定义以满足客户自身需求。以下部分即描述了自定义内置组件的方法。

CLIK组件设置皮肤与任何Flash标准符号相同，双击FLA文件库中的一个组件获得组件任意一个时间轴：

- 在Flash中修改默认皮肤；
- 在Flash中从头开始创建自定义皮肤；或者
- 导入Photoshop或Illustrator®中创建的艺术资源到Flash。

请浏览文档[CLIK入门](#)获得关于皮肤绘制指南的详细信息。

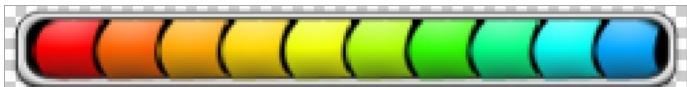
3.3.1 StatusIndicator皮肤绘制



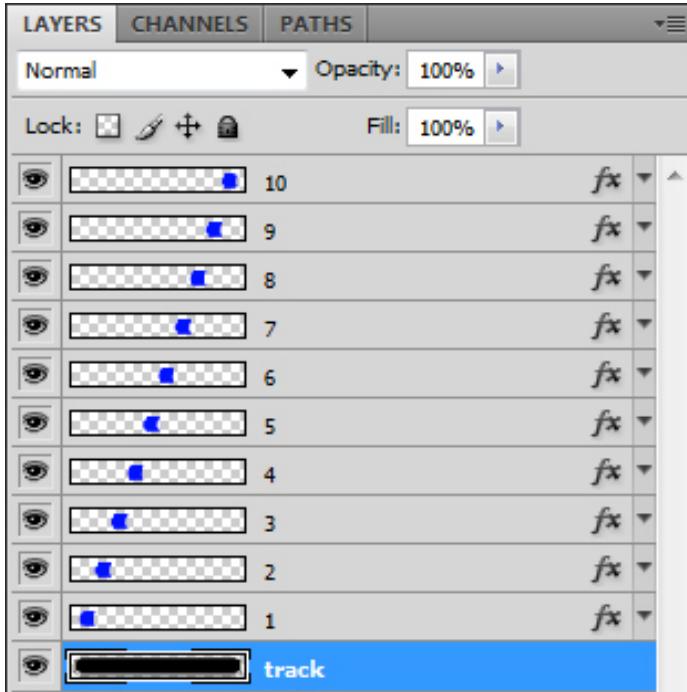
□ 55: 无皮肤StatusIndicator.

状态指示器StatusIndicator为一个唯一的组件在绘制皮肤时与其他大多数基于按钮的组件需要略微不同的方法，打开状态指示器组件，记录时间轴，具有两个图层：*indicator* 和*track*。*Track*用来显示背景图像；无其他功能。*Indicator*图层使用若干关键帧包括位图或矢量图来从最低到最高的逐步递增状态。

指南中使用Photoshop文件中创建的若干个图层的位图来绘制状态指示器。这为状态指示器皮肤绘制的方法之一，但是，还有其他的方法可以在场景中创建和装配图形。最终的结果应该都相同，能够使状态指示器正确工作。



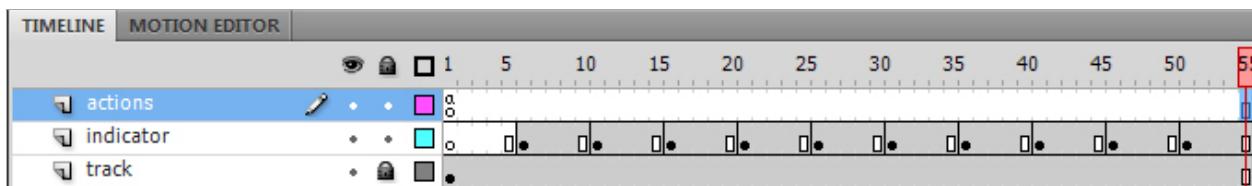
□56: 状态指示器StatusIndicator PSD 文件



□ 57: 在Photoshop中为StatusIndicator PSD 文件设置的图层

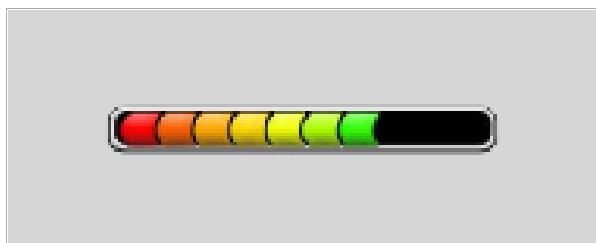
1. 在Photoshop中创建状态指示器StatusIndicator位图与上面描述类似。注意图层顺序和编号以及在图层中每个图像的位置。确保背景透明，为本使用手册创建一个类似文件；
2. 保存文件为PSD格式；
3. 在Flash中，从菜单选择File，然后选择Import -> Import to Stage；
4. 浏览并选择状态指示器StatusIndicator皮肤PSD文件；
5. 在Import窗口，确保Convert layers to:下拉菜单设置为‘Layers’；
6. 按下OK；
7. 一个新的Flash时间轴图层应该在PSD文件中的每个层中创建，在上面图像例子中，需要创建10个图层，因为PSD文件内部拥有10个图层（每个渲染为一个层）。每个图层标签分别从1到10，1为最底部图层10为最顶部图层，选择layer 1；
8. 在场景中的位图图像周围绘制一个复选框；
9. 将图像移动到老的无皮肤轨道位置，根据要求进行缩放；
10. 在layer 1上选择第一个关键帧并拖动到第6帧；
11. 点击帧选择Insert关键帧添加新的帧到layer 1图层的第11帧；
12. 在layer 2图层选择位图并按下(Ctrl+X)进行剪切；
13. 选择layer 1图层中第11帧位置为新关键帧并按下(Ctrl+Shift+V)放置位图到layer 1图层；
14. 重复此过程直到10个位图都在同一个图层(layer 1)，位于正确的关键帧。每个并排的位图应该拷贝到最后关键帧第5帧后的一个关键帧中。图层layer 2中的位图应该被拷贝到关键帧11；图层layer 3中的位图应该被拷贝到第16帧；图层layer 4中的位图应该被拷贝到第21帧，等等。使用10个PSD图像，最后关键帧应该位于第51帧；
15. 删除空图层（2-10）进行清理；

16. 如果在最后的位图关键帧之后时间轴上有附加的帧，加上另外五个关键帧，然后选择剩余帧并删除，删除操作为首先选择然后右键点击并选择*Remove Frames*即可。在本使用手册中，最后帧应该位于第55帧；
17. 选择原来的*indicator layer*并删除；
18. 选择原来的*track*图层并删除，确保不删除新导入的*track*图层；
19. 选择*layer 1*并重命名为‘*indicator*’；
20. 拖动*indicator*图层和*track*图层到*actions layer*图层下方，确保*track*图层在*indicator*图层下面；



□ 58: 最终时间轴（注意关键帧位置和最终帧位置）

21. 推出状态指示器StatusIndicator时间轴；
22. 设置检查参数值为1到10之间的任何值；
23. 保存文件；
24. 发布文件查看新绘制皮肤的指示器。



□59: StatusIndicator皮肤绘制

3.4 字体和本地化

本节详细描述了在Scaleform CLIK组件中字体使用。

3.4.1 概要

为了在Scaleform中字体能被正确渲染，所需字形（字符）必须内置到SWF或使用Scaleform本地化系统进行设置。以下为探究在Flash UI界面中管理字体的方法。

3.4.2 内置字体

和Flash播放器不同，Scaleform

需要内置字体以便显示。Scaleform播放器在未嵌入字体情况下将显示空的进行矩形，甚至这些字体在系统中已存在也是如此。这个效果的优势为方便判断在Scaleform

Scaleform中字体是否正确嵌入。在内置组件设置中，在每个文本区域textFidld实例中都嵌入了Slate Mobile。注意Slate

Mobile不包含任何中文、日文或韩文（CJK）字符或字形，内置组件只包含了ASCII字形。这可以通过将Slate Mobile替换为包含CJK字形并设置适当的嵌入选项进行改变。

