

Autodesk® Scaleform®

Scaleform CLIK用戶指南

本文包括了Scaleform CLIK框架及所帶內置元件執行方法的詳細使用說明

作者 : Prasad Silva, Matthew Doyle
版本 : 2.0
最後修訂 : 2010年8月19號

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 4.0 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 錄

1 簡介	6
1.1 概要	6
1.1.1 在Scaleform CLIK包含的內容	6
1.2 元件理解	7
1.3 UI關注	7
2 內置元件	8
2.1 基本按鈕和文本類型	9
2.1.1 Button	10
2.1.2 CheckBox	15
2.1.3 Label	18
2.1.4 TextInput	20
2.1.5 TextArea	24
2.2 分組類型	27
2.2.1 RadioButton	27
2.2.2 ButtonGroup	31
2.2.3 ButtonBar	33
2.3 滾動類型	35
2.3.1 ScrollIndicator	35
2.3.2 ScrollBar	37
2.3.3 Slider	40
2.4 列表類型	41
2.4.1 NumericStepper	43
2.4.2 OptionStepper	45
2.4.3 ListItemRenderer	47
2.4.4 ScrollingList	50
2.4.5 TileList	55
2.4.6 DropdownMenu	59
2.5 進度類型	65
2.5.1 StatusIndicator	65
2.5.2 ProgressBar	67
2.6 其他類型	69
2.6.1 Dialog	69
2.6.2 UILoader	71
2.6.3 ViewStack	73

3 藝術細節	75
3.1 最優方法	75
3.1.1 圖元圖像	75
3.1.2 蒙版	76
3.1.3 動畫	76
3.1.4 圖層和繪製元素	77
3.1.5 複雜皮膚介面	77
3.1.6 2的指數倍	77
3.2 已知問題和推薦工作流程	77
3.2.1 複製元件	77
3.3 皮膚繪製實例	79
3.3.1 StatusIndicator皮膚繪製	80
3.4 字體和本地化	82
3.4.1 概要	82
3.4.2 內置字體	83
3.4.3 在textField嵌入字體	83
3.4.4 本地化系統	83
4 編程詳述	85
4.1 UI元件UIComponent	86
4.1.1 初始化	86
4.2 元件狀態	86
4.2.1 按鈕元件	87
4.2.2 非按鈕交互元件	88
4.2.3 非交互元件	89
4.2.4 特殊案例	89
4.3 事件模型	89
4.3.1 最佳使用方法	89
4.4 焦點處理	90
4.4.1 最佳使用方法	90
4.4.2 在複合附件捕獲焦點	91
4.5 輸入處理	92
4.5.1 最佳使用方法	92
4.5.2 多滑鼠游標	94
4.6 失效	95
4.6.1 最佳使用方法	95
4.7 元件縮放	95
4.7.1 Scale9Grid	96

4.7.2	強制	96
4.8	元件和資料設置	97
4.8.1	最佳使用方法	97
4.9	動態動畫	98
4.9.1	最佳使用方法	98
4.10	彈出式支援	99
4.10.1	最佳使用方法	99
4.11	拖放	99
4.11.1	最佳使用方法	100
4.12	雜項	101
4.12.1	代理Delegate	101
4.12.2	本地Locale	101
5	實例	102
5.1	基礎	102
5.1.1	包含兩個textField的ListItem Renderer	102
5.1.2	圖元滾動視圖	104
5.2	合成	105
5.2.1	使用ScrollingList 的TreeView	105
5.2.2	可複用視窗	107

1 簡介

本文檔提供了Scaleform® 通用精簡介面工具（Common Lightweight Interface Kit，CLIK™）框架和元件的詳細使用說明。在深入認識CLIK用戶指南之前，希望開發者能夠先閱讀[CLIK入門](#)和[CLIK按鈕入門](#)。這兩個使用指南介紹了創建和運行Scaleform CLIK所需要的步驟，介紹了基本的概念並提供了創建和美化CLIK元件的詳細指南。然而，專業級用戶更喜歡在學習入門文檔時直接查閱本文檔作為參考。

1.1 概要

Scaleform CLIK為一組ActionScript™ 2.0 (AS2) 用戶介面元素、庫和工作流增強工具集，Scaleform 4.0用戶為遊戲控制器、PC和掌上遊戲機應用快速實現豐富和高效的介面。框架的主要目標為構建一個精簡（依據記憶體和CPU使用量）、易於繪製皮膚並高度個性化的架構。另外，對於一個基本的UI內核類和系統基本框架，CLIK包含了15個以上內置通用介面元素（例如，按鈕、捲軸、文本輸入框），將幫助開發者快速創建和重構用戶介面介面。

1.1.1. 在Scaleform CLIK包含的內容

元件：簡單擴展內置UI控制元件

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

類：系統核心 API函數

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

文檔和實例文件：

- CLIK入門
- CLIK按鈕入門
- CLIK API 函數參考
- CLIK 用戶指南
- CLIK Flash 實例
- CLIK 視頻指南

1.2 元件理解

在開始之前，開發者需要理解Flash元件的確切技術細節。Flash的一系列默認介面創建工具和編譯塊我們稱之為元件。但是，在本文檔中“元件”指使用Scaleform CLIK框架創建的內置元件，這些元件由Scaleform 與世界知名的gskinner.com團隊聯合開發。

gskinner.com由Grant Skinner領導，Grant Skinner由Adobe任命負責為Flash Creative Suite® 4 (CS4)創建元件，為世界知名的Flash領先開發團隊之一。關於gskinner.com的更多資訊，請訪問<http://gskinner.com/blog>。

為更深入理解內置CLIK元件，在Flash studio中開打默認的CLIK文件：

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

1.3 UI關注

創建一個美觀的UI介面，第一步為在紙面或畫圖編輯器上如Microsoft Visio®上繪製草圖。第二步為在Flash中開始原型化UI介面，其目的為在專門圖形設計之前繪製出所有的介面項和流程圖。Scaleform CLIK就是專門設計用於使用戶快速原型化和重構以完成整個過程。

構建一個完整的UI介面有很多不同的方法。在Flash中，不存在頁的概念，而在Visio或其他流程圖繪製軟體中有此概念，這裏使用了關鍵幀和動畫剪輯代替。如果所有的頁面都在同一個Flash文件中，文件將佔據更多記憶體，但是在頁間切換更加快速且容易。如果每個頁分別在單獨的文件內存放，整體的記憶體使用量可以降低，但是導入時間更長。將每個頁作為單獨的文件也具有一些優勢，可以使多個設計師和

美工人員同時處理相同介面。而且作為技術和設計的需求，在決定UI專案結構時可以確保考慮到工作流程。

與傳統的桌面或web應用不同，這裏特別需要瞭解可以有很多種不同的方法來創建豐富的多媒體遊戲介面。每個引擎，甚至不同的平臺，實現方法也有所不同，有的方法使用起來更加高效，而有的則更加低效。例如，放置一個多頁介面到單個Flash文件。這樣管理起來更加方便，轉換操作也更加容易，但是增加了記憶體的消耗，對於老的遊戲控制器或移動終端則非常不利。如果一切都在單個Flash文件中進行，則不能使多個設計師同時工作在同一個介面的不同部分。如果專案為一個複雜的介面設計，需要一個龐大的設計團隊，或者具有其他特殊的技術和設計需求，則最好將每個UI頁面放置到不同的Flash文件。在很多情況下，這兩種方案都是可行的，但是，在特定的專案中，其中一種可能比另外一種方案更具有優勢。例如，螢幕可能需要根據需求導入和導出，或者在網路上動態更新和傳輸。衡量的底線為應該仔細評估UI介面資源的管理，從UI的邏輯規劃到工作流實現以及性能都需要考慮完善。

雖然Scaleform強烈建議開發者使用Flash和ActionScript作為UI介面主要開發手段，但沒有完美的答案，某些團隊也許更喜歡用應用引擎腳本語言（如Lua或UnrealScript）來完成大多數繁重工作。在這些情況下，Flash應該被主要用作動畫實現工具，只包含極少數量的動態腳本ActionScript和Flash文件內部的交互內容。

在早期瞭解技術、設計和工作流程的需求非常重要，然後繼續評估整個過程的整體方法，特別是在深入專案之前，確保順利和最終的成功。

2 內置元件

初次觀察，Scaleform

CLIK為一個基本的內置UI元件集，但是CLIK的真實意圖是為創建豐富元件和介面提供一個框架。開發者可以自由 - 並可以按設想進行擴展 - 創建自定義元件適應自身需求並在CLIK框架上進行構建。

內置CLIK元件提供標準的UI功能，從基本的按鈕和核取方塊到列表框、下拉功能表和模式對話方塊等複合元件。開發者可以方便得將這些標準元件進行擴展，增加更多特性或在從頭創建自定義元件時作為簡單的參考元件。

以下選項詳細得描述了每個內置元件。他們按照複雜性和功能性進行分組。每個元件使用子章節列表進行描述。

- **User interaction**：用戶如何與元件進行交互。
- **Component setup**：當在Flash授權環境中構造元件時需要的元素。
- **States**：元件到函數所需要的不可視狀態（關鍵幀）。

- **Inspectable properties**：在Flash授權環境中公佈的屬性，便於不使用代碼配置特定元件特性。
- **Events**：在UI介面中其他物件可以監聽的元件所觸發的事件列表。
- **Tips and tricks**：完成各種與討論元件相關的任務的實例代碼。

2.1 基本按鈕和文本類型

基本類型包括Button、CheckBox、Label、TextInput

和TextArea元件。按鈕Button構成了多數用戶介面的主幹，CheckBox繼承了Button的功能。Label為一個靜態標簽類型，而TextInput 和 TextArea分別可以表示單行文本和多行文本。



圖1：來自Free Realms的主功能表按鈕實例

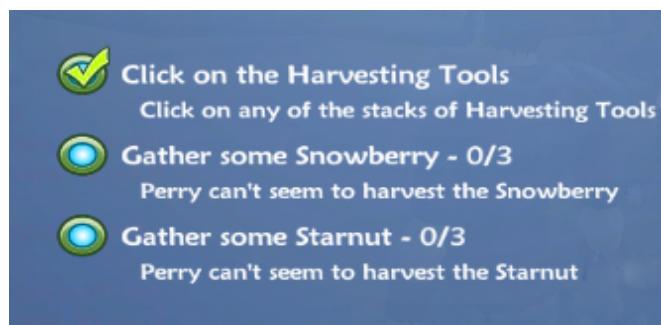


圖2：來自Free Realms核取方塊（Check box）實例



圖3：來自Free Realms的標簽（Label）實例

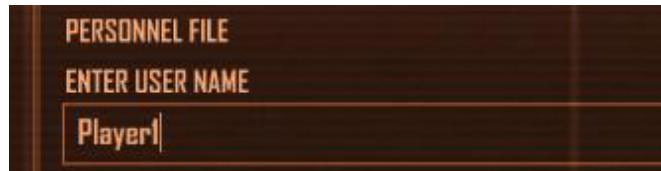


圖4:來自Crysis Warhead的文本輸入框（Text input）實例

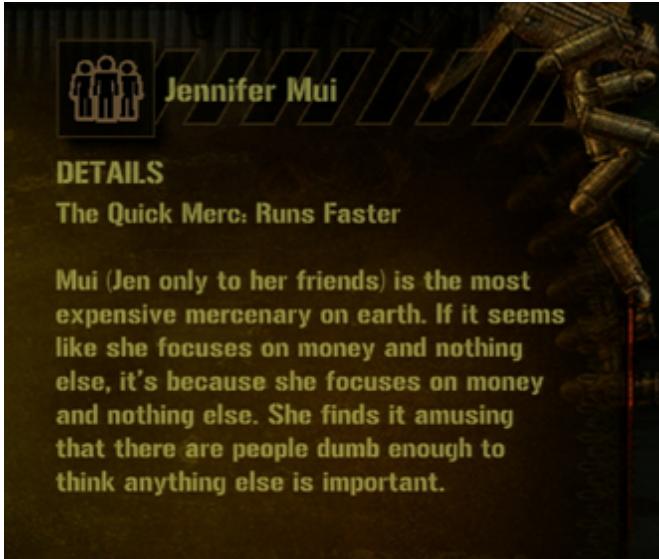


圖5：來自*Mercenaries 2*的文本區域（Text area）實例

2.1.1 Button



圖6：無皮膚按鈕

按鈕Button為CLIK框剪的基本元件，可以在任何地方使用需要一個能發出滴答聲的介面控制。默認的Button類（gfx.controls.Button）支援一個textField來顯示一個標簽，並制定視覺化用戶互動。按鈕可以單獨使用，也可以作為合成元件的一部分，如作為ScrollBar的方向按鈕或者Slider翻頁按鈕。大多數交互元件可以回應點擊動作或為擴展按鈕。

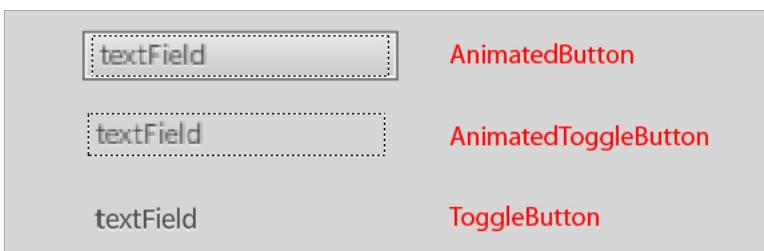


圖7：AnimatedButton、AnimatedToggleButton和ToggleButton

CLIK

Button為一個通用按鈕元件，支援滑鼠交互，鍵盤交互，狀態和其他功能，可以在多種用戶介面中使用。同時也支援選中功能以及動畫狀態。ToggleButton、AnimatedButton和AnimatedToggleButton位於Button.fla元件原始檔案中，使用相同的基本元件類。