注意直接内置字体到文本区域textFields与Scaleform本地化系统不兼容（在3.4.4小节有所描述）。Scaleform实际上推荐使用Scaleform本地化系统来设置字体，比直接嵌入具有很多优势。但是在某些情况下，如不需要本地化，内置字体最好为可选项。与UI界面管理类似，字体管理也需要几点考虑如本地化和内存管理。

3.4.3 在textField嵌入字体

只需在动态或输入文本区域textFields中嵌入字体，静态文本在编译时自动转换为字形轮廓（原始矢量图形）。在动态文本区域textFidld中嵌入字体，在场景中选择动态文本区域的属性检查(Window > Property Inspector)框中点击*Embed*按钮。选择应该嵌入的字符或字符集并选择*OK*。一旦完成了这些步骤，在你的FLA文件中就可以使用该字体。在相同的FLA文件的多行文本区域中如果需要使用相同的字体，这些步骤只需要执行一次。同样，多次嵌入相同字体不会增加内存使用量。注意字符集包含了大量的字形如中文在导入时占用大量的内存。

3.4.4 本地化系统

Scaleform本地化系统使用热交换机制无论在当前位置是否发生变化均可导入和导出字体库。这些字体库本身为SWF文件包括内置字体字形可以被SWF文本使用。Scaleform能够使用来自字体库的字形，为根据需求动态改变字体提供了强大的方法。

由于Scaleform本地化系统在文本区域textField直接嵌入字形时不能交换字体，文本区域必须使用一个导入的字形符号。通常字体符号在一个名为gxfontlib.fla的文件中创建并导入到swf文本中。这个导入的字体设置到所有支持字体交换的文本区域中。Scaleform当前可以截留该字体符号链接并基于当前位置导入不同的字体。

字体库不需要导出任何字体，只需要插入适当的字体（见前面小节的如何插入字体）。Scaleform本地化系统使用fontconfig.txt

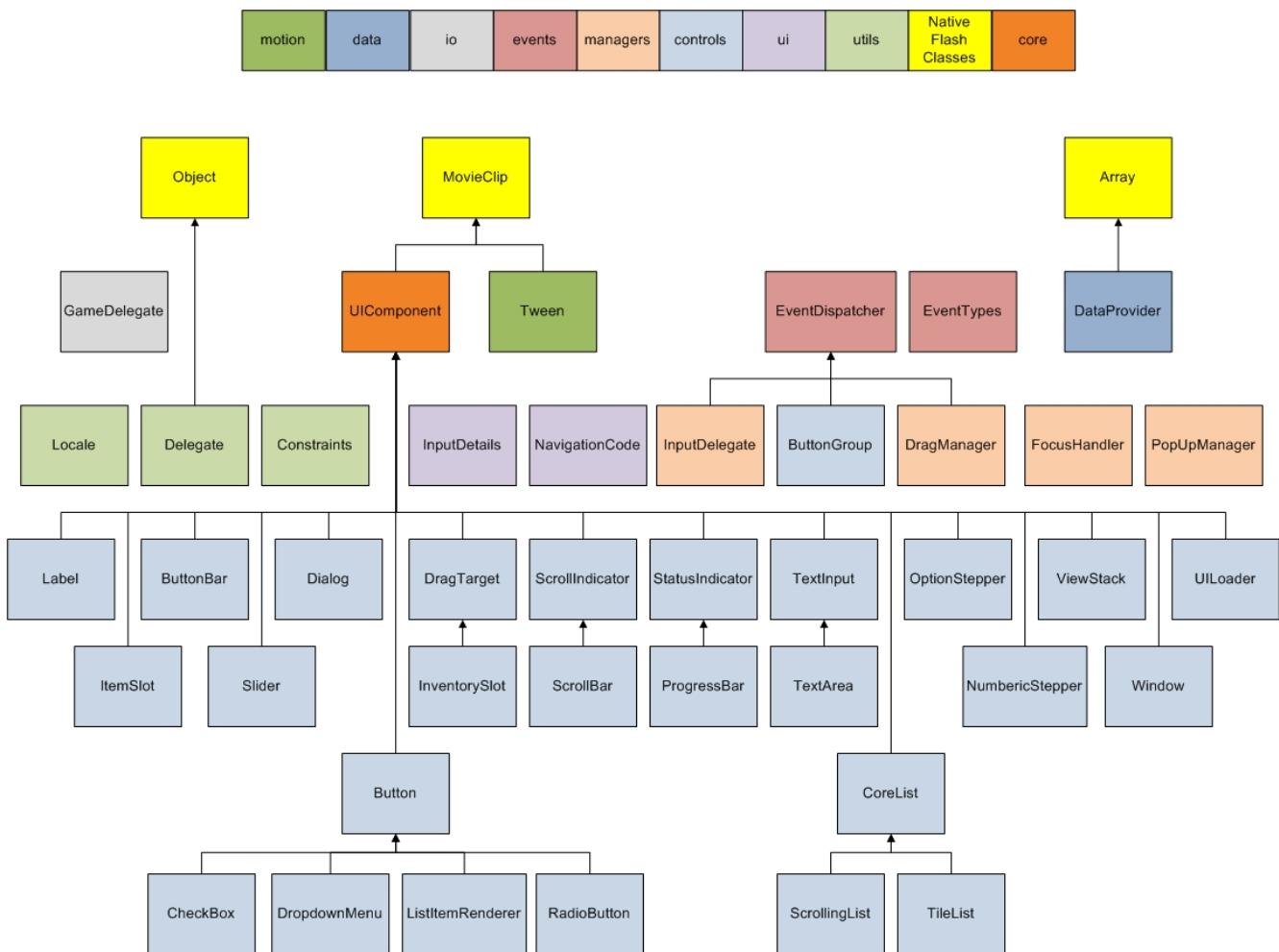
文件来定义每个位置的字体库，以及文本区域使用的字体符号之间的字体映射关系和插入到字体库中的实体字体。

fontconfig.txt文件也包含转换对应关系。Scaleform本地化系统可以自行文本的快速转换，这些文本显示在每个场景中的动态或输入文本区域。例如，如果一个文本区域包含文本‘\$TITLE’，转换映射具有一个对照表如\$TITLE=Scaleform，然后文本区域将自动显示‘Scaleform’来代替。

本章中描述的信息作为Scaleform字体和本地化系统的一个概括。为深入理解Scaleform中字体和本地化，请参考文档[字体概述](#)。

4 编程详述

本章描述了子系统框架和亮点的具体细节，提供了Scaleform CLIK组件架构的上层理解。



□ 60: Scaleform CLIK 类层次结构

4.1 UI组件UIComponent

内置组件章描述了与Scaleform

CLIK捆绑的组件使用的类。所有这些组件都继承了UIComponent类(gfx.core.UIComponent).的核心功能。这些类为所有CLIK组件的基础，Scaleform推荐自定义组件为UIComponent的子类。

UIComponent类本身从Flash 8

视频剪辑MovieClip类继承而来，因此继承了标准Flash视频剪辑MovieClip的所有属性和方法。一些自定义读/写属性也被UIComponent类所支持。

- **disabled;**
- **visible;**
- **focused;**
- **width;**
- **height;**
- **displayFocus:** 如果组件应该显示焦点状态则设置为true，更多信息请参考[焦点处理](#) 小节。

UIComponent拥有几个需要子类实现的空方法。这些方法包括：

- **configUI:** 组件配置调用；
- **draw:** 但组件无效时候调用，更多信息，请参考[失效](#) 小节；
- **changeFocus:** 当组件接收或失去焦点时被调用；
- **scrollWheel:** 当鼠标光在元件上方滚动时被调用。

UIComponent混合到EventDispatcher类用来支持事件预订和指派，因此，所有的子类都支持事件预订和指派。更多信息，请参考[事件模型](#) 小节。

4.1.1 初始化

改类在onLoad()事件句柄内执行下列初始化步骤：

- 设置MovieClip默认尺寸大小；
- 执行组件配置，改步骤调用configUI()方法；
- 绘制内容，该步骤调用validateNow()方法，立即执行画面刷新，例如调用draw()函数。

4.2 组件状态

几乎所有的Scaleform

CLIK组件支持视觉状态，状态通过组件时间轴上的一个特殊关键帧的导航来设置，或者传递状态信息到子单元。具有三个常用的状态设置，详细内容见下面内容。

4.2.1 按钮组件

任何行为类似按钮的组件，响应鼠标动作，可以被选到此分类。例如使用此类的CLIK组件为Button及其变量、ListItemRenderer、RadioButton和 CheckBox。复杂组件的子元素如滚动条ScrollBar也属于按钮Button组件。归到此类的CLIK组件可以直接使用按钮Button类(gfx.controls.Button)或者使用从按钮Button类继承而来的类。

按钮组件支持的基本状态为：

- **up** 或默认状态；
- 当鼠标箭头在组件上方或者获得焦点时为**over** 状态；
- 当按钮被点击时候为**down** 状态；
- **disabled** 状态；

按钮Button组件也支持前缀状态，可以根据其他属性的值进行设置。默认情况下，核心CLIK按钮Button组件只支持一个“**selected_**” 前缀，但组件处于选中状态时追加到帧的标签。

按钮组件支持的基本状态，包括选中状态，为以下所示：

- **up** 或默认状态；
- 当鼠标箭头在组件上方或者获得焦点时为**over** 状态；
- 当按钮被点击时候为**down** 状态；
- **disabled** 状态；
- **selected_up** 或者默认状态；
- 当鼠标箭头位于组件上方或获得焦点时为**selected_over** 状态；
- 当按钮被按下时为 **selected_down** 状态；
- **selected_disabled** 状态；

注释：本节中涉及的状态只是CLIK按钮组件支持的状态类型里的一部分。详细的状态列表请参考[CLIK按钮入门](#)文档。

CLIK按钮类提供了一个getStatePrefixes()方法，使开发者能够根据组件属性改变列表前缀。该方法定义如下：

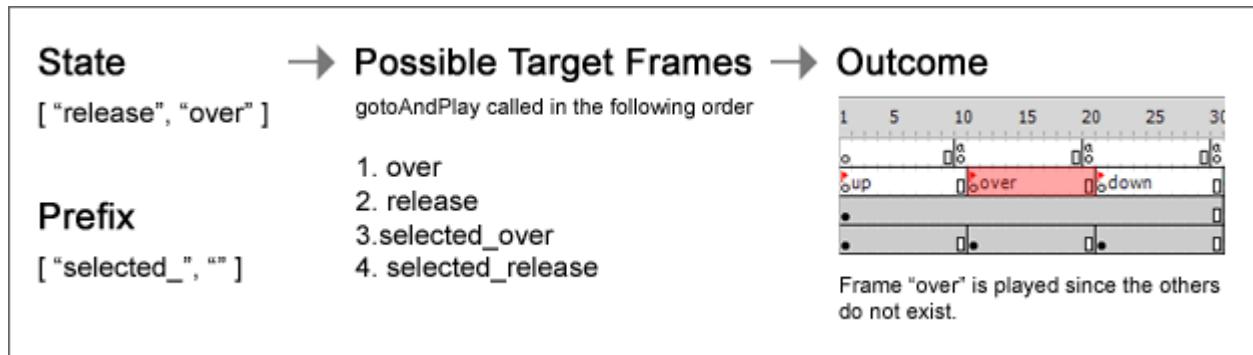
```
private function getStatePrefixes():Array {
    return (_selected) ? ["selected_","",] : [""];
}
```

如之前提到的，CLIK按钮默认情况下只支持“**selected_**” 前缀。getStatePrefixes()方法根据选择属性返回不同前缀序列。该前缀序列将于适当的状态标签协同使用来确定帧的播放。

当状态在内部进行设置，例如鼠标滚动，出现一个帧标签列表的查找表。按钮类中的stateMap属性定义了帧标签映射的状态。以下为CLIK按钮类中定义的状态映射关系：

```
private var stateMap:Object = {  
    up:[ "up" ],  
    over:[ "over" ],  
    down:[ "down" ],  
    release: [ "release", "over" ],  
    out:[ "out", "up" ],  
    disabled:[ "disabled" ],  
    selecting: [ "selecting", "over" ],  
    kb_selecting: [ "kb_selecting", "up" ],  
    kb_release: [ "kb_release", "out", "up" ],  
    kb_down: [ "kb_down", "down" ]  
}
```

每个状态可能有多个目标标签，从状态表返回的值与getStatePrefixes()返回的前缀结合产生一个播放的目标帧列表。下图描述了用来决定正确帧播放的完整过程：



□ 61:决定正确帧播放的过程

如果不能找到特定前缀的帧，播放位置总是跳转到最后的帧，组件为之前请求帧的默认状态。开发者可以忽略状态映射而创建自定义行为。

4.2.2 非按钮交互组件

这些涉及任何组件具有交互和接收焦点功能，但不响应鼠标事件。例如使用这类方法的CLIK组件为ScrollingList、OptionStepper、Slider

和TextArea。这些组件可能包含子单元可以响应鼠标事件。无按钮交互组件支持的状态为：

- **default** 状态；
- **focused** 状态；
- **disabled** 状态。

4.2.3 非交互组件

这些指向任何组件为非交互性质，但是可以被禁止。标签组件为状态所支持的默认设置组件中唯一的非交互组件。状态支持的非交互组件为：

- **default** 状态；
- **disabled** 状态。

4.2.4 特殊案例

有几个组件不遵守上面描述的状态规则，状态指示器StatusIndicator及其子类进度条ProgressBar使用时间轴显示组件值。播放位置将设置到帧的位置表示组件值的比例。例如，状态指示器中的一个50帧时间轴最小值为0，最大值为10，当前值设置为5（50%）将gotoAndStop()到第25帧（50帧的50%处）。扩展这些组件管理这些显示程序非常容易。`updateValue()`方法可以被修改或忽略以改变其行为。

在某些情况下，某些组件除默认行为外还具有特殊模式，支持额外组件状态。“文本输入”和“文本区域”组件在设置了“动作按钮”的情况下，能够支持“滚动”和“滑出”状态，以支持鼠标滚动和滑出事件。

4.3 事件模型

Scaleform CLIK组件框架使用一个通信范例称之为事件模型*event model*。组件在改变或交互时“分派”事件，容器组件可以访问不同的事件。使得多个对象可以被通知所发生的改变，而不是只有单个对象，这也是ActionScript 2调用的工作方法。

EventDispatcher类(gfx.events.EventDispatcher)提供了易于使用的API函数，支持事件模型。一个CLIK组件能够扩展该类或者使用EventDispatcher.initialize()方法将行为混合。但是，如果CLIK组件从UIComponent继承而来，则只需要在其构造器中调用super()，因为UIComponent组件已经与EventDispatcher相混合。一个EventDispatcher的子类或混合组件支持事件预订和指派。

4.3.1 最佳使用方法

4.3.1.1 预订一个事件

预订一个事件，可以使用addEventListener()方法，包括一个类型参数指定监听交互事件的类型。反过来使用removeEventListener()方法将取消事件预订。如果多个监听器包含相同的参数，只能触发一个。这些方法中的每一个都需要一个范围值参数，作为监听对象和回收信号，其为一个String类型指派事件时调用的函数名称。

EventTypes类(gfx.events.EventTypes)包含一个常用事件的枚举。可以用来替代字符串表示事件的类型(EventTypes.SHOW 替代 “show”)。

```
buttonInstance.addEventListener("itemClick", this, "callBack");
function callBack(eventObj:Object):Void {
    buttonInstance.removeEventListener("itemClick", this, "callBack");
}
```

在本例中按□□例□置□□听“itemClick”事件并在接收到事件□□行函数“callback”。回收函数移除事件□听器。

事件□象的属性基于原始□型但各不相同。CLIK组件□生的□□事件□象（及其属性）列表，□参考[内](#)[置□件相关章□。](#)