2.1.1.1 用戶交互

按鈕元件可以用滑鼠或任何類似控制器點擊。當獲得焦點時也可以通過鍵盤按鈕控制。

通常，只有一個獲得焦點的元件接收鍵盤事件。設置一個元件的焦點有多種方法。其中一些方法在本文的[Programming](#)

[設計細節](#)中有所描述。大多數CLIK元件當交互時，特別是按下滑鼠左鍵或類似控制器在上面按下（點擊）時也可以接收焦點。（Tab）

和(Shift+Tab)建（或對應導航控制）可以在顯示的焦點元件上移動焦點。這種特性也是大多數桌面應用軟體和網頁上所提供的。注意非CLIK元素使用的焦點也可以被CLIK元件使用。這意味著一個Flash開發者能夠將CLIK元素和非CLIK元素在場景相互混合和匹配並使焦點行為按意圖進行，特使在使用(Tab)和(Shift+Tab)鍵時可以在場景中移動焦點。

默認情況下按鈕回應（Enter）鍵和空白鍵。將滑鼠箭頭移動到按鈕上面然後移開也會使元件產生動作，拖動滑鼠游標移入和移出也一樣。

在遊戲控制器或掌上遊戲機，開發者能簡單用對應遊戲杆來控制鍵盤和滑鼠控制事件。例如，（Enter）鍵在Xbox360或

PS3控制器上通常映射為（A）或（X）按鈕。此映射使UI介面中使用CLIK應用到多種類型的平臺之上。

2.1.1.2 元件設置

一個使用CLIK Button類的MovieClip必須具備下面所列的子單元。也列出了可選元素。

- **textField:** (可選) TextField 類型。按鈕標簽
- **focusIndicator:** (可選) MovieClip 類型。一個單獨的MovieClip
用來顯示焦點狀態。如果被使用，必須有兩個命名幀：“show” 和 “hide”。默認情況下，over狀態用來表示一個獲得焦點的Button元件。但是在有些類中，這個行為限制了使用，設計師可能要將焦點狀態和滑鼠over狀態分開使用。

2.1.1.3 狀態

CLIK 按鈕元件支援基於用戶交互的不同的狀態。這些狀態包括：

- **up** 或默認狀態；

- 當滑鼠箭頭在元件上方或者獲得焦點時**over** 狀態；
- 當按鈕按下時**down**狀態；
- **disabled** 狀態；

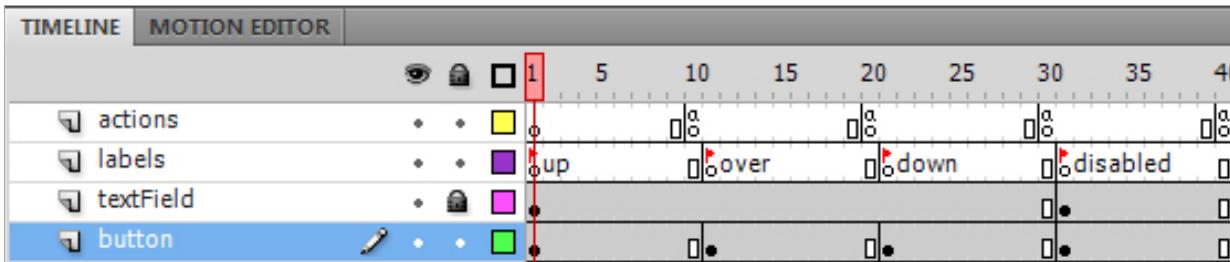


圖8：Button時間軸

這些狀態在Flash時間軸中用關鍵幀來表示，為按鈕元件所需要的最少關鍵幀設置的正確操作。同時還有其餘狀態擴展元件功能以支援複雜用戶交互和動畫轉換，這些資訊在文檔[CLIK 按鈕入門](#)中有所描述。

2.1.1.4 屬性檢查

元件的重要屬性可以通過Flash IDE的Component

Inspector面板或者Parameters標簽進行設置。需要在CS4中打開該面板，在上方工具條中選擇Window下拉功能表，點擊啟動Component Inspector視窗，或者按下(Shift+F7)鍵。這將打開Component Inspector面板。這些被稱之為“檢查屬性”。這為不熟悉AS編程美工和設計師配置元件行為和功能提供了一種便利的方法。

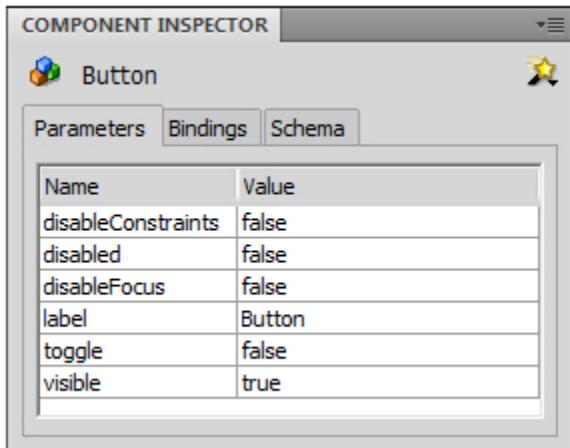


圖9：CS4元件觀察中的按鈕元件屬性檢查視窗

按鈕元件的檢查屬性為：

disableConstraints

按鈕Button元件包含了在元件調整大小時按鈕中的textField的定位和縮放參數。設置該屬性為true則禁止強制物件。這在需要通過時間軸動畫

調整按鈕、 textField 大小或者重新定位特別有用，按鈕大小不會發生變化。如果為 disabled 、 textField 在整個生命周期都將使用其默認參數值，這將禁止在按鈕時間軸上創建的任何 textField 轉換/縮放動作。	
Disabled	如果設置為 true 則禁止按鈕。一旦為 disabled ，則按鈕不接收焦點。
disableFocus	默認情況下接受焦點用於用戶交互。設置該屬性為 true 將禁止焦點獲取。
Label	設置按鈕中的文本顯示。
Toggle	設置按鈕套索模式。如果設置為 true ，按鈕將作為一個套索按鈕使用。
Visible	如果設置為 false 則隱藏按鈕。

改變屬性只能在發佈包含該元件的SWF文件後才有效。**Flash IDE**在設計階段場景中不顯示任何改變，因為**CLIK**元件不屬於編譯剪輯。這是用來確保元件易於使用和繪製皮膚。

在設計階段場景中不顯示任何改變，因為**CLIK**元件不屬於編譯剪輯。這是用來確保元件易於使用和繪製皮膚。

2.1.1.5 事件

大多數元件產生事件用於用戶交互、狀態改變和焦點管理。這些事件在使用**CLIK**元件創建一個功能豐富的用戶介面時非常重要。

所有事件調用接收一個物件參數，包含事件相關資訊。以下為所有事件的常用屬性。

- **type:** 事件類型。
- **target:** 產生事件的目標。

按鈕元件產生事件列表如下。事件旁邊的屬性列表為通用屬性的補充內容。

Show	運行時可視屬性已設置為 true
Hide	運行時可視屬性已設置為 false
focusIn	按鈕獲得焦點
focusOut	按鈕失去焦點
Select	所選屬性已改變。 <i>Selected</i> ：按鈕的選擇狀態， true 為被選擇，為布林類型。
stateChange	按鈕狀態已改變。 <i>State</i> ：按鈕新狀態。String類型。值為“up”、“over”、“down”等。 參考 CLIK按鈕入門 文檔獲取詳細的狀態列表資訊。
rollover	滑鼠箭頭在按鈕上方滾動。 <i>mouseIndex</i> ：用來產生事件的滑鼠游標索引（只應用在多滑鼠-游標環境）。數值類型。值為0-3。

rollout	滑鼠游標從按鈕移開。 <i>mouseIndex</i> ：用戶產生事件的滑鼠游標索引（只應用在多滑鼠游標環境）數值類型。值為0-3。
Press	按鈕被點擊。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
doubleClick	按鈕被雙擊。只在 <i>doubleClickEnabled</i> 屬性為true時被觸發。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-游標環境）。數值類型。值為0-3。
Click	按鈕被點擊。 <i>mouseIndex</i> ：用來產生事件的滑鼠游標索引（只應用在多滑鼠-游標環境）。數值類型。值為0-3。
dragOver	滑鼠游標拖動到按鈕上方（滑鼠左鍵被按下） <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
dragOut	滑鼠箭頭從按鈕拖開（滑鼠左鍵被按下）。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
releaseOutside	滑鼠箭頭從按鈕拖開滑鼠左鍵鬆開。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。

ActionScript代碼片段用來捕獲或處理這些事件。下例中展示了如何處理按鈕點擊事件。

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

第一行代碼為“click”事件安裝事件監聽器，按鈕名稱為‘myButton’，當事件觸發時使其調用onButtonPress函數。相同的代碼類型也可以用來處理其他的事件。事件控制碼中返回的Object參數（例子中名為‘event’）包含了事件的相關資訊。這些資訊作為Object參數屬性返回並與多數事件有所不同。

2.1.1.6 提示和技巧

運行過程中用自定義屬性創建按鈕元件：

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

設置**CLIK**按鈕實例可以在選中和未選中之間切換。這裏不需要通過屬性檢查視窗進行**Toggle**屬性的設置。

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

使能滑鼠左鍵雙擊：

```
btn.doubleClickEnabled = true;
btn.addEventListener("doubleClick", this, "onDoubleClick");
function onDoubleClick(e:Object):Void {
    // 滑鼠被雙擊!
}
```

在按鈕被按下時使能點擊事件的重複觸發：

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

2.1.2 CheckBox



圖 10：無皮膚核取方塊**CheckBox**

核取方塊**CheckBox**

(**gfx.controls.CheckBox**)為一個按鈕元件當被點擊時設置為選中狀態。核取方塊用來顯示**true/false**（布林值）的變化。與**ToggleButton**功能類似，但是隱藏設置**Toggle**屬性。

2.1.2.1 用戶交互

使用滑鼠或者任何相關鍵盤控制器點擊**CheckBox**元件可以使其為選中或未選中。在其他方面，核取方塊行為與按鈕相同。

2.1.2.2 元件設置

使用CLIK CheckBox類的動畫剪輯MovieClip必須具備下面所列的子單元。也列出了可選元素：

- **textField:** (可選) TextField 類型，按鈕標簽。
- **focusIndicator:** (可選) MovieClip 類型，一個獨立的 MovieClip用來顯示焦點狀態。如果被使用，該MovieClip必須有兩個名字為“show”和“hide”的幀。

2.1.2.3 狀態

根據Toggle屬性，核取方塊CheckBox需要另外的關鍵幀集來表示選擇狀態。這些狀態包括：

- **up** 或默認狀態；
- 當滑鼠箭頭在元件上方或者獲得焦點時為**over** 狀態；
- 當按鈕被點擊時候為**down** 狀態；
- **disabled** 狀態；
- **selected_up** 或者默認狀態；
- 當滑鼠箭頭位於元件上方或獲得焦點時為**selected_over** 狀態；
- 當按鈕被按下時為 **selected_down** 狀態；
- **selected_disabled** 狀態；

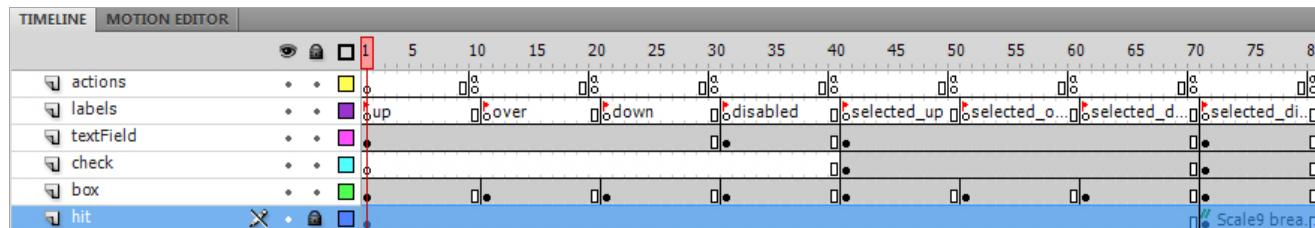


圖 11: CheckBox 時間軸

這裏為核取方塊CheckBox所需要的最少關鍵幀設置。按鈕Button元件支援狀態和關鍵幀的擴展設置，與核取方塊元件相同，在[CLIK 按鈕入門](#)文檔中有詳細描述。

2.1.2.4 屬性檢查

由於從控制按鈕繼承而來，核取方塊CheckBox包含了與按鈕相同的檢查屬性，具有**disableFocus**屬性和**Toggle**屬性。

Data	自定義字串可以配合元件作為獨立資料使用，而不同於核取方塊Check Box上的標簽。
disableConstraints	按鈕元件包含一個在元件調整大小時按鈕內部 textField 的定位和縮放物件。設置該屬性為 true 將禁止該強制物件。這在通過時間軸動畫調整或

	重定位 textField 時特別有用，而按鈕元件不需改變尺寸。如果為 disabled ， textField 在整個生命周期都將使用其默認參數值，這將禁止在按鈕時間軸上創建的任何 textField 轉換/縮放動作。
Disabled	如果設置為 true 禁止按鈕
Label	設置按鈕標簽
Selected	當設置為 true 使得核取方塊 CheckBox 有效（被選中）。
Visible	如果設置為 false 隱藏按鈕。

2.1.2.5 事件

所有的事件調用都接收一個物件**Object**參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 產生事件的目標物件。