4.3.1.2 指派一个事件

CLIK组件希望使用dispatchEvent()方法通知预订监听器发生的变化或交互动作。该方法需要一个参数：一个包含相关数据的对象，包括一个强制类型属性指定指派事件类型。组件的框架自动添加一个目标属性，作为事件指派对象的索引，但是能够手动设置为用自定义目标进行替换。

```
dispatchEvent({type:"itemClick", item:selectedItem});
```

4.4 焦点处理

Scaleform

CLIK组件使用一个自定义焦点处理框架，在多数组件中执行，并且应该在无框架组件和符号中正常工作。所有的焦点变化发生在Scaleform播放器层，通过鼠标或键盘（游戏控制器）改变焦点，或者通过调用ActionScript中的Selection.setFocus（实例）来实现。FocusHandler管理器(gfx.managers.FocusHandler)在一个组件类创建后立即可以显现，无需直接显示焦点句柄FocusHandler。

4.4.1 最佳使用方法

当前焦点需要设置在CLIK组件实例上，否则焦点为播放器默认状态。如果未进行设置，焦点管理系统将不能正常工作（如Tab键不切换焦点等）。可以通过在任何组件上设置焦点属性为true使得焦点得以应用。另外一个应用焦点的方法，也是应用在非组件元素中，如下所示：

```
Selection.setFocus(firstComponentOrMovieClip);
```

视频剪辑MovieClips中无鼠标句柄（例如，`onRelease`、`onRollOver`）不能由Scaleform或Flash播放器获得焦点，也不能产生焦点变化事件。按钮组件可以自动添加鼠标句柄，其他组件将设置为标准视频剪辑MovieClip为`focusEnabled`和`tabEnabled`属性。

具有可获得焦点的子元素组件，如滚动条ScrollBar，使用焦点目标属性传递焦点到它们自身组件。当组件焦点发生变化时，焦点句柄FocusHandler将递归搜索焦点目标链，将焦点传给上一个焦点不返回一个焦点目标。

有时当一个组件不是播放器或引用程序的实际焦点时需要出现一个焦点。`displayFocus`属性可以设置为`true`使组件表现为获得焦点的样子。例如，当滑动条Slider获得焦点，滑动条轨道也需要获得焦点。注意当焦点属性发生改变时组件调用`UIComponent.changeFocus()`方法。

```
function changeFocus():Void {
    track.displayFocus = _focused;
}
```

反过来，有时候组件需要可以被点击，但不获得焦点，如面板拖动，活任何其他只受鼠标控制的组件。在这种情况下，设置`tabEnabled`属性为`false`。

```
background.tabEnabled = false;
```

4.4.2 在复合附件捕获焦点

复合组件为那些由其他组件组成，如`ScrollingList`、`OptionStepper`或`ButtonBar`。组件本身可能拥有子组件的鼠标句柄，但是不具有自身鼠标句柄。意思为在Flash和Scaleform中的Selection引擎不支持项和内置导航焦点将无法识别组件，而错误得去检查其子组件。

使组件行为不受合成的约束作为一个独立的入口，需要以下步骤：

1. 在所有具有鼠标句柄的子组件上设置`tabEnabled = false`（如`OptionStepper`中的箭头按钮）；
2. 在所有具有鼠标句柄的子组件上设置焦点目标属性。确保在容器组件中设置`focusEnabled = true`；
3. 如果有必要设置子组件中的`displayFocus`属性为`true`，该属性位于容器组件`changeFocus`方法之内；

现在当子组件获得焦点，焦点将转递给容器组件。

4.5 输入处理

由于Scaleform CLIK组件从Flash 8

MovieClip类继承而来，当接收到用户输入时与任何其他MovieClip实例具有相同的行为。如果安装了鼠标句柄组件可以捕获鼠标事件。然而，在CLIK上相关键盘或类似空孩子气事件处理具有概念上的区别。

4.5.1 最佳使用方法

所有的非鼠标输入都可以被InputDelegate管理类class

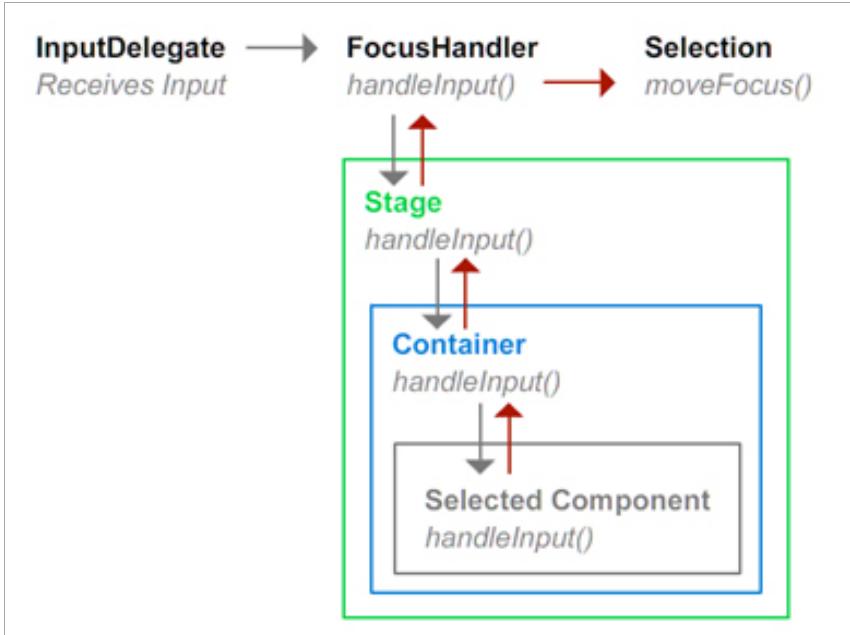
(gfx.managers.InputDelegate)截取，InputDelegate将输入命令转换为内部的任意一个InputDetails对象(gfx.ui.InputDetails)，或者从游戏引擎请求输入命令的值。后者包括了修改InputDelegate.readInput()方法以支持目标应用程序。一旦InputDetails被创建，一个输入事件就从InputDelegate得到指派。

InputDetails包含了以下属性：

- 类型，如“key”
- 编码，如按下键的键值
- 这是一个数值，它提供了诸如按钮矢量或按键类型等输入的附加信息。关口事件是由□的按下与□起□作用□生的，相□的□入□□的数□参数也分□□“keyUp”或“keyDown”；
- navEquivalent(navigation equivalent)，定义了可读的导航方向如“up”、“down”、“left”或者“right”只要可以进行映射。NavigationCode类(gfx.ui.NavigationCode)提供了一个手动通用导航枚举列表。

FocusHandler从InputDelegate监听输入事件并通过焦点路径传递给组件。获得焦点的组件用来决定焦点路径，为一个自上而下的组件列表，位于实现the handleInput()方法的现实列表层次当中。

本方法被焦点链的最顶部组件调用，InputDetails和pathToFocus队列作为参数传递。



□ 62: handleInput() 函数链

FocusHandler句柄预期从handleInput()函数调用返回一个布尔值，用来表示组件或焦点路径中的任何组件是否处理了输入。如果接收到响应为false，输入不为null具有一个navEquivalent，则输入信息传递给Stage或播放器。

```

function handleInput(details:InputDetails, pathToFocus:Array):Boolean {
    if (details.navEquivalent == "left")
    { // 和 NavigationCode.LEFT
        doSomething();
        return true;
    }
    return false;
}
  
```

每个组件处理输入将事件传递给pathToFocus序列作用的下一个组件，如果输入信息被处理则返回true或false。最好不要设想焦点路径中的下一个组件将正确执行输入处理，因此该方法应该进行确认，传递回一个布尔值不仅仅返回输入信息传递的值。

```

function handleInput(details:InputDetails, pathToFocus:Array):Boolean {
    var nextItem:MovieClip = pathToFocus.shift();
    var handled:Boolean = nextItem.handleInput(details, pathToFocus);
    if (handled) { return true; }
    //自定义处理代码
    return false; //如果被处理则为true
}
  
```

输入事件也可以传递给组件而不通过焦点路径。例如，在下拉菜单DropdownMenu中，`handleInput`传递到下拉列表组件，即使不在`pathToFocus`序列也是如此。这使得列表可以响应键盘命令。

```
function handleInput(details:InputDetails, pathToFocus:Array):Boolean {
    var handled:Boolean = dropdown.handleInput(details);
    if (handled) { return true; }
    // 自定义处理代码
    return false; // 如果处理则为true
}
```

也可以添加一个事件监听器到`InputDelegate.instance`中手动监听输入事件，并按照这种方法处理输入信息。注意这种情况下输入信息仍然可以被`FocusHandler`句柄捕获并传递到焦点层次。

```
InputDelegate.instance.addEventListener("input", this, "handleInput");
function handleInput(event:Object):Void {
    var details:InputDetails = event.details;
    if (details.value = Key.TAB) { doSomething(); }
}
```

`InputDelegate`应该在游戏中用来管理预期输入信息。默认`InputDelegate`处理键盘控制、方向键转换以及通用游戏导航键W、A、S和D直接转换到相等的导航键。

4.5.2 多鼠标光标

在一些系统中，如Nintendo

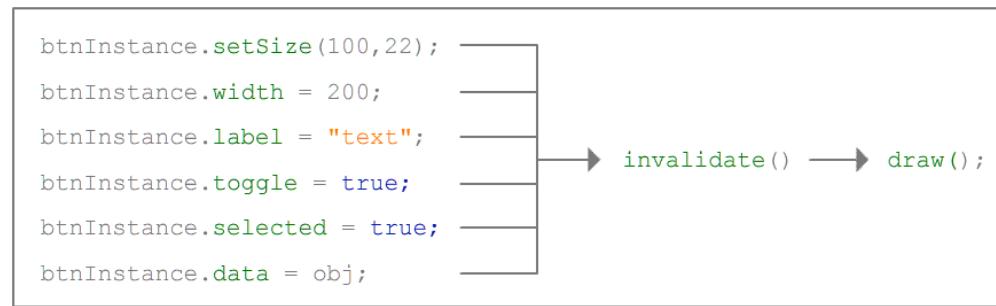
Wii™，支持多光标设备。CLIK组件框架支持多个光标，但是不允许在单个SWF文件中多个组件获得焦点。如果两个用户都点击独立的按钮，最后的点击项即为焦点项。

所有在框架中指派的鼠标事件包含了一个鼠标索引属性，作为产生事件的输入设备索引。

```
myButton.addEventListener("click", this, "onCursorClick");
function onCursorClick(event:Object):Void {
    switch (event.mouseIndex) {
        ...
    }
}
```

4.6 失效

失效为组件使用的一种机制，用来限制组件在多个属性发生变化时候组件的刷新次数。当一个组件为失效状态，在下一帧将重绘组件。这使开发者可以暂时忽略组件发生的改变，组件只要在特定时间更新一次即可。在有些情况下组件需要立即更新；但是大多数情况下失效性非常有用。



□ 63: CLIK 组件当特定属性改变时自动变为无效

4.6.1 最佳使用方法

在任何内部组件发生改变后（通常由设置功能引起），则调用UIComponent.invalidate()函数，该函数最终调用UIComponent.draw()函数重绘组件。invalidate()方法基于定时器延时产生draw()调用避免不必要的更新。开发者使用的组件可能不需要失效属相，但是至少也需要了解这项功能。

在需要立即重绘组件的情况下，开发者可以使用UIComponent.validateNow()方法函数。

4.7 组件缩放

Scaleform CLIK组件按照两种方式进行缩放：

1. 使用重复绘制，重新安排组件尺寸，组件元素进行缩放适合原来的比例尺寸。具有子党员并没有背景的组件采用此方法。
2. 元素不进行缩放维持纵横比例关系，组件进行缩放，但是元素不缩放。该方法是用在有图形背景的组件中，scale9grid进行了伸展，内部的文本缩放而不扭曲。

由于受到Flash

scale9Grid的限制，组件通常使用回流方法。这要求开发者建立“皮肤”标识，类似于Flash/Flex组件。如果部件拥有可以放缩的子元素，则回流方法的效果最好，因此它应用于container和类似实体。

反缩放方法专门为CLIK创建，主要是由于Scaleform中的scale9Grid扩展功能。它允许利用帧状态创建一个单一资产组件，而无需使用设置其他组件时所用的密集分层方法。基本的CLIK组件都具有简约的特性

，通常包括一个背景、一个标签和一个可选图标或子按钮，因此非常适合反缩放方法。然而，反缩放并不打算进行类container设置（面板设计等等）。在这种情况下，我们建议使用回流方法，因为它具有约束其余子元素的背景。

组件可以在Flash

IDE的场景中进行缩放，或者使用宽度和高度属性进行动态缩放，或者使用setSize()方法函数。缩放后的组件外观可能在Flash

IDE中不是那么精确。这就是其局限性，组件作为未编译的视频剪辑MovieClips不能进行实时预览LivePreviews。在Scaleform

Player中测试动画为确认缩放组件在游戏中外观是否精确的唯一方法。CLIK扩展了一个启动面板改进了这项工作流。

4.7.1 Scale9Grid

多数组件使用第二种缩放方法。Scaleform Player

中的Scale9Grid视频剪辑MovieClip资源在多数部分都附带了scale9grid，尽管Flash在包含了MovieClips的情况下将丢弃栅格。这意味着即使scale9grid在Flash

Player中不能工作，在Scaleform中可能可以很好发挥作用。

需要注意的一点是当视频剪辑MovieClip使用了scale9grid，其子单元也将根据此规则进行缩放。增加Scale9grid到子单元将导致忽略其上级栅格，正常进行绘制。

4.7.2 强制

Constraints类(gfx.utils.Constraints)辅助缩放组件内部的资源缩放和定位。使开发者在场景中定位资源，使资源保持到上级组件边缘的距离。例如，滚动条ScrollBar组件将根据在场景中的位置重新缩放轨道大小及其位置。Constraints在两种组件缩放方法中都被用到。

以下代码增加滚动条ScrollBar资源到configUI()方法的强制对象，向下方向键底部对齐，缩放轨道根据其上级组件进行延伸。draw()方法函数包含了代码以更新强制对象，从而任何元素都用此进行登记。这项更新在draw()函数中完成，因为在组件无效时被调用，通常在组件尺寸发生改变后。

```
private function configUI():Void {
    ...
    constraints = new Constraints(this);
    // 向上方向键upArrow已位于左上方。
    constraints.addElement(downArrow, Constraints.BOTTOM);
    constraints.addElement(track, Constraints.TOP | Constraints.BOTTOM);
    ...
}
```

```
private function draw():Void {
    ...
    constraints.update(__width, __height);
    ...
}
```

4.8 组件和数据设置

组件需要使用一个数据源dataProvider方法的列表数据。数据源dataProvider为一个数据存储和对象检索单元，开放了Scaleform CLIK

IDataProvider类(gfx.interfaces.IDataProvider)中定义的所有或部分API函数。

数据源dataProvider方法是用一个调用函数的请求模型，代替直接的属性访问。这允许数据源在需要时可以从游戏引擎获取数据。这具有存储优势，也可以将大块数据设置为小块，易于管理的数据。

组件使用数据源dataProvider包括任何扩展的CoreList (ScrollingList, TileList)、OptionStepper 和DropdownMenu。没有CLIK框架内的组件需要使用数据源IDataProvider接口，只在其API函数中包含了方法。数据源IDataProvider类只提供索引。

4.8.1 最佳使用方法

数据源DataProvider

类(gfx.data.DataProvider)包含在框架中使用其静态initialize()方法，将数据源dataProvider方法添加到任何ActionScript序列中去。组件使用一个数据源dataProvider将自动初始化序列使他们可以使用dataProvider方法进行访问。这意味着下列语法初始化一个静态声明序列作为完全可操作的数据源dataProvider，以及在IDataProvider中描述的方法。

```
myComponent.dataProvider = [ "data1",
    4.3,
    {label:"anObjectElement", value:6} ];
```