核取方塊**CheckBox**元件產生的事件列表如下。事件旁邊的屬性列表為通用屬性的補充內容。

Show	運行時可視屬性已設置為 true
Hide	運行時可視屬性已設置為 false
focusIn	元件獲得焦點
focusOut	元件失去焦點
Select	所選屬性已改變。 <i>Selected</i> ：按鈕的選擇狀態， true 為被選擇，為布林類型。
stateChange	按鈕狀態已改變。 <i>State</i> ：按鈕新狀態。String類型。值為“up”、“over”、“down”等。 參考 CLIK 按鈕入門 文檔獲取詳細的狀態列表資訊。
Rollover	滑鼠箭頭在按鈕上方滾動。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
Rollout	滑鼠箭頭從按鈕移開。 <i>mouseIndex</i> ：用戶產生事件的滑鼠箭頭索引（只應用在多滑鼠箭頭環境）。數值類型。值為0-3。
Press	按鈕被點擊。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
doubleClick	按鈕被雙擊。只在 doubleClickEnabled 屬性為 true 時被觸發。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。

Click	按鈕被點擊。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
dragOver	滑鼠箭頭拖動到按鈕上方（滑鼠左鍵被按下） <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
dragOut	滑鼠箭頭從按鈕拖開（滑鼠左鍵被按下）。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
releaseOutside	滑鼠箭頭從按鈕拖開滑鼠左鍵鬆開。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。

下例中代碼展示了如何處理核取方塊CheckBox的Toggle動作：

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

2.1.3 Label



圖 12: 未繪製皮膚的標簽

CLIK

標簽Label元件(gfx.controls.Label)為一個不可編輯的標準textField元件，由MovieClip符號進行圍繞，具有一些附加的便利特性。在本質上，標簽Label支援與標準textField相同的屬性和行為，但是，有一些常用的特性為該元件本身所特有。如果用戶需要直接改變其屬性，支援訪問標簽實際上的textField區域。在某些情況下，如一些下面內容將會描述的內容，開發者可能會使用textField替代標簽元件。

由於標簽Label為一個MovieClip符號，可以用圖形元素進行修飾，而這在標準的textField無法做到。作為一個符號，不需要如textField實例一樣對每一個進行配置。標簽Label還提供了disabled狀態在時間軸可以被定義。然而，用標準的textField來類比這些功能需要很多複雜的AS2代碼。

標簽Label元件默認強制使用，這意味著在運行時在場景中改變一個標簽Label實例大小不會顯示出效果，開發者應該在大多數情況下使用textField來提到Label標簽。通常，文本元素不需要經常被重用，則textField比起標簽Label使用更加簡單。

2.1.3.1 用戶交互

在標簽Label內無用戶交互。

2.1.3.2 元件設置

使用CLIK Label類的MovieClip必須擁有下列命名的子元素。標注了對應的可選元素：

- **textField**: TextField 類型， Label 標簽文本

2.1.3.3 狀態

CLIK Label元件支援基於標準屬性的兩種狀態：

- **default** 或者enabled 狀態；
- **disabled** 狀態

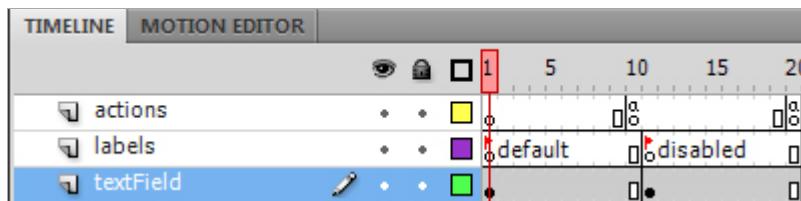


圖 13: Label 時間軸

2.1.3.4 檢查屬性

標簽Label的檢查屬性如下：

text	設置標簽文本
visible	如果設置為false隱藏標簽
disabled	如果設置為true禁止標簽

2.1.3.5 事件

所有事件調用接收單個Object參數包含事件相關資訊。以下為所有事件的通用屬性：

- **type:** 事件類型
- **target:** 產生事件的目標

由標簽Label元件產生的事件列表如下。事件旁邊的屬性列表為通用屬性的補充內容。

Show	運行時元件可視屬性已設置為true
Hide	運行時元件可視屬性已設置為false
stateChange	Label標簽狀態已改變。 State : Label新狀態。String類型。值為“default”或“disabled”。

下面例子為如何監聽標簽Label狀態改變：

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

2.1.3.6 提示和技巧

在標簽Label元件中顯示HTML文本。關於在標準textField 中支援HTML標簽請參考Flash 8文檔：

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

2.1.4 TextInput

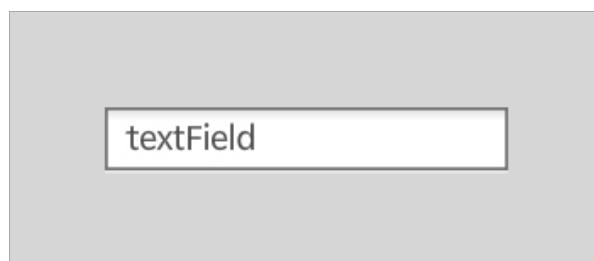


圖 14: 無皮膚TextInput.

文本輸入TextInput

(gfx.controls.TextInput)為一個可編輯的textField元件，用來捕獲用戶文本輸入。與標簽Label類似，該元

件僅僅為一個標準**textField**的外框，因此支援**textField**的功能，如密碼模式、最大字元數和HTML文本。只有一部分屬性為該元件本身所有，其他的屬性可以直接進入**TextInput**的**textField**實例進行更改。

textField元件應該用於輸入，因為無需編輯文本可以使用**Label**標簽來顯示。與**Label**標簽類似，開發者可能會根據自身需求用標準的**textFields**代替**TextInput**元件使用。但是，當開發複雜UI介面時，特別是在PC應用中，**TextInput**元件提供了基於標準**textField**的有價值的功能擴展。

作為特殊屬性，**TextInput**支援焦點和**disabled**狀態，這在標準**textField**中很難實現。根據獨立的焦點狀態，**TextInput**支援自定義焦點指示器，這在標準**TextInput**也不包含。複雜的AS2代碼需要改變標準**TextInput**的外觀類型，而**TextInput**的外觀類型可以在時間軸上簡單得進行配置。**TextInput**檢查屬性為不熟悉Flash Studio的設計師和編程人員提供了一種簡單的工作流程。開發者可以簡便得監聽**TextInput**觸發的事件以創建自定義行為。

TextInput同時也支援**textField**提供的標準選擇和剪切、拷貝和粘貼功能，包括了多行HTML格式文本。默認情況下，快捷鍵命令為選擇 (Shift+Arrows)、剪切 (Shift+Delete)、拷貝 (Ctrl+Insert)、和粘貼(Shift+Insert)。

“文本輸入”可支援滑鼠滾動和滑出事件。可通過對這一特殊的**actAsButton**屬性進行設置，來提供能夠執行兩種滑鼠事件的兩個額外關鍵幀。這些幀被命名為“over”和“out”，分別代表滾動和滑出狀態。如果設置了**actAsButton**模式，並且“over”/“out”幀不存在，那麼“文本輸入”將會按照“預設值”關鍵幀執行這兩種事件。請注意，這些幀不會通過預設的“文本輸入”元件而出現。開發商會根據具體要求對他們進行添加。

當用戶未設定或輸入值時，該元件還支援默認顯示的文本。可將默認文本屬性設置為任何字串。默認文本的主題（顏色和樣式）為淺灰(0xAAAAAA)，斜體。可通過向“默認文本格式”屬性分配一個新的“文本格式”物件的方法對樣式進行自定義。

2.1.4.1 用戶交互

點擊**TextInput**使其獲得焦點，則在**textField**中出現一個箭頭。當箭頭顯現時，用戶能夠通過鍵盤或類似控制設備輸入字元。按下坐右方向鍵可以移動箭頭。當箭頭已經位於**textField**的左邊緣，使用左方向鍵則焦點將轉移到左邊的控制部件。使用右方向鍵也如此。

2.1.4.2 元件設置

使用CLIK **TextInput**類的**MovieClip**必須具備下面所列的子單元。也列出了可選元素：

- **textField:** textField 類型

2.1.4.3 States狀態

CLIK TextInput元件支援三種狀態，均基於焦點和disabled屬性：

- **default** 或enabled 狀態；
- **focused** 狀態，通常為textField周圍突出顯示的邊框；
- **disabled** 狀態。

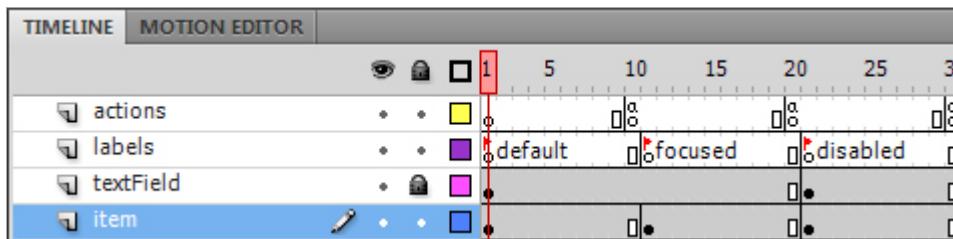


圖 15: TextInput 時間軸

2.1.4.4 檢查屬性

文本輸入TextInput元件的檢查屬性如下所示：

text	設置textField文本
visible	如果設置為false則隱藏元件
disabled	如果設置為true則禁止元件
editable	如果設置為false則使TextInput不可編輯
maxChars	一個大於零的數位，作為textField中可以輸入的最多字元數。
password	如果為true，設置textField顯示“*”字元代替實際字元。而textField接收的值為輸入字元的實際值，返回對應文本。
defaultText	當“文本欄位”為空時所顯示的文本。該文本由“默認文本格式”物件控制，該物件的默認設置為淺灰和斜體。
actAsButton	如果為“真”，則“文本輸入”的行為在未選中狀態下類似一個按鈕，並支援滾動和滑出狀態。一旦按下滑鼠或tab鍵，“文本輸入”將會轉為正常模式，直至退出選中狀態。

2.1.4.5 事件

所有調用接受一個Object參數，包含了事件的相關資訊。以下為事件的通用屬性。

- **type:** 事件類型。
- **target:** 產生事件的目標。

文本輸入TextInput元件產生的事件列表如下。事件旁邊的屬性列表為通用屬性的補充內容。

Show	運行時可視屬性已設置為true
Hide	運行時可視屬性已設置為false
focusIn	元件獲得焦點
focusOut	元件失去焦點
textChange	textField 內容發生改變
rollover	滑鼠游標在元件上滾動。只有當設置了“動作按鈕”後才會啟動。 <i>mouseIndex</i> ：用於生成事件的滑鼠索引（僅適用於多滑鼠游標環境）。數位類型值為0到3.
rollout	滑鼠游標在元件上滑出。只有當設置了“動作按鈕”後才會啟動。 <i>mouseIndex</i> : 用於生成事件的滑鼠索引（僅適用於多滑鼠游標環境）。數位類型值為0到3.

以下代碼為揭示如何監聽textField內容變化的實例：

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

2.1.4.6 提示和技巧

當獲得焦點時停止自動選擇文本：

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

在TextInput元件中顯示HTML文本，關於在標準textField 中支援HTML標簽請參考Flash 8文檔：

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

2.1.5 TextArea

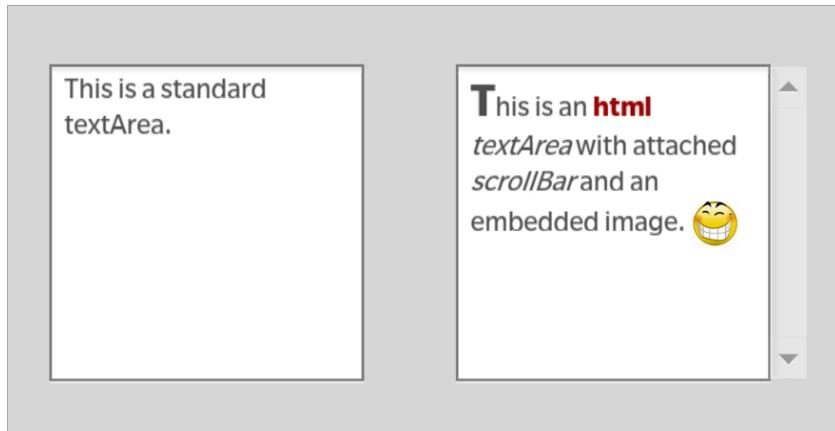


圖 16: 無皮膚TextArea。

文本區域TextArea (`gfx.controls.TextArea`)從CLIK

`TextInput`繼承而來，共用相同功能、屬性和狀態，但是具有一個可選捲軸`ScrollBar`用於多行可編輯滾動文本輸入。請參考“文本輸入”描述，學習更多的有關“文本輸入”和“文本區域”共用的特殊功能。

類似於Label和`TextInput`，`TextArea`也為一個標準的多行`textField`的外框，因此支援`textField`的屬性和行為，如HTML文本、文字邊框、選擇、剪切、拷貝、粘貼。開發者可以簡單得用一個標準的`textField`替代`TextArea`，但是，強烈建議使用該元件，因為其具有擴展功能、狀態、檢查屬性和事件。

儘管標準`textField`能夠用於`ScrollIndicator`或者`ScrollBar`，`TextArea`提供了與這些元件緊湊的組合功能。與標準的`textField`不同，`TextArea`能夠在使用鍵盤或類似控制器使其獲得焦點時進行滾動，甚至在不可編輯時也如此。由於滾動元件不能獲得焦點，`TextArea`能夠展現更多有沒的焦點圖型狀態，能夠在獲得焦點的時候裝飾自身和滾動元件。

2.1.5.1 用戶交互

點擊`TextArea`使其獲得焦點，則在`textField`中出現一個箭頭。當箭頭顯現時，用戶能夠通過鍵盤或類似控制設備輸入字元。按下左右方向鍵可以移動箭頭。當箭頭已經位於`textField`的邊緣，使用方向鍵則焦點將轉移到相鄰的控制部件。