数据源dataProvider应该实现内容为：

- *length*: 可以作为一个属性或获取函数返回数据设置的长度；
- *requestItemAt*: 从数据源dataProvider请求一个指定项，通常被列表组件使用以在任何时间显示一个项，如OptionStepper；
- *requestItemRange*: 从数据源dataProvider请求一系列项包括一个开始和结束序号。通常由列表组件使用以显示更多项，如ScrollingList；
- *indexOf*: 返回项的序号；
- *invalidate*: 标记数据源dataProvider变化，提供一个新的数据长度。该方法也应该指派一个“**dataChange**”事件来通知组件数据已更新。数据源dataProvider应该支持一个可以公开访问和反应数据长度的长度属性。

实例需要数据简介列表能够使用一个数据作为数据源dataProvider。开发者应该明白在ActionScript 2中存储数据比在应用程序中自带数据需要更多的存储空间和性能开销。在大量数据设置推荐绑定数据源dataProvider和ExternalInterface.call 基于需求从游戏引擎获取数据。

4.9 动态动画

Scaleform CLIK提供一个自定义Tween类(gfx.motion.Tween)，与Flash 8

Tween类具有类似的功能，但是与Scaleform完全兼容。该两种Tween类均支持easing函数，如在mx.transitions.easing.* 软件包下的函数。

但是，CLIK Tween类与其副本有明显区别。首先，它安装Tween方法到Flash 8

视频剪辑MovieClip类的原型。这向所有的MovieClips开放了Tween功能，不仅仅是CLIK组件。其次，CLIK

Tween类使用onEnterFrame因此每个MovieClip每次只能发出一个Tween。如果一个已经发出一个Tween的MovieClip需要创建一个新的Tween，则新的Tween将停止之前的一个。然而Tween方法支持单个MovieClip的多个属性，允许同一个对象的不同属性在相同时间发生变化。下节中的例子展示了如何创建Tweens在同一时间影响多个属性。

4.9.1 最佳使用方法

Tween类在MovieClip开放了两个主要的方法：tweenTo() 和

tweenFrom()。tweenTo()方法将MovieClip的当前属性值映射到在函数参数中指定的一个属性。tweenFrom()方法从参数列表中指定的属性值映射到当前值。

在使用Tween方法之前，必须随MovieClip原型进行安装。Tween类提供了一个帮助静态函数执行函数：Tween.init()。

```
import mx.transitions.easing.*;
import gfx.motion.Tween;
Tween.init();           // 安装tween方法到MovieClip原型
// 执行一个1秒钟的Tween动画从当前垂直位置和alpha值到指定的值
myMovieClip.tweenTo(1, {_x: 200, _alpha: 0}, Strong.easeIn);
```

当Tween动画结束时被通知，只需在Tween对象中创建一个onTweenComplete()函数。

```
import mx.transitions.easing.*;
import gfx.motion.Tween;
Tween.init();
```

```
mc.onTweenComplete = function() {
    trace("Tween has finished!");
}
mc.tweenTo(5, {_rotation: 20}, Bounce.easeOut);
```

4.10 弹出式支持

Scaleform

CLIK包含了PopUpManager类(gfx.managers.PopUpManager)，支持弹出视窗如对话框和提示工具。对话框组件使用PopUpManager来显示内容。

4.10.1 最佳使用方法

PopUpManager拥有几个静态方法辅助弹出视窗的创建和维护。`createPopUp()`方法能够被用到任何视频剪辑MovieClip符号（包括CLIK组件）的链接ID创建弹出视窗。文本参数被用来创建一个弹出视窗的实例，同时相关参数用来定位弹出菜单。这些参数在视频剪辑MovieClips中都需要用到。

```
import gfx.managers.PopUpManager;
PopUpManager.createPopUp(context, "MyToolTipLinkageID",
    {_x: 200, _y: 200}, relativeTo);
```

Scaleform扩展`topmostLevel`参数用来保证弹出视窗总是显示在场景中所有其他部件的上面，由于与库相关的问题，使用此方案替代在`root`层创建弹出窗口。如果导入一个子SWF文件到上级组件，试图创建一个在上级内容所在的路径的库定义的符号实例，则将产生符号查找错误，因为上级组件无法访问子元素的库。不幸的是，没有专门工作区域来解决这个问题，因此`topmostLevel`方案为最佳选择。

注意`topmostLevel`也能应用到非弹出视窗，在场景中保持此类元素的Z轴序列。因此，在弹出菜单顶部绘制鼠标光标，光标可以创建在最高处位置或层次并设置`topmostLevel`属性为`true`。

4.11 拖放

DragManager 类

(gfx.managers.DragManager)支持初始化和拖放操作。该类是一单个组件，提供了一个实例属性访问唯一的对象。使用此对象，开发者可以开始和结束拖放操作。

`startDrag()`方法能够被调用使用场景中的MovieClip或者一个符号ID开始拖放操作，该操作将创建一个用来拖动的符号实例。`stopDrag()`方法结束拖动操作并通知所有监听器。DragManager与InputDelegate一同注册，允许使用键盘或类似控制器退出拖放操作。

4.11.1 最佳使用方法



□ 64:动作中的DragDemo

DragManager只执行拖动管理。依赖于开发者创建组件利用DragManager来提供拖放功能。Scaleform CLIK包括一个例子为拖放目标类名为DragTarget (gfx.controls.DragTarget)。DragTarget扩展UIComponent因此分类为一个CLIK组件。拥有几个唯一特性作为DragManager的补充。

DragTarget具有一个拖放类型的概念。这些拖动类型能够按照每个DragTarget进行配置使组件允许或者禁止拖动操作。为实现这些拖动类型，DragTarget也需要随DragManager为dragBegin 和dragEnd安装事件监听器。这使得DragTarget可以显示其是否支持拖放操作。

CLIK捆绑的演示文件中使用两个组件，作为子类或者组成DragTarget来展示使用DragManager 和 DragTarget进行的拖放操作。这两个组件为InventorySlot 和 ItemSlot，这些组件类作为CLIK框架的一部分提供，但是，注意他们只是作为一个例子表示可以实现的功能，并不是一个所有应用场合下的完整解决方案。

InventorySlot 类

(gfx.controls.InventorySlot)支持显示一个图标集，用来表示各个条目，通常在角色游戏中看到。将DragTarget类分为子类提供放置支持，包含了代码可以使用DragManager产生拖动操作。当拖动操作开始时，

伴随光标创建一个图标的拷贝。在一个有效的放置操作，目标InventorySlot将改变其拖动操作时的图标。

ItemSlot 类

(gfx.controls.ItemSlot)与InventorySlot非常类似，但是包含了额外的功能以支持鼠标事件，如鼠标点击。替代子类，ItemSlot包含了DragTarget的一个实例。也包含了CLIK按钮的一个实例。这两个子类为ItemSlot提供了必要的功能以支持拖动和放置操作以及按钮操作。

4.12 杂项

4.12.1 代理Delegate

代理Delegate 类

(gfx.utils.Delegate)提供一个静态方法在对应范围内以创建一个函数代理。不被Scaleform CLIK组件使用，主要被DragManager使用，使用实例如下：

```
eventProvider.onMouseMove = Delegate.create(this, doDrag);
```

4.12.2 本地Locale

Locale类 (gfx.utils.Locale)

为自定义本地化方案提供了一个接口。Scaleform本地化方案执行内部的转换查询不需要来自ActionScript的外部查询。但是，自定义转换可能需要此查询。Button、Label 和 TextInput 组件以及它们的子类都需通过Locale.getTranslatedString() 方法查询一个本地化字符串。默认的实现方法为简单返回相同的值。需要由开发者修改Locale类以支持自定义本地化方案。

5 实例

下面小节包括一些使用Scaleform CLIK组件的例子。

5.1 基础

一下实例比较简单，但展示了易于使用的Scaleform CLIK框架执行常规任务。

5.1.1 包含两个textField的ListItem Renderer



□ 65: 滚动列表ScrollingList 展示包含两个标签的列表项

滚动列表组件默认使用ListItemRenderer来显示行内容。然而ListItemRenderer只支持单个文本区域textField。很多情况下列表项需要多个文本区域textField用来显示，或者甚至不需要文本区域如图标。本例展示了如何添加两个文本区域到列表项。

首先，定义需求，对象为一个自定义ListItemRenderer支持两个文本区域。自定义ListItemRenderer应该与滚动列表ScrollingList相兼容。由于目的为使得每个列表项具有两个文本区域textFields，数据也应该不止单个字符串列表。本例中我们使用以下数据源dataProvider:

```
list.dataProvider = [{fname: "Michael", lname: "Jordan"},  
                    {fname: "Roger", lname: "Federer"},  
                    {fname: "Michael", lname: "Schumacher"},  
                    {fname: "Tiger", lname: "Woods"},  
                    {fname: "Babe", lname: "Ruth"},  
                    {fname: "Wayne", lname: "Gretzky"},  
                    {fname: "Usain", lname: "Bolt"}];
```

数据源包含的对象具有两个属性: fname 和 lname。该两个属性将显示在两个列表项的文本区域textField中。

由于默认的ListItemRenderer只支持一个文本区域textField，需要能够支持两个文本区域textField。最简单的实现方法为将ListItemRenderer类作为子类。一下为调用MyItemRenderer类的源代码，其中使用了ListItemRenderer子集并添加了两个文本区域的基本功能支持。（代码位于*MyItemRenderer.as*文件中，文件位于FLA工作目录）：

```
import gfx.controls.ListItemRenderer;

class MyItemRenderer extends ListItemRenderer {

    public var textField1:TextField;
    public var textField2:TextField;

    public function MyItemRenderer() { super(); }

    public function setData(data:Object):Void {
        this.data = data;
        textField1.text = data ? data.fname : "";
        textField2.text = data ? data.lname : "";
    }

    private function updateAfterStateChange():Void {
        textField1.text = data ? data.fname : "";
        textField2.text = data ? data.lname : "";
    }
}
```

ListItemRenderer 的setData

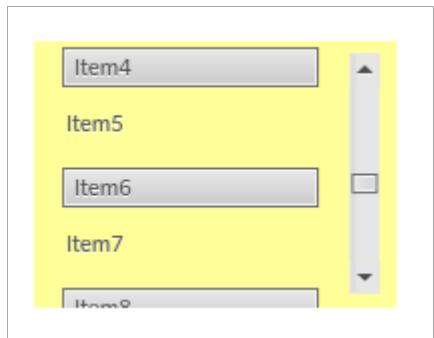
和updateAfterStateChange方法在本子类中没有涉及。setData方法在列表项收到来自容器列表组件(ScrollingList、TileList等)项数据时被调用。在ListItemRenderer中，本方法设置一个textField的值，而MyItemRenderer用来设置textField的值，同时也存储一个索引到内部项目对象。此项目对象在updateAfterStateChange方法中被重用，此方法在ListItemRenderer状态发生改变时被调用，此状态的变化需要文本区域textField刷新值。

到此，已经定义了具有复杂列表项支持列表项渲染的ListItemRenderer类的数据源dataProvider。要连接所有相关列表组件，必须创建一个符号以支持该型的ListItemRenderer类。本例中，最快的实现方法为修改ListItemRenderer符号使其具有两个文本区域textField调用‘textField1’和

‘textField2’。下一步该符号标识和类必须修改为MyItemRenderer。为在列表中使用MyItemRenderer组件，修改列表实例的itemRenderer检查属性，从‘ListItemRenderer’改变到‘MyItemRenderer’。

现在运行FLA，列表应该显示出来包含列表元素显示两个标签：在数据源dataProvider中设置的fname 和 lname属性。

5.1.2 像素滚动视图



□ 66: 按像素滚动包含CLIK元素的视图

按像素滚动通常用户复杂用户界面。本例展现了如何使用CLIK滚动条ScrollBar组件简单得实现此功能。

首先，创建一个新的符号用于滚动视图，这提供了一个容器，简化了所需的偏移量计算。在滚动视图容器中，从顶部到底部顺序设置一下图层。这些图层并不需要，但是用来作为说明：

- **actions:** 包含ActionScript代码使得滚动视图工作；
- **scrollbar:** 包含一个CLIK滚动条ScrollBar实例，该实例称为 ‘sb’ ；
- **mask:** 由矩形或MovieClip定义的蒙板图层，设置图层属性为一个蒙板；
- **content:** 包含需要滚动的内容；
- **background:** 可选层包含了一个背景突出无蒙板区域。

添加相关元素到图层并创建一个符号将其保存。通过创建一个符号保存所有的内容，滚动内容的任务变得十分简单。调用这些内容实例 ‘content’ 并设置y轴为0。这确保滚动逻辑上不需要计算不同的偏移量。然而，这些偏移根据滚动视图的复杂性也许需要用到。

到此，定义了所需创建的一个简单滚动视图，以及需要互相关联的元素名称结构。将以下ActionScript代码放到code图层的第一帧：

```
// 133 为视图尺寸（蒙板高度）
sb.setScrollProperties(1, 0, content._height - 133);
sb.position = 0;

sb.addEventListener("scroll", this, "onScroll");
function onScroll() {
    content._y = -sb.position;
}
```

由于滚动条不与其他组件连接，需要手动配置。`setScrollProperties`方法就是用来使其在一个位置滚动，并设置完全显示内容的最大值和最小值。本例中的滚动条位置为像素格式。运行文件，滚动视图可以通过与滚动条交互进行改变。

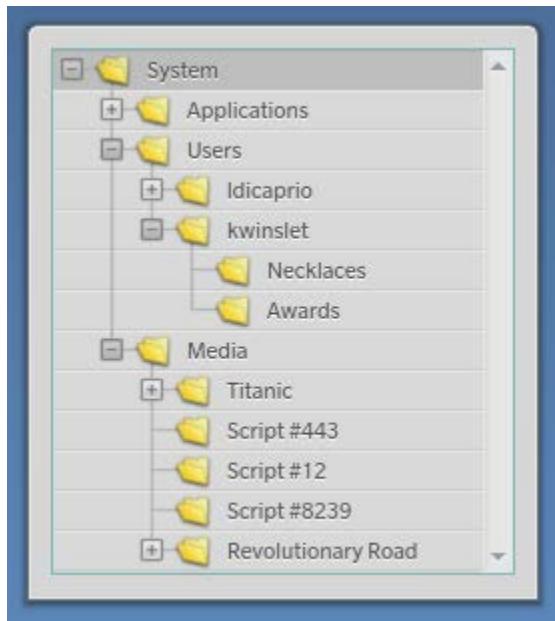
注意在Scaleform中的扩展蒙板使用将明显降低性能，特别是HUD的应用。Scaleform建议开发者需要评估其性能指标。

5.2 合成

本节以下内容介绍了几个Scaleform

CLIK捆绑的合成演示。只提供了上层实现细节。注意学习这些内容需要了解Flash和ActionScript 2。

5.2.1 使用ScrollingList 的TreeView



□ 67: TreeView 演示

树状视图演示TreeView Demo

(*Resources/AS2/CLIK/demos/TreeViewDemo.fla*)使用ScrollingList作为基本组件实现一个树状视图控制界面，并包含了自定义ListItemRenderer 和 DataProvider。

让我们分析普通滚动视图ScrollingList和树状视图的区别：

- 树状视图中的元素数量能够根据项目层次状态进行改变，但是仍然为元素的一个线性序列；
- 树状视图项鄙视使用一个可视导航器显示树的层次深度；在多数情况下将提供一个“tabbed”查找功能。
只要改变ListItemRenderer为‘tab’其内容。

- 树状视图必须包含一个机制允许用户扩展和收缩每个项。同样，可以修改**ListItemRenderer**以包含一个按钮元素提供此项功能。

滚动列表为构建一个树状视图的有效工具。在演示文件中，定义了一个自定义的**ListItemRenderer**名为**TreeViewItemRenderer**以及一个自定义的数据源名为**TreeViewDataProvider**。演示也使用了一个较小的工具类名为**TreeViewConstants**提供了常规列举功能。所有这些类都可以在*Resources/AS2/CLIK/demos/com*目录下找到。