2.1.5.2 元件設置

一個使用CLIK `TextArea`類的MovieClip必須具備下面所列的子單元。也列出了可選元素：

- **`textField`:** `TextField` 類型

2.1.5.3 狀態

與上級元件TextInput類似，TextArea元件支援三種狀態，以焦點和disabled屬性為基礎。

- **default** 或enabled 狀態；
- **focused** 狀態，通常為textField周圍突出顯示的邊框；
- **disabled** 狀態；

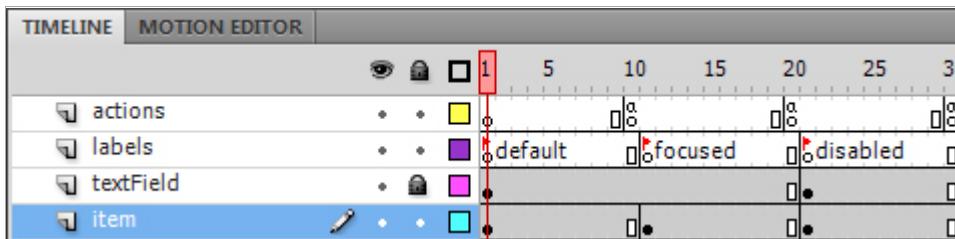


圖 17: TextArea 時間軸

2.1.5.4 檢查屬性

TextArea元件檢查屬性與TextInput類似，具有一對輔助和冗長的密碼特性。輔助特性與CLIK ScrollBar元件有關，將在2.4節描述：

text	設置 textField文本
visible	如果設置為false則隱藏元件
disabled	如果設置為false則禁止元件
editable	如果設置為false則使TextInput不可編輯
maxChars	一個大於零的數位，作為textField中可以輸入的最多字元數。
scrollBar	CLIK ScrollBar元件使用的實例名，或者一個到ScrollBar符號的鏈結ID（本例中由TextArea創建一個實例）。
scrollPolicy	當設置為“auto”時，捲軸將只顯示是否有足夠的文本可以需要滾動。如果設置為“on”則ScrollBar將長期顯示，如果設置為“off”則不顯示，該屬性只影響分配一個ScrollBar的元件（參考ScrollBar特性）
defaultText	當“文本欄位”為空時所顯示的文本。該文本由“默認文本格式”物件控制，該物件的默認設置為淺灰和斜體。
actAsButton	如果為“真”，則“文本輸入”的行為在未選中狀態下類似一個按鈕，並支援滾動和滑出狀態。一旦按下滑鼠或tab鍵，“文本輸入”將會轉為正常模式，直至退出選中狀態。

2.1.5.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

文本區域TextArea元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行時已設置為false
focusIn	元件獲得焦點
focusOut	元件失去焦點
textChange	textField 內容已改變
Scroll	在文本區域滾動
rollover	滑鼠游標在元件上滾動。只有當設置了“動作按鈕”後才會啓動。 <i>mouseIndex</i> ：用於生成事件的滑鼠索引（僅適用於多滑鼠游標環境）。數位類型值為0到3.
rollout	滑鼠游標在元件上滑出。只有當設置了“動作按鈕”後才會啓動。 <i>mouseIndex</i> : 用於生成事件的滑鼠索引（僅適用於多滑鼠游標環境）。數位類型值為0到3.

以下例子展示了如何監聽TextArea滾動事件：

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

2.1.5.6 提示和技巧

當獲得焦點時停止自動選擇文本：

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

在TextArea元件中顯示HTML文本。在標準textField中支援HTML標簽請參考Flash 8 文檔。

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

2.2 分組類型

分組類型包括RadioButton、ButtonGroup和

ButtonBar元件。ButtonGroup為一個管理類型具有特別的邏輯來維護按鈕Button的分組。不具有可視外觀不存在於場景中。但是，ButtonBar存在於場景中也用來維護按鈕分組。RadioButton為一個特殊的按鈕可以自動與同類元件分組到ButtonGroup中。

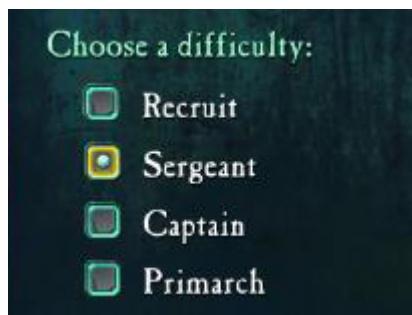


圖 18:來自*Dawn of War II*的選擇按鈕組

2.2.1 RadioButton

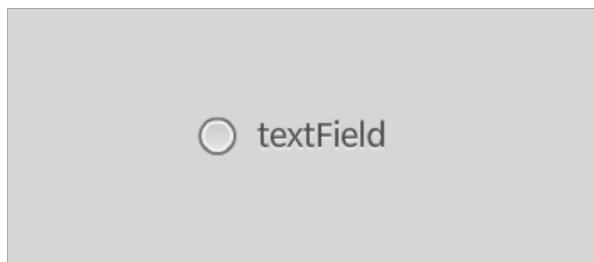


圖 19:無皮膚的選擇按鈕RadioButton

選擇按鈕RadioButton

(gfx.controls.RadioButton)為一個按鈕元件，通常屬於一個集用來顯示和改變一個值。在這個集中只能選擇一個選擇按鈕，點擊集中另外一個選擇按鈕將選中一個新的元件，之前被選中的元件將失去被選擇狀態。

CLIK選擇按鈕與核取方塊CheckBox元件非常類似，共用功能、狀態和行為。主要的區別為選擇按鈕支援分組屬性，可以指派一個自定義按鈕組ButtonGroup（見下節）。選擇按鈕不需要固定設置為選中屬性，因為選中屬性由按鈕組實例進行管理。

2.2.1.1 用戶交互

使用滑鼠或類似控制器點擊未選中的選擇按鈕元件將其選中。如果選擇按鈕為被選中狀態，另外一個同屬一個按鈕組的選擇按鈕被點擊，則先前選中的選擇按鈕將失去選中狀態。其他方面，選擇按鈕行為與按鈕相同。

2.2.1.2 元件設置

使用CLIK RadioButton類的MovieClip必須具備下面所列的子單元。也列出了可選元素：

- **textField**：(可選) TextField 類型，按鈕標簽。
- **focusIndicator**：(可選) MovieClip 類型，一個獨立的 MovieClip 用來顯示焦點狀態。如果被使用，該 MovieClip 必須具有兩個幀名為：“show” 和 “hide”。

2.2.1.3 狀態

由於選擇按鈕可以在選中和未選中狀態間轉換，與核取方塊CheckBox類似，需要至少以下幾種狀態：

- **up** 或默認狀態；
- 當滑鼠箭頭在元件上方或者獲得焦點時為**over** 狀態；
- 當按鈕被點擊時候為**down** 狀態；
- **disabled** 狀態；
- **selected_up** 或者默認狀態；
- 當滑鼠箭頭位於元件上方或獲得焦點時為**selected_over** 狀態；
- 當按鈕被按下時為 **selected_down** 狀態；
- **selected_disabled** 狀態；

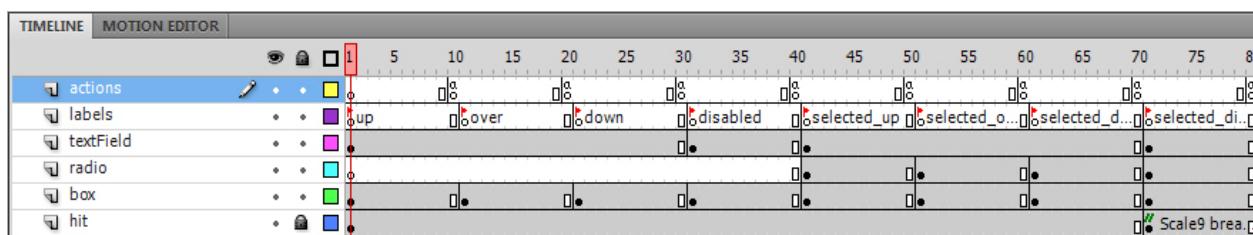


圖 20:選擇按鈕RadioButton時間軸

這裏為選擇按鈕RadioButton所需要的最少關鍵幀設置。按鈕Button元件支援狀態和關鍵幀的擴展設置，與選擇按鈕RadioButton元件相同，在[CLIK 按鈕入門](#)文檔中有詳細描述。

2.2.1.4 檢查屬性

由於從按鈕Button控制按鈕繼承而來，選擇按鈕RadioButton包含了與按鈕相同的檢查屬相，具有disabled Focus屬性和Toggle屬性。

Label	設置按鈕Button的標簽
Visible	如果設置為false則隱藏按鈕
Disabled	如果設置為true則禁止按鈕
disableConstraints	Button元件包含了在元件調整大小時按鈕中的textField的定位和縮放參數。設置該屬性為true則禁止強制物件。這在需要通過時間軸動畫調整按鈕textField大小或者重新定位特別有用，按鈕大小不會發生變化。如果為disabled，textField在整個生命周期都將使用其默認參數值，這將禁止在按鈕時間軸上創建的任何textField轉換/縮放動作。
Selected	當設置為true時確認（或選中）選擇按鈕RadioButton
Data	自定義字串可以配合元件作為獨立資料使用，而不同于選擇按鈕RadioButton上的標簽。
Group	一個名為按鈕組ButtonGroup的實例應該被使用或由選擇按鈕RadioButton自動創建。如果由選擇按鈕創建，新的ButtonGroup將在選擇按鈕容器記憶體在。例如，如果選擇按鈕存在於_root，則選擇按鈕物件將在_root下創建。所有的選擇按鈕使用相同的組屬於同一個集

2.2.1.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

選擇按鈕RadioButton元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行時已設置為false
focusIn	元件獲得焦點
focusOut	元件失去焦點
Select	所選屬性已改變。 <i>Selected</i> ：按鈕的選擇狀態，true為被選擇，為布林類型。
stateChange	按鈕狀態已改變。 <i>State</i> ：按鈕新狀態。String類型。值為“up”、“over”、“down”等。參考 CLIK按鈕入門 文檔獲取詳細的狀態列表資訊。

rollover	滑鼠箭頭在按鈕上方滾動。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
rollOut	滑鼠箭頭從按鈕移開。 <i>mouseIndex</i> ：用戶產生事件的滑鼠箭頭索引（只應用在多滑鼠箭頭環境）數值類型。值為0-3。
Press	按鈕被點擊。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
doubleClick	按鈕被雙擊。只在 <i>doubleClickEnabled</i> 屬性為true時被觸發。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
Click	按鈕被點擊。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
dragOver	滑鼠箭頭拖動到按鈕上方（滑鼠左鍵被按下） <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
dragOut	滑鼠箭頭從按鈕拖開（滑鼠左鍵被按下）。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
releaseOutside	滑鼠箭頭從按鈕拖開滑鼠左鍵鬆開。 <i>mouseIndex</i> ：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。

以下例子顯示如何處理選擇按鈕RadioButton的選中狀態：

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

2.2.1.6 提示和技巧

選擇按鈕RadioButton組不在相同的層：

```
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

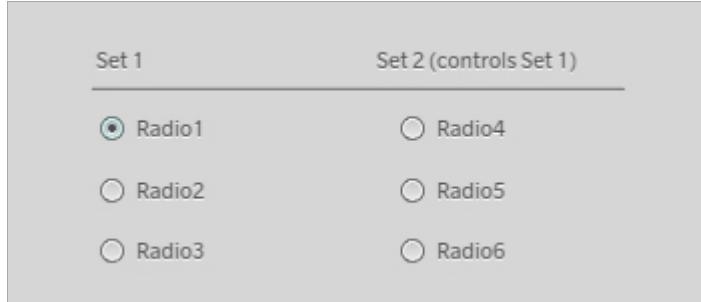


圖 21: 無皮膚按鈕分組 **ButtonGroup**.

CLIK ButtonGroup

(gfx.controls.ButtonGroup)本身不是一個元件，但是有重要作用用於管理按鈕集。可以使在集中的一個按鈕被選中，確保其餘的未選中。如果用戶選擇集中的另一個按鈕，則當前選中按鈕將變為未選中。任何從CLIK按鈕元件繼承的元件（如核取方塊CheckBox和選擇按鈕RadioButton）能夠使用按鈕分組ButtonGroup實例。

2.2.2.1 用戶交互

按鈕組不具有用戶交互功能，因為不是可視元件。但是，在下屬的選擇按鈕RadioButton被點擊時起到間接的作用。

2.2.2.2 元件設置

使用CLIK按鈕組ButtonGroup類的MovieClip不需要任何子單元，因為不具有視覺化外觀。

2.2.2.3 狀態

按鈕組ButtonGroup在場景中不具有可視外觀。因此無關聯狀態。

2.2.2.4 檢查屬性

按鈕組ButtonGroup在場景中不具有可視外觀。因此無需檢查屬性。

2.2.2.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

ButtonGroup元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

change 在組中選擇一個新的按鈕

- *item*：選擇按鈕，CLIK按鈕Button類型
- *data*：選擇按鈕的數值，AS2物件類型

itemClick 組中的按鈕被點擊

- *item*：選擇按鈕，CLIK按鈕Button類型

下例展示了如何判斷按鈕組ButtonGroup中哪個按鈕被選中：

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

2.2.2.6 提示和技巧

從組中找出當前選中的選擇按鈕：

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

在場景中為按鈕組ButtonGroup中所屬的默認選擇按鈕安裝監聽器

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

2.2.3 ButtonBar

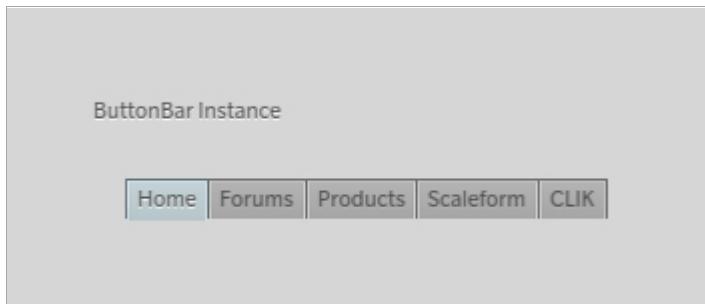


圖 22: 無皮膚按鈕欄 ButtonBar.

按鈕欄ButtonBar

(gfx.controls.ButtonBar)與按鈕組ButtonGroup類似，儘管具有視覺化外觀。也可以基於資料源dataProvider（參見[編程細節](#)描述dataProvider章節）動態創建按鈕實例。按鈕欄可用與創建動態tab欄UI元素。

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

2.2.3.1 用戶交互

按鈕欄與按鈕組ButtonGroup具有相同的行為，也不能與用戶直接交互，按鈕欄也由按鈕點擊事件簡介影響。

2.2.3.2 元件設置

使用CLIK按鈕欄ButtonBar類的視頻剪輯MovieClip不需要任何子單元，因為不具有視覺化外觀。

2.2.3.3 狀態

CLIK按鈕欄不具有任何可視狀態，因為其管理按鈕元件只用來顯示組的狀態。

2.2.3.4 檢查屬性

儘管按鈕欄元件沒有內容（只為Flash

IDE場景中的一個簡單小圓圈），但包含幾個檢查屬性。主要由按鈕欄ButtonBar創建的按鈕實例分佈設置決定。

Visible	如果設置為false則隱藏ButtonBar
Disabled	如果設置為true則禁止buttonBar
itemRenderer	按鈕元件符號的鏈結ID，該符號將根據按鈕欄的資料分配需求進行實例化

Direction	按鈕位置，水平並排放置按鈕實例或垂直疊放
Spacing	按鈕實例間隔，只影響當前方位（見 <i>direction</i> 屬性）
autoSize	如果設置為true，重新調整按鈕實例以適應顯示標簽
buttonWidth	設置所有的按鈕實例為常用寬度，如果autoSize設置為true忽略該屬性

2.2.3.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

ButtonBar元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行時已設置為false
focusIn	元件獲得焦點
focusOut	元件失去焦點
change	在組中選擇一個新的按鈕 <ul style="list-style-type: none"> • <i>index</i>：按鈕欄的選擇序列，數值類型，值為0到最小為1的按鈕數 • <i>renderer</i>：選擇的按鈕，CLIK按鈕Button類型 • <i>item</i>：資料源dataProvider選擇項，AS2 Object類型 • <i>data</i>：選擇資料源dataProvider的數值，AS2 Object類型
itemClick	在組中點擊按鈕 <ul style="list-style-type: none"> • <i>index</i>：點擊按鈕的按鈕欄序列，數值類型，值為0到最小為1的按鈕數 • <i>item</i>：資料源dataProvider選擇項，AS2 Object類型 • <i>data</i>：選擇資料源dataProvider的數值，AS2 Object類型 • <i>mouseIndex</i>：用來產生事件的滑鼠游標序號（只在多滑鼠游標環境中使用），數值類型，值為0-3

以下例子展示了當在按鈕欄中的按鈕實例被點擊後如何進行事件監聽。

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

2.3 滾動類型

滾動類型包括ScrollIndicator、ScrollBar和

Slider。滾動指示器ScrollIndicator無需交互用來顯示目標物件元件的滾動位置，而捲軸ScrollBar支援用戶交互來改變滾動位置。捲軸由四個按鈕組成：翻頁按鈕、軌道、向上方向箭和向下方向鍵。滑動條Slider與捲軸類似，但是只包含一個交互的翻頁按鈕和軌道，不根據目標元件的單元數來改變翻頁按鈕大小。

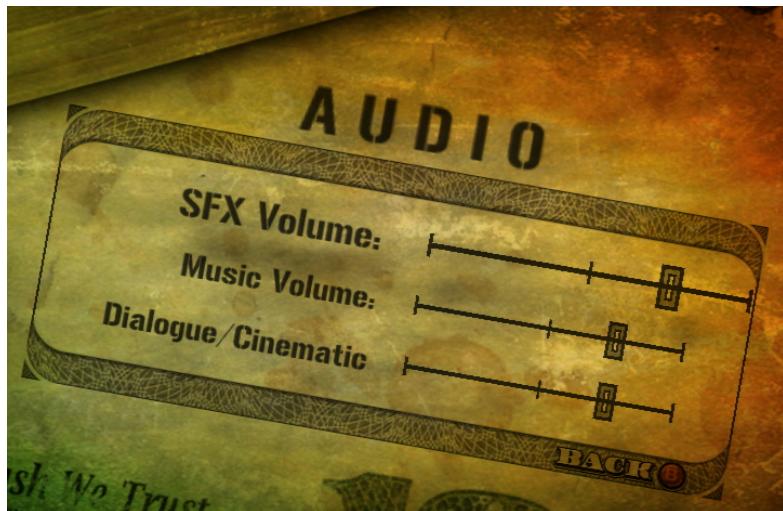


圖 23: 來自 **Mercenaries 2** 的音頻功能表滑動條實例

2.3.1 ScrollIndicator



圖 24: 無皮膚 ScrollIndicator.

CLIK滾動指示器ScrollIndicator

(gfx.controlsScrollIndicator)顯示了其他元件的滾動位置，如多行文本區域TextField。可以在文本區域用來自動顯示滾動位置。所有基於列表的元件，包括文本區TextArea，具有捲軸屬性可以由滾動指示器或捲軸（見下節）實例或鏈結ID進行顯示。

2.3.1.1 用戶交互

滾動指示器與具有用戶交互功能，因為只用來顯示。

2.3.1.2 元件設置

一個使用CLIK ScrollIndicator類的MovieClip必須具備下面所列的子單元。也列出了可選元素：

- **thumb**: CLIK按鈕Button類型，滾動指示塊。
- **track** : 動畫剪輯MovieClip類型，滾動指示軌道，軌道的邊界決定了滾動塊可以移動的位置。

2.3.1.3 狀態

滾動指示器沒有顯性狀態，使用子單元的狀態：滾動塊和軌道按鈕元件。

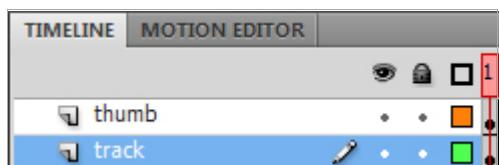


圖 25:滾動指示器 ScrollIndicator 時間軸

2.3.1.4 檢查屬性

滾動指示器的檢查屬性如下所示：

scrollTarget	設置一個文本區域TextArea或者常用多行文本域textField作為滾動目標自動回應滾動事件。非文本域textField類型需要手動更新滾動指示器ScrollIndicator屬性。
Visible	如果設置為false則隱藏元件
Disabled	如果設置為false則禁止元件
offsetTop	拖動點頂部偏移。正值將使拖動點向頂部位置的更高處移動。
offsetBottom	拖動點底部偏移。正值將使拖動點向底部位置的更低處移動。

2.3.1.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。

- **target:** 事件產生的目標。

ScrollIndicator元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行時已設置為false
Scroll	滾動位置發生改變。 <ul style="list-style-type: none"> • <i>position</i>：新滾動位置，數值類型，值為最小位置到最大位置

下例顯示如何監聽滾動事件：

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

2.3.1.6 提示和技巧

設置一個獨立的手動滾動指示實例：

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

設置滾動方向，使用`_rotation`屬性在場景中創建元件時自動設置。如果滾動指示器元件不可旋轉默認為水平方向，則需要明確的設置該值。

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

2.3.2 ScrollBar

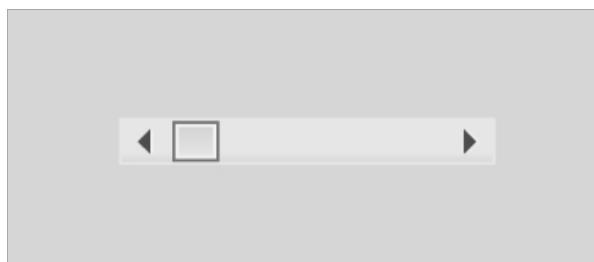


圖 26: 無皮膚捲軸ScrollBar.

CLIK捲軸ScrollBar(gfx.controls.ScrollBar)顯示和控制其他元件的滾動位置。通過拖動滾動塊按鈕增加交互功能到滾動指示器，同時還包括向上和向下方向鍵和一個可以點擊的軌道。

2.3.2.1 用戶交互

捲軸ScrollBar上的滾動塊可以由滑鼠或類似控制器進行控制，可以在捲軸軌跡邊界之內拖動。當滑鼠游標位於捲軸上方時移動滑鼠滾輪可以進行滾動操作。點擊向上方向鍵可以向上滾動，點擊向下方向鍵可以向下滾動。點擊軌道有兩種行爲：滾動塊繼續沿點擊點位置滾動，或者立即跳轉到點擊點位置並設為拖動狀態。軌道模式由*trackMode*檢查屬性決定（見檢查屬性小節）。忽略軌道模式*trackMode*設置，按下(Shift)鍵並點擊軌道將立即將滾動快移動到滑鼠游標處並設為拖動狀。

2.3.2.2 元件設置

一個使用CLIK ScrollBar類的MovieClip必須具備下面所列的子單元。也列出了可選元素：

- **upArrow**: CLIK 按鈕類型，向上滾動按鈕；通常位於捲軸頂部。
- **downArrow**: CLIK 按鈕類型，向下滾動按鈕；通常位於捲軸底部。
- **thumb**: CLIK按鈕類型，捲軸上的滾動塊。
- **track**: CLIK按鈕類型，捲軸軌道，其邊界決定了滾動塊的活動方位。

2.3.2.3 狀態

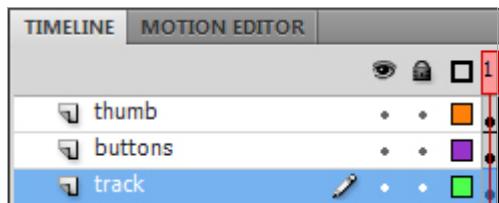


圖 27: 捲軸ScrollBar 時間軸

捲軸ScrollBar，與滾動指示器ScrollIndicator類似，不需要顯性狀態。使用子單元的狀態包括：滾動塊、向上按鈕、向下按鈕和軌道按鈕元件。

2.3.2.4 檢查屬性

捲軸檢查屬性和滾動指示器類似，只增加一條：

scrollTarget	回應滾動事件時設置文本區TextArea或常用多汗文本域textField的滾動目標位置
trackMode	當用戶用滑鼠點擊捲軸，滾動頁scrollPage設置將導致滾動塊翻頁知道釋放師表。scrollToCursor設置可以使滾動塊立即跳轉到滑鼠游標所在位置並轉入拖動模式直到釋放滑鼠。
Visible	如果設置為false則隱藏元件
Disabled	如果設置為false則禁止元件
offsetTop	拖動點頂部偏移。正值將使拖動點向頂部位置的更高處移動。

offsetBottom 拖動點底部偏移。正值將使拖動點向底部位置的更低處移動。

2.3.2.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

ScrollBar元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

show	元件可視屬性在運行時已設置為true
hide	元件可視屬性在運行時已設置為false
scroll	滾動位置已改變。 <ul style="list-style-type: none">• <i>position</i>：新滾動位置，數值類型，值為最小位置到最大位置之間。

下列代碼指示如何監聽滾動事件：

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

2.3.2.6 提示和技巧

手動設置一個獨立的滾動指示器：

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

設置滾動方向，使用`_rotation`屬性在場景中創建元件時自動設置。如果滾動指示器元件不可旋轉默認為水平方向，則需要明確的設置該值。

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

設置軌道在`scrollPage`模式下被點擊時的滾動位置值。預設值為1：

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

2.3.3 Slider

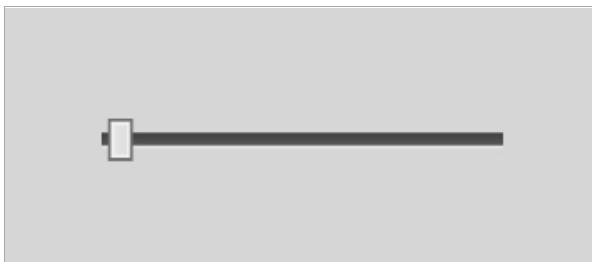


圖 28: 無皮膚滑動條Slider。

滑動條Slider(gfx.controls.Slider)顯示了一個範圍內的數值，用滾動塊來表示值，通過拖動來改變值。

2.3.3.1 用戶交互

滑動塊可以被滑鼠或類似控制器在滑動軌道邊界內拖動。在軌道上點擊可以立即移動滾動塊到滑鼠游標所在位置並設置為拖動狀。當獲得焦點時，左右方向鍵在對應的方向移動滾動塊，而home和end鍵可以將滾動塊移動到捲軸的開始和末尾處。

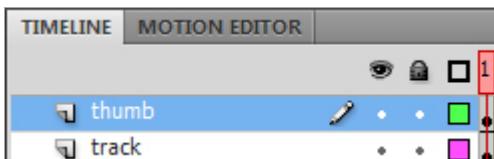
2.3.3.2 元件設置

使用CLIK Slider類的MovieClip必須具備下面所列的子單元。也列出了可選元素：

- **thumb** : CLIK 按鈕類型，滑動快。
- **Track** : CLIK 按鈕類型，滑動軌道邊界決定了滑動塊可以移動位置。

2.3.3.3 狀態

與滾動指示器和捲軸類似，滑動條沒有顯性狀態，使用子單元的狀態包括：滾動塊和軌道按鈕元件。



□ 29:滑动条 Slider 时间轴

2.3.3.4 檢查屬性

滑動條Slider元件的檢查屬性如下所示：

Visible 如果設置為false則隱藏元件

Disabled 如果設置為false則禁止元件

Value	滑動條Slider顯示的數值
Minimum	滑動條範圍內的最小值
Maximum	滑動條範圍內的最大值
Snapping	如果設置為true，滾動快將按照多被間隔進行移動
snapInterval	滾動間隔決定滾動塊跳轉間隔函數，設置為false時該值無效
liveDragging	如果設置為true，拖動滾動塊時則滑動條Slider將產生一個改變事件，滑動條只在拖動結束後產生事件改變
offsetLeft	拖動點向左偏移。正值將會向內擠壓拖動點。
offsetRight	拖動點向右偏移。正值將會向內擠壓拖動點。

2.3.3.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

Slider元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

show	元件可視屬性在運行時已設置為true
hide	元件可視屬性在運行時已設置為false
focusIn	元件獲得焦點
focusOut	元件失去焦點
change	改變滑動條Slider值

下例顯示了如何監聽滑動條Slider值的改變：

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

2.4 列表類型

CLIK列表類型包括NumericStepper、 OptionStepper、 ScrollingList、 TileList 和DropdownMenu元件。所有元件，除了NumericStepper，都用DataProvider來管理資訊並顯示。ListItemRenderer元件也包括在此目錄中，因為ScrollingList 和TileList元件需要用來顯示列表專案。

NumericStepper和 OptionStepper元件只用一次顯示一個單元，而ScrollingList 和TileList能顯示多個單元。最後兩個元件可以支援ScrollIndicator或 ScrollBar元件。DropdownMenu元件在靜止狀態顯示一個單元，但是在打開時使用ScrollingList 或 TileList可以顯示多個單元。



圖 30:來自 *Mercenaries 2*的滾動指示器滾動列表實例

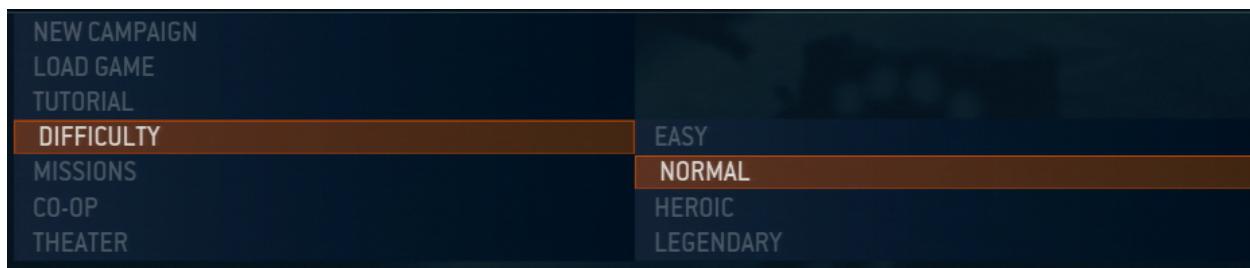


圖 31:來自 *Halo Wars*的'Difficulty Settings'下拉功能表實例

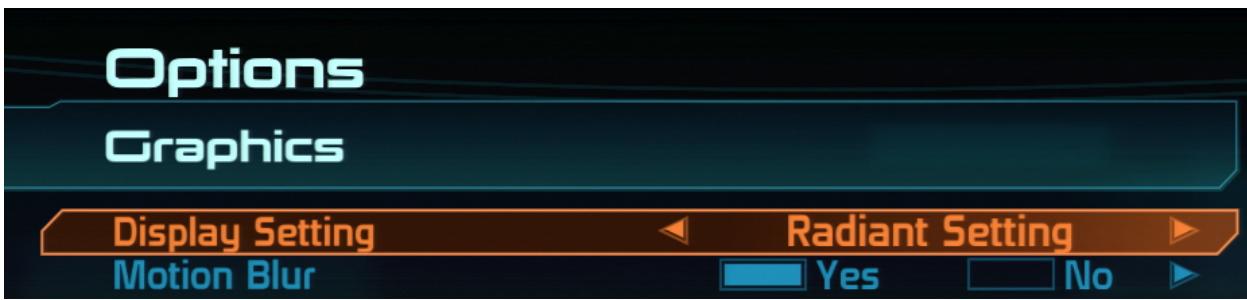


圖 32:來自 *Mass Effect*的'Display Setting'選項實例

2.4.1 NumericStepper



圖 33:無皮膚Numeric Stepper.

NumericStepper

(gfx.controls.NumericStepper)顯示在分配範圍內的一個數位，支援任意步長值的增加和減少。

2.4.1.1 用戶交互

NumericStepper包括兩個方向鍵可以通過滑鼠或類似控制器點擊以改變數位值。當獲得焦點時，數位值可以通過鍵盤左右方向鍵或類似控制器改變。這些鍵可以根據步長增加和減少當前值。按下(Home)和(End)鍵或類似控制器將在對應的最大值和最小值之間改變數位值。

2.4.1.2 元件設置

使用CLIK NumericStepper類的MovieClip必須具備下面所列的子單元。也列出了可選元素：

- **textField:** TextField 類型，用來顯示當前值。
- **nextBtn:** CLIK 按鈕Button類型，點擊可以根據步長增加當前值。
- **prevBtn:** CLIK 按鈕Button類型，點擊可以根據步長減少當前值。

2.4.1.3 狀態

NumericStepper元件支援三種狀態，基於焦點和禁止屬性：

- **default**或 **enabled** 狀態；
- **focused** 狀態，突出顯示**textField** 區域；
- **disabled** 狀態；

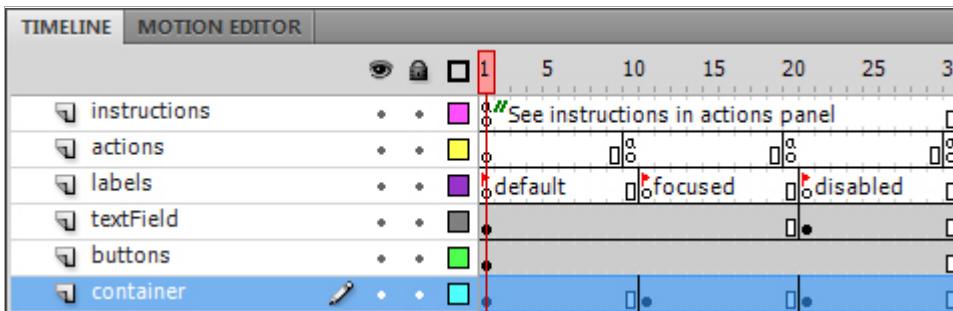


圖34: NumericStepper 時間軸

2.4.1.4 檢查屬性

從數位步長NumericStepper元件繼承來的動畫剪輯MovieClip將有以下檢查屬性：

visible	如果設置為false則隱藏元件
disabled	如果設置為false則禁止元件
value	NumericStepper.顯示的數值
minimum	NumericStepper數值範圍內的最小值
maximum	NumericStepper數值範圍內的最大值

2.4.1.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

NumericStepper元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行時已設置為false
Change	NumericStepper數值發生改變
stateChange	NumericStepper焦點或禁止屬性發生改變。 <ul style="list-style-type: none"> • state : 新狀態名，String類型，值為“default”、“focused” 或 “disabled”。

下例顯示了對NumericStepper值改變的監聽：

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

2.4.1.6 提示和技巧

改變增加/減少間隔：

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

增加數值尾碼，例如：

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

2.4.2 OptionStepper



圖 35: 無皮膚OptionStepper.

OptionStepper

(gfx.controls.OptionStepper),與NumericStepper類似，用來顯示一個值，但可以顯示更多資訊。使用資料源dataProvider實例查詢當前值；因此支援不同類型元素的任意數值。應用OptionStepper的選擇項索引屬性通過代碼對顯示值進行設置，該索引是附在資料提供者中的一個從零開始的索引。資料源dataProvider通過代碼進行指派，如下例所示：

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

2.4.2.1 用戶交互

與NumericStepper類似，OptionStepper包括兩個方向鍵，可以通過滑鼠或類似控制器點擊改變當前值。當獲得焦點時，當前值可以通過左右方向鍵或類似控制器進行改變。這些鍵按照順序向前或者向後改變當前值。按下(Home)和(End)鍵或類似控制器可以將當前值改變為資料源dataProvider的第一個和最後一個單元。

2.4.2.2 元件設置

使用CLIK NumericStepper類的動畫剪輯MovieClip必須具備下面所列的子單元。也列出了可選元素：

- **textField** : TextField類型，用來顯示當前值。
- **nextBtn** : CLIK按鈕 Button類型，改變當前值為資料源dataProvider中的下一個值。
- **prevBtn** : CLIK 按鈕 Button類型，當前值為資料源dataProvider中的上一個值。

2.4.2.3 狀態

OptionStepper元件支援三種狀態，基於焦點和禁止屬性：

- **default**或 enabled 狀態；
- **focused** 狀態，突出顯示textField 區域；
- **disabled** 狀態；

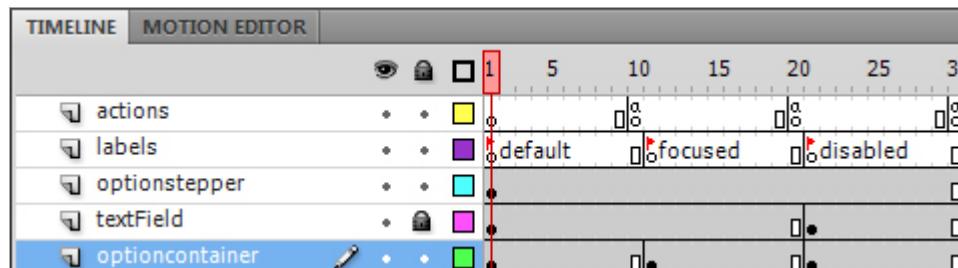


圖 36: OptionStepper 時間軸

2.4.2.4 檢查屬性

來自選項步長OptionStepper元件的視頻剪輯MovieClip具有以下檢查屬性。

Visible	如果設置為false則隱藏元件
Disabled	如果設置為false則禁止元件

2.4.2.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

OptionStepper元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行時已設置為false
Change	OptionStepper值發生改變
stateChange	OptionStepper焦點或禁止屬性發生改變。 <ul style="list-style-type: none">• state: 新狀態名稱，String類型，值為“default”、“focused”或“disabled”。

下列顯示了OptionStepper值改變的監聽：

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

2.4.3 ListItemRenderer



图 1:无皮肤 ListItemRenderer.

列表項渲染器ListItemRenderer

(gfx.controls.ListItemRenderer)從CLIK按鈕Button類繼承而來，擴展了列表相關特性，在容器元件中需要用到。但是，不是作為一個單獨組建設計，只與滾動列表ScrollingList、標題列表TitleList和下拉功能表DropdownMenu元件共同使用。

2.4.3.1 用戶交互

由於ListItemRenderer從按鈕Button元件繼承而來，與按鈕具有相同的用戶交互如使用滑鼠點擊。滾動滑鼠游標到ListItemRenderer或將游標移開都將對元件產生影響，拖動滑鼠游標效果也一樣。ListItemRenderer的容器元件定義了鍵盤或類似控制器的交互。

2.4.3.2 元件設置

使用CLIK ListItemRenderer類的動畫剪輯MovieClip必須具備下面所列的子單元。也列出了可選元素：

- **textField** : (可選) TextField 類型，列表項標簽。
- **focusIndicator**: (可選) MovieClip類型，一個獨立的動畫剪輯
MovieClip用來顯示焦點狀態。如果被使用，該動畫剪輯必須擁有兩個幀分別命名為：“show”和“hide”。

2.4.3.3 狀態

由於可以在一個容器元件內部被選擇，列表項渲染器ListItemRenderer需要關鍵幀為**selected**設置來表示其選擇狀態。元件狀態包括：

- **up** 或默認狀態；
- 當滑鼠箭頭在元件上方或者獲得焦點時為**over** 狀態；
- 當按鈕被點擊時候為**down** 狀態；
- **disabled** 狀態；
- **selected_up** 或者默認狀態；
- 當滑鼠箭頭位於元件上方或獲得焦點時為**selected_over** 狀態；
- 當按鈕被按下時為 **selected_down** 狀態；
- **selected_disabled** 狀態；

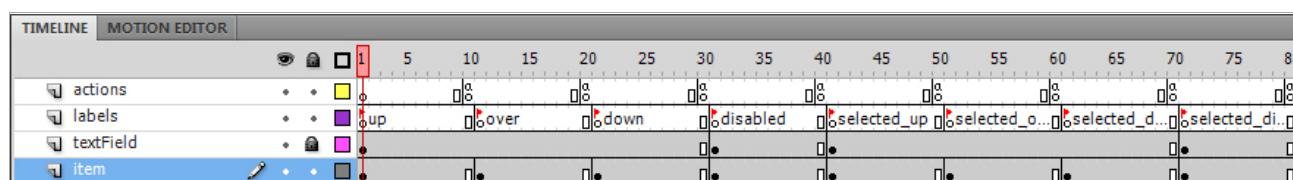


圖 38: ListItemRenderer 時間軸

這裏為選擇按鈕ListItemRenderer所需要的最少關鍵幀設置。按鈕Button元件支援狀態和關鍵幀的擴展設置，與ListItemRenderer元件相同，在[CLIK 按鈕入門](#)文檔中有詳細描述。

2.4.3.4 檢查屬性

由於列表項渲染器ListItemRenderer由一個容器元件按控制，用戶無需手動配置，值包含了按鈕的一小部分檢查屬性。

Label	ListItemRenderer標簽設置
Visible	如果設置為false則隱藏按鈕
Disabled	如果設置為true則禁止按鈕

2.4.3.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

列表項渲染器ListItemRenderer元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行時已設置為false
focusIn	元件獲得焦點
focusOut	元件失去焦點
Select	所選屬性已改變。 <ul style="list-style-type: none">• Selected: 按鈕的選擇狀態，true為被選擇，為布林類型。
stateChange	按鈕狀態已改變 State：按鈕新狀態。String類型。值為“up”、“over”、“down”等。 參考 CLIK 按鈕入門 文檔獲取詳細的狀態列表資訊。
Rollover	滑鼠箭頭在按鈕上方滾動。 <ul style="list-style-type: none">• mouseIndex: 用來產生事件的滑鼠箭頭索引（只應用在多滑鼠箭頭環境）。數值類型。值為0-3。
Rollout	滑鼠箭頭從按鈕移開。 <ul style="list-style-type: none">• mouseIndex: 用戶產生事件的滑鼠箭頭索引（只應用在多滑鼠箭頭環境）數值類型。值為0-3。
Press	按鈕被點擊。 <ul style="list-style-type: none">• mouseIndex: 用來產生事件的滑鼠箭頭索引（只應用在多滑鼠箭頭環境）。數值類型。值為0-3。
doubleClick	按鈕被雙擊。只在 doubleClickEnabled 屬性為true時被觸發。 <ul style="list-style-type: none">• mouseIndex: 用來產生事件的滑鼠箭頭索引（只應用在多滑鼠箭頭環境）。

	箭頭環境)。數值類型。值為0-3。
Click	按鈕被點擊。 <ul style="list-style-type: none"> • <i>mouseIndex</i>：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
dragOver	滑鼠箭頭拖動到按鈕上方（滑鼠左鍵被按下） <ul style="list-style-type: none"> • <i>mouseIndex</i>：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
dragOut	滑鼠箭頭從按鈕拖開（滑鼠左鍵被按下）。 <ul style="list-style-type: none"> • <i>mouseIndex</i>：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。
releaseOutside	滑鼠箭頭從按鈕拖開滑鼠左鍵鬆開。 <ul style="list-style-type: none"> • <i>mouseIndex</i>：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠-箭頭環境）。數值類型。值為0-3。

2.4.4 ScrollingList



圖 39: 無皮膚滾動列表ScrollingList.

滾動列表ScrollingList

(gfx.controls.ScrollingList)為一個元件可以滾動其元素。可以自身展示列表項或者使用現存的場景中的列表項。可以附加滾動指示器ScrollIndicator

或捲軸ScrollBar元件提供滾動反饋和控制功能。該元件可與資料源dataProvider一起使用。資料源通過代碼指派，如下面實例所示：

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

默認情況下滾動列表crollingList使用列表項渲染器ListItemRenderer元件作為內容。因此列表項渲染器必須在FLA文件庫中存在，除非列表項渲染器檢查屬性改變成另外一個元件。見檢查屬性小節獲得更多資訊。

2.4.4.1 用戶交互

點擊一個列表項或者一個附屬的捲軸ScrollBar實例將焦點傳遞到滾動列表ScrollingList元件。當獲得焦點，按下向上向下方向鍵或類似的控制器控制列表選項的滾動選擇一個元素。如果沒有選中元素，則默認選中頂部元素。如果游標位於滾動列表ScrollingList頂部則滑鼠滾輪能夠在列表中滾動。

在邊界上的滾動行爲由滾動列表ScrollingList的*wrapping*屬性決定，無檢查項。如果*wrapping*設置爲“*normal*”，當選項達到列表開始處或者結束處則焦點將離開元件。如果*wrapping*設置爲“*wrap*”，則選項將圍繞開始處或結束處。如果*wrapping*設置爲“*stick*”，則選項在達到資料結尾時停止，焦點不轉移到相鄰元件。

按下鍵盤(Page Up) 和 (Page

down)或類似控制器將按頁滾動選項，例如，列表中的可視選項數。按下(Home) 和 (End)鍵，或者類似控制器，將滾動列表到相應元素的開始和末尾處。與附屬捲軸ScrollBar元件交互將按預期影響滾動列表ScrollingList。見捲軸ScrollBar小節瞭解其用戶交互功能。

開發者可以簡單得將遊戲控制杆映射到鍵盤和滑鼠控制鍵。例如，鍵盤方向鍵通常引導控制器上的D-Pad。這個對應關係可以使CLIK構建的UI介面工作在更加廣泛的平臺之上。

2.4.4.2 元件設置

滾動列表ScrollingList不需要任何子單元，但是，在場景上的滾動列表ScrollingList元件上放置一個元件或者調整元件大小時一個視覺化背景就能起到輔助作用。

2.4.4.3 狀態

滾動列表ScrollingList元件支援三種狀態，基於焦點和禁止屬性：

- **default** 或enable狀態；
- **focused**狀態，通常突出顯示元件邊框區域；
- **disabled** 狀態。

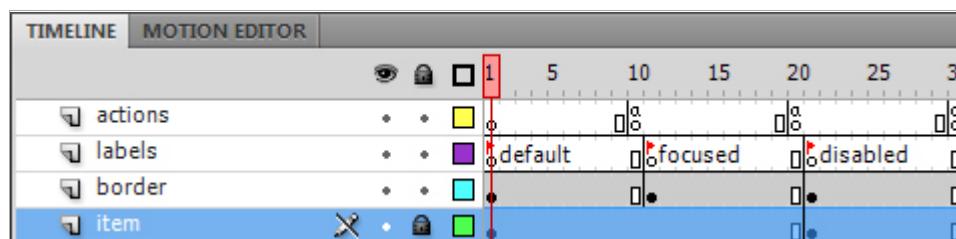


圖40:滾動列表 ScrollingList 時間軸

2.4.4.4 檢查屬性

來自滾動列表ScrollingList 元件的視頻剪輯MovieClip具有以下檢查屬性。

Visible	如果設置為false則隱藏元件。不隱藏附屬捲軸或其他任何擴展列表項圖形
Disabled	如果設置為true則禁止元件，不禁止附屬捲軸和列表項（包括內部創建和外部渲染）
itemRenderer	列表項渲染器ListItemRenderer符號名稱。用於創建內部列表項實例，如果設置了 <i>rendererInstanceName</i> 屬性則不產生影響
rendererInstanceName	列表項渲染器擴展字首，與滾動列表ScrollingList元件一起使用。場景中的列表項實例必須用該屬性值作為字首。如果屬性設置為‘r’，則所有本元件使用的列表項實例必須具有以下值：‘r1’、‘r2’、‘r3’……第一個項應該為數值1。
Scrollbar	場景中的捲軸ScrollBar元件實例名稱或符號名。如果指定了實例名稱，則滾動列表ScrollingList將作為實例的鉤，如果符號名未指定，將由滾動列表ScrollingList創建一個符號實例。
Margin	內部創建的列表元件和列表項元件的頁面邊緣。如果設置了 <i>rendererInstanceName</i> 屬性該值不產生影響。
rowHeight	內部創建的列表項實例高度，如果設置了 <i>rendererInstanceName</i> 屬性該值不產生影響。
paddingTop	在列表項的頂部進行額外填充。如果設置了 <i>rendererInstanceName</i> 屬性，則這一數值無效。不影響自動生成的捲軸。
paddingBottom	在列表項的底部進行額外填充。如果設置了 <i>rendererInstanceName</i> 屬性，則這一數值無效。不影響自動生成的捲軸。
paddingLeft	在列表項的左側進行額外填充。如果設置了 <i>rendererInstanceName</i> 屬性，則這一數值無效。不影響自動生成的捲軸。
paddingRight	在列表項的右側進行額外填充。如果設置了 <i>rendererInstanceName</i> 屬性，則這一數值不會產生影響。不影響自動生成的捲軸。
thumbOffsetTop	捲軸拖動點頂部偏移。如果列表未能自動創建一個捲軸示例，則該屬性不會產生影響。
thumbOffsetBottom	捲軸拖動點底部偏移。如果列表未能自動創建一個捲軸示例，則該屬性不會產生影響。
thumbSizeFactor	捲軸拖動點的頁面尺寸因素。如果某一指定因素的數值大於1.0，則拖動點的尺寸將會增加。小於1.0的正值將會縮小拖動點的尺寸。如果捲軸未附在列表中，則這一數值不會產生影響。

2.4.4.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

滾動列表ScrollingList元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行中設置為false
focusIn	元件獲得焦點
focusOut	元件失去焦點
Change	選擇序號發生變化。 <ul style="list-style-type: none">• <i>index</i>：新選擇序號，數值類型，值為0到列表項數量最小為1。
itemPress	一個列表項被按下。 <ul style="list-style-type: none">• <i>renderer</i>：按下的列表項，CLIK按鈕類型。• <i>item</i>：與列表項關聯的資料，該值從列表資料源dataProvider重新得到。AS2物件類型。• <i>index</i>：與資料源dataProvider 關聯的列表項序號。數值類型，值為0到列表項數量最小為1。• <i>mouseIndex</i>：用來產生事件的滑鼠游標序號（只引用在多滑鼠游標環境），數值類型，值為0到3。
itemClick	A list item has been clicked. 一個列表項被點擊。 <ul style="list-style-type: none">• <i>renderer</i>：按下的列表項，CLIK按鈕類型。• <i>item</i>：與列表項關聯的資料，該值從列表資料源dataProvider重新得到。AS2物件類型。• <i>index</i>：與資料源dataProvider 關聯的列表項序號。數值類型，值為0到列表項數量最小為1。• <i>mouseIndex</i>：用來產生事件的滑鼠游標序號（只引用在多滑鼠游標環境），數值類型，值為0到3。
itemDoubleClick	一個列表項被雙擊。 <ul style="list-style-type: none">• <i>renderer</i>：按下的列表項，CLIK按鈕類型。• <i>item</i>：與列表項關聯的資料，該值從列表資料源dataProvider重新得到。AS2物件類型。• <i>index</i>：與資料源dataProvider 關聯的列表項序號。數值類型，值為0到列表項數量最小為1。

	<ul style="list-style-type: none"> • <i>mouseIndex</i>：用來產生事件的滑鼠游標序號（只引用在多滑鼠游標環境），數值類型，值為0到3。
itemRollOver	滑鼠游標在列表項上方滾動。 <ul style="list-style-type: none"> • <i>renderer</i>：按下的列表項，CLIK按鈕類型。 • <i>item</i>：與列表項關聯的資料，該值從列表資料源dataProvider重新得到。AS2物件類型。 • <i>index</i>：與資料源dataProvider關聯的列表項序號。數值類型，值為0到列表項數量最小為1。 • <i>mouseIndex</i>：用來產生事件的滑鼠游標序號（只引用在多滑鼠游標環境），數值類型，值為0到3。
itemRollOut	滑鼠游標滾動出列表項。 <ul style="list-style-type: none"> • <i>renderer</i>：按下的列表項，CLIK按鈕類型。 • <i>item</i>：與列表項關聯的資料，該值從列表資料源dataProvider重新得到。AS2物件類型。 • <i>index</i>：與資料源dataProvider關聯的列表項序號。數值類型，值為0到列表項數量最小為1。 • <i>mouseIndex</i>：用來產生事件的滑鼠游標序號（只引用在多滑鼠游標環境），數值類型，值為0到3。

下例顯示了如何監聽列表項的點擊動作：

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

2.4.4.6 提示和技巧

顯示一個來自複雜物件集合的標簽：

```
// ScrollingList將自動使用屬性名稱 'label'
// 如果在專案物件中找到：
list.dataProvider = [{label: "one", data:1}, {label: "two", data:2}];

// 但是如果專案物件具有不同的標簽屬性，如下所示，
// 則列表可以配置成使用該屬性來代替：
list.labelField = "name";
list.dataProvider = [{name: "one", data:1}, {name: "two", data:2}];
```

```

// 如果從專案物件上構建一個標簽邏輯上比較複雜並需要一個函數，  

// 則設置labelFunction屬性  

list.labelFunction = function(itemObj:Object):String {  

    // 邏輯上構建一個標簽  

}  

list.dataProvider = [{p1: "foo", p2: 1}, {p1: "bar", p2: 2}];
```

2.4.5 TileList



圖 41: 無皮膚標題列表TileList.

標題列表TileList

(gfx.controls.TileList)與滾動列表ScrollingList類似，為一個可以滾動其元素的元件。可以自身展示列表項或者使用現存的場景中的列表項。可以附加滾動指示器ScrollIndicator或捲軸ScrollBar元件提供滾動反饋和控制功能。標題列表TileList和滾動列表ScrollingList的區別為標題列表同時支援多個行和列，列表項選擇可以向四個主要方向移動。該元件可與資料源dataProvider一起使用。資料源通過代碼指派，如下面實例所示：

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

默認情況下標題列表TileList使用列表項渲染器ListItemRenderer元件作為內容。因此列表項渲染器必須在FLA文件庫中存在，除非列表項渲染器檢查屬性改變成另外一個元件。見檢查屬性小節獲得更多資訊。

2.4.5.1 用戶交互

點擊一個列表項或者一個附屬的捲軸ScrollBar實例將焦點傳遞到標題列表TileList元件。當獲得焦點，按下向上向下方向鍵或者類似的控制器，如果包含多行逐個單元垂直滾動列表選項。也可以用向左向右方向鍵或類似控制器，如果包含多列逐個單元水平滾動列表選項。如果標題列表TileList包含多行和多列，四個方向鍵或類似控制器可以用來導航列表項。如果沒有選中元素，頂部元素自動作為默認選項。如果游標位於標題列表邊界上滑鼠滾輪可以滾動列表。

按下鍵盤(Page Up)和(Page down)或類似控制器將按頁滾動選項，例如，列表中的可視選項數。按下(Home)和(End)鍵，或者類似控制器，將滾動列表到相應元素的開始和末尾處。與附屬捲軸ScrollBar元件交互將按預期影響標題列表TileList。見捲軸ScrollBar小節瞭解其用戶交互功能。

2.4.5.2 元件設置

標題列表TileList不需要任何子單元，但是，在場景上的標題列表TileList元件上放置一個元件或者調整元件大小時一個視覺化背景就能起到幫助作用。

2.4.5.3 狀態

標題列表TileList元件支援三種狀態，基於焦點和禁止屬性：

- **default** 或enable狀態；
- **focused**狀態，通常突出顯示元件邊框區域；
- **disabled** 狀態。

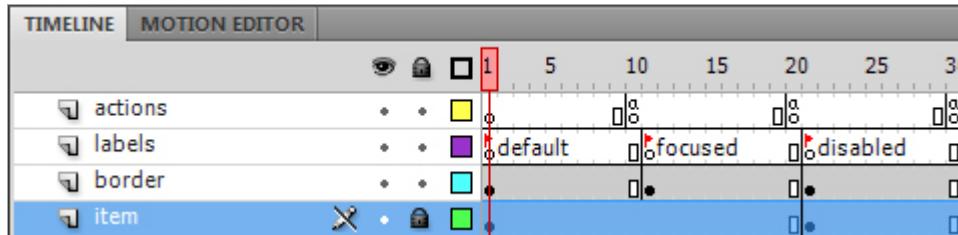


圖 42: 標題列表TileList 時間軸

2.4.5.4 檢查屬性

來自標題列表TileList元件的視頻剪輯MovieClip具有以下檢查屬性：

Visible	如果設置為false則隱藏元件。不隱藏附屬捲軸或其他任何擴展列表專案圖型
Disabled	如果設置為true則禁止元件，不禁止附屬捲軸和列表項（包括內部創建和外部渲染）
itemRenderer	列表項渲染器ListItemRenderer符號名稱。用於創建內部列表項實例，如果設置了rendererInstanceName屬性則不產生影響
renderInstanceName	列表項渲染器擴展字首，與標題列表TileList元件一起使用。場景中的列表項實例必須用該屬性值作為字首。如果屬性設置為‘r’，則所有本元件使用的列表項實例必須具有以下值：‘r1’、‘r2’、‘r3’

第一個項應該為數值1。
scrollbar	場景中的捲軸ScrollBar元件實例名稱或符號名。如果指定了實例名稱，則標題列表TileList將作為實例的鈎，如果符號名未指定，將由標題列表TileList創建一個符號實例。
Margin	內部創建的列表元件和列表項元件的頁面邊緣。如果設置了 <i>rendererInstanceName</i> 屬性該值不產生影響。
rowHeight	內部創建的列表項實例高度，如果設置了 <i>rendererInstanceName</i> 屬性該值不產生影響。
columnWidth	內部創建的列表項實例寬度，如果設置了 <i>rendererInstanceName</i> 屬性該值不產生影響。
externalColumnCount	當設置了 <i>rendererInstanceName</i> 屬性，該值用來指示外部渲染器使用的列數的標題列表TileList
Direction	滾動方向，語義上行和列不根據該值進行改變。

2.4.5.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

標題列表TileList元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行時已設置為false
focusIn	元件獲得焦點
focusOut	元件失去焦點
Change	選擇序號發生變化。 <ul style="list-style-type: none"> • <i>index</i>：新選擇序號，數值類型，值為0到列表項數量最小為1。
itemPress	列表項被按下。 <ul style="list-style-type: none"> • <i>renderer</i>：按下的列表項，CLIK按鈕類型。 • <i>item</i>：與列表項關聯的資料，該值從列表資料源dataProvider重新得到。AS2物件類型。 • <i>index</i>：與資料源dataProvider關聯的列表項序號。數值類型，值為0到列表項數量最小為1。 • <i>mouseIndex</i>：用來產生事件的滑鼠游標序號（只引用在多滑鼠游標環境），數值類型，值為0到3。

itemClick	列表項被點擊。
	<ul style="list-style-type: none"> • <i>renderer</i>：按下的列表項，CLIK按鈕類型。 • <i>item</i>：與列表項關聯的資料，該值從列表資料源dataProvider重新得到。AS2物件類型。 • <i>index</i>：與資料源dataProvider關聯的列表項序號。數值類型，值為0到列表項數量最小為1。 • <i>mouseIndex</i>：用來產生事件的滑鼠游標序號（只引用在多滑鼠游標環境），數值類型，值為0到3。
itemDoubleClick	列表項被雙擊。
	<ul style="list-style-type: none"> • <i>renderer</i>：按下的列表項，CLIK按鈕類型。 • <i>item</i>：與列表項關聯的資料，該值從列表資料源dataProvider重新得到。AS2物件類型。 • <i>index</i>：與資料源dataProvider關聯的列表項序號。數值類型，值為0到列表項數量最小為1。 • <i>mouseIndex</i>：用來產生事件的滑鼠游標序號（只引用在多滑鼠游標環境），數值類型，值為0到3。
itemRollOver	滑鼠游標在列表項上方滾動。
	<ul style="list-style-type: none"> • <i>renderer</i>：按下的列表項，CLIK按鈕類型。 • <i>item</i>：與列表項關聯的資料，該值從列表資料源dataProvider重新得到。AS2物件類型。 • <i>index</i>：與資料源dataProvider關聯的列表項序號。數值類型，值為0到列表項數量最小為1。 • <i>mouseIndex</i>：用來產生事件的滑鼠游標序號（只引用在多滑鼠游標環境），數值類型，值為0到3。
itemRollOut	滑鼠游標移開列表項。
	<ul style="list-style-type: none"> • <i>renderer</i>：按下的列表項，CLIK按鈕類型。 • <i>item</i>：與列表項關聯的資料，該值從列表資料源dataProvider重新得到。AS2物件類型。 • <i>index</i>：與資料源dataProvider關聯的列表項序號。數值類型，值為0到列表項數量最小為1。 • <i>mouseIndex</i>：用來產生事件的滑鼠游標序號（只引用在多滑鼠游標環境），數值類型，值為0到3。

下例顯示如何判斷標題列表TileList接收焦點：

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

```
// 執行操作  
}
```

2.4.5.6 提示和技巧

顯示一個來自複雜物件集合的標簽：

```
// ScrollingList將自動使用屬性名稱 'label'  
//如果在專案物件中找到  
list.dataProvider = [{label: "one", data:1}, {label: "two", data:2}];  
  
// 但是如果專案物件具有不同的標簽屬性，如下所示，  
// 則列表可以配置成使用該屬性來代替：  
list.labelField = "name";  
list.dataProvider = [{name: "one", data:1}, {name: "two", data:2}];  
  
// 如果從專案物件上構建一個標簽邏輯上比較複雜並需要一個函數，  
// 這設置labelFunction屬性  
list.labelFunction = function(itemObj:Object):String {  
    // 邏輯上構建一個標簽Logic to construct a label  
}  
list.dataProvider = [{p1: "foo", p2: 1}, {p1: "bar", p2: 2}];
```

2.4.6 DropdownMenu

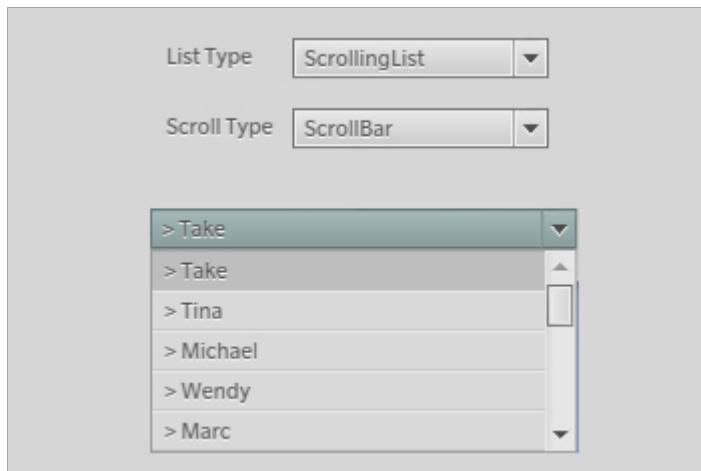


圖 43: 無皮膚下拉功能表DropdownMenu.

下拉功能表DropdownMenu

(gfx.controls.DropdownMenu)包含了按鈕和列表的行為。點擊元件打開列表包含了選擇元素。下拉功能

表DropdownMenu

只在靜止狀態顯示選擇元素。可以配置成使用滾動列表`ScrollingList`或標題列表`TileList`，並可以配上捲軸`ScrollBar`或滾動指示器`ScrollIndicator`。列表與資料源`dataProvider`相關聯，下拉功能表`DropdownMenu`列表元素與資料源`dataProvider`進行連接。資料源`dataProvider`通過代碼指派，如下例所示：

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

默認情況下下拉功能表`DropdownMenu`使用列表項渲染器`ListItemRenderer`元件作為內容。因此下拉功能表`DropdownMenu`和列表項渲染器`ListItemRenderer`必須在FLA文件庫中存在，除非下拉屬性改變成另外一個元件。參考檢查屬性小節獲得更多資訊。

也需注意默認情況下下拉功能表`DropdownMenu`在列表元素中不附帶一個捲軸，捲軸`ScrollBar`或滾動指示器`ScrollIndicator`必須通過代碼附加到下拉功能表`DropdownMenu`列表元素。見提示和技巧小節獲得更多資訊。

2.4.6.1 用戶交互

點擊下拉功能表`DropdownMenu`實例或者按下(`Enter`)鍵或類似控制器將開打可選元素列表。當打開時焦點轉移到該列表，如在用戶交互小節所描述用戶能夠與列表中`ScrollingList`、`TileList`和`ScrollBar`元件進行交互。點擊一個列表項將其選中，關閉列表並在下拉功能表`DropdownMenu`元件中顯示選擇項。在列表邊界外用滑鼠點擊將自動關閉列表，焦點將傳遞回下拉功能表元件。

2.4.6.2 元件設置

下拉功能表`DropdownMenu`繼承了按鈕元件的絕大多數功能。從而視頻剪輯`MovieClip`使用下拉功能表類必須有以下名稱的子單元。列表和捲軸動態創建，注出了對應的可選單元：

- **`textField`**：(可選) `TextField` 類型，按鈕標簽。
- **`focusIndicator`**: (可選)
`MovieClip`類型，一個單獨的視頻剪輯用來顯示焦點狀態，如果被使用，則視頻剪輯必須擁有兩個幀名為：“show” 和 “hide” 。

2.4.6.3 狀態

下拉功能表`DropdownMenu`開打時為選中狀態，因此需要與`ToggleButton`和`CheckBox`具有相同的狀態來指示選中狀態，這些狀態包括：

- **`up`** 或默認狀態：
- 當滑鼠箭頭在元件上方或者獲得焦點時為**`over`** 狀態；
- 當按鈕被點擊時候為**`down`** 狀態；

- **disabled** 狀態；
- **selected_up** 或者默認狀態；
- 當滑鼠箭頭位於元件上方或獲得焦點時為**selected_over** 狀態；
- 當按鈕被按下時為**selected_down** 狀態；
- **selected_disabled** 狀態；

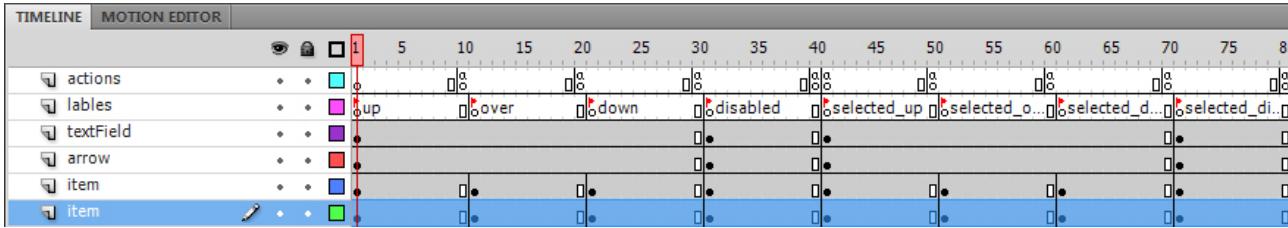


圖 44: 下拉功能表DropdownMenu 時間軸

這裏為選擇按鈕下拉功能表DropdownMenu所需要的最少關鍵幀設置。按鈕Button元件支援狀態和關鍵幀的擴展設置，與DropdownMenu元件相同，在[CLIK 按鈕入門](#)文檔中有詳細描述。

2.4.6.4 檢查屬性

下拉功能表DropdownMenu元件的檢查屬性為：

Visible	如果設置為false則隱藏元件
Disabled	如果設置為false則禁止元件
Dropdown	列表元件 (ScrollingList 或 TileList) 的符號名稱，與下拉功能表DropdownMenu元件一起使用
dropdownWidth	下拉清單寬度，如果該值為-1，則下拉功能表將依據元件寬度決定列表寬度
itemRenderer	下拉清單的專案渲染符號名稱，由下拉清單實例創建
scrollbar	下拉清單捲軸的符號名，由下拉清單實例創建。如果值為空，則下拉清單無捲軸
margin	列表部件與內部創建的列表項之間的邊距。該邊距還會對自動生成的捲軸產生影響。
paddingTop	在列表項的頂部進行額外填充。不影響自動生成的捲軸。
paddingBottom	在列表項的底部進行額外填充。不影響自動生成的捲軸。
paddingLeft	在列表項的左側進行額外填充。不影響自動生成的捲軸。
paddingRight	在列表項的右側進行額外填充。不影響自動生成的捲軸。
thumbOffsetTop	捲軸拖動點頂部偏移。如果沒有自動生成捲軸示例，則這一屬性不會產生影響。
thumbOffsetBottom	捲軸拖動點底部偏