第一项为数据描述决定何时实现一个树状视图。**ActionScript**

2对象本质上为一个哈希表，该演示使用了此项属性。树由一个对象树构成，如下所示：

```
var root:Object =
{
    label: "System",
    nodes:[
        {label: "Applications",
         nodes:[
             {label: "CGStudio",
              nodes:[
                  {label: "Data"},
                  {label: "Samples"},
                  {label: "Config"}
              ]}
            },
            {label: "Money Manager"}, 
            {label: "Role Call v6"}, 
            {label: "Super Draft"}, 
            nodes:[
                {label: "Templates"}]
        ...
    ]}
```

树中的每个项描述为带有两个属性的一个对象：**label**

和**nodes**，通过数组存放子节点。用滚动列表**ScrollingList**展现层次，需要一个自定义数据源用来解析和维护树状结构并开放相关数据到列表。

TreeViewDataProvider将树状对象收接受到其构造器并执行一个预处理步骤，用特殊属性注释每个项，只对数据源和自定义

ListItemRenderer有用。这些属性包括项的类型、扩展/收缩状态、上级/同级链接和一个线性图形描述器数组，描述了渲染时用到的线性图像。**TreeViewDataProvider**实现了**CLIK**数据源需要的相关公共方法，但是服从多数工作的检索帮助功能。例如，项目范围检索首先使用列表顶索引查找树中对象，然后搜集一个项的线性序列，通过树的遍历开始于顶部项。

`TreeViewItemRenderer`通过滚动列表`ScrollingList`从`TreeViewDataProvider`接收数据。获得的数据对象包括了所有渲染项需要的元数据，不需要知道树的结构。未经更改的`ListItemRenderer`组件符号在此演示中被重新使用，其类链接设置为`TreeViewItemRenderer`。列表项的文件夹图标和线性链接图形可以被动态创建。这些图形元素预先保存在缓存中以避免相同‘tab’中复制相同的符号。`ListItemRenderer`中的文本区域`textField`元素向右移动，以项目的深度为基础。

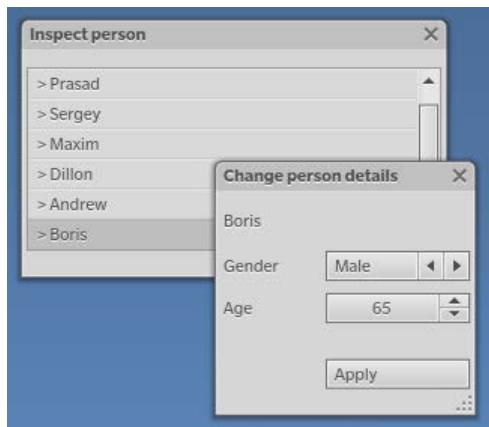
`TreeViewConstant`类包含的列举项如下所示：

- 层次中的节点类型：`open node`、`closed node`、`leaf node`；
- 文件夹图标类型；
- 线性连接器类型。

后面的列举项只用来显示，该演示使用文件夹图标和线性图形提供了项目之间的无缝连接，不管其深度为多少。为实现这种效果，文件夹图标和线性图形必须覆盖所有不同展开和搜索视图设置。

由于该树状视图基于运动列表`ScrollingList`，一个滚动条`ScrollBar`组件能够被连接，滚动条根据树的层次状态改变其外观，树的展开/收缩/状态影响滚动条的控制范围。与滚动条交互可以根据预期改变滚动列表`ScrollingList`视图。

5.2.2 可复用视窗



□ 68:视窗演示

视窗演示Window Demo

(`Resources/AS2/CLIK/demos/WindowDemo.fla`)提供了一个简单的可复用窗口组件，可以显示任意内容。窗口组件类能够在`Resources/AS2/CLIK/demos/com/scaleform/`目录下找到。

该窗口组件以核心`UIComponent`类为子类因此规类为CLIK组件。不属于核心框架类之下因通常窗口组件定义比较困难，单独使用实现起来更加高效。组件添加更多的特性，将增加内存使用降低性能。

因此，本类只实现了窗口组件所需要的常用特性核心集，它们为：

- 可修改的标题栏；
- 可拖动的标题栏和窗口背景；
- 关闭按钮；
- 伸缩边界（只能向下和向右伸缩）；
- 最小和最大维数。

Window类开放了几个检查属性：

Title	窗口标题
formPadding	窗口边界和内容的填充数
formType	导入内容类型，该值可以为‘symbol’或‘swf’，如果为‘symbol’内容从库导入，如果为‘swf’则导入一个扩展SWF文件。
formSource	内容的来源，可以为一个符号名称（如果 formType 为‘symbol’）或者SWF文件名（如果 formType 为‘swf’）。
allowResize	显示或隐藏调整大小的按钮。
minWidth, maxWidth, minHeight, maxHeight	窗口的调整尺寸，如果最大值设置为一个负值，则按照最小值来计算。将最大值都设为负值与隐藏调整按钮效果相同，完全禁止了恢复。如果最大值设置为0，组件可以调整大小没有最大值限制。最小值总是为内容的初始化大小。

添加一个检查属性到自定义组件类非常简单。如果一个公共属性在获取或设置属性时不需要执行任何代码，可以如下声明：

```
[Inspectable(name="formType", enumeration="symbol,swf")]
private var _formType:String = "symbol";
```

关于检查元数据语法的更多信息，请参考Flash文档。注意检查元数据支持实际属性的混合，即为_formType属性可以作为formType的别名以增强可读性。

如果属性在设置或获取值后需要执行代码，则属性应该定义获取和设置函数：

```
[Inspectable(defaultValue="Title")]
public function get title():String { return _title; }
public function set title(value:String):Void {
    _title = value;
}
```

```
    invalidate();
}
```

本窗口组件本身支持多个子单元，如标题按钮、关闭按钮和调整按钮。这些按钮每一个的都是一个**CLIK**按钮组件。由于这些按钮在类中需要用到，组件符号必须包含相应的子单元以反映这个需求。

由于窗口类以**UIComponent**类为其子类，包括了两个主要的方法：**configUI()**和**draw()**。**configUI()**方法设置按钮监听器和调整大小的强制对象。**draw()**方法通常以手动或者强制对象形式执行组件排布绘制。然而，也包括使用一个符号名或者文件路径导入内容。在导入一个私有方法后，调用**configForm()**设置内容。

使用Flash 8

MovieClipLoader执行文件导入。由于当线程支持有效时文件导入在**Scaleform**中异步执行，组件从**MovieClipLoader.onLoadComplete()**内部调用**configForm()**函数，这在文件导入完毕时候被触发。导入的形式内容，无论是通过符号还是文件路径导入，都添加到内部强制对象。在窗口组件进行调整后，将为无效状态。这将导致**draw()**方法的调用，从而在强制对象上执行一个更新操作使用新的尺寸。由于形式内容注册到强制对象，通过**validateNow()**方法通知变化。

如果形式内容继承自**UIComponent**类，则将调用自身的**draw()**方法。但调用**draw()**方法后可以进行形式内容的分布管理，然后可以定义个**validateNow()**方法以重新绘制自身元素。窗口演示包括了几个符号和**SWF**文件的例子，其中使用了**draw()**方法和**validateNow()**方法来维持形式分布。

窗口组件也包含特殊的逻辑移动在前景中被鼠标光标按下的窗口。确保所有窗口组件实例存在于相同的层次，使组件获得焦点到下一个最高深入。

```
private function onMouseDown() {
    var targetObj:Object = Mouse.getTopMostEntity();
    while (targetObj != null && targetObj != _root)
    {
        if (targetObj == this) {
            swapDepths(_parent.getNextHighestDepth());
            return;
        }
        targetObj = targetObj._parent;
    }
}
```

这为实现该行为的一种方法。另一种方法为创建**WindowManager**以维持场景中所有窗口组件的z轴。**WindowManager**方法实际上比本例中窗口组件具有更强的易用性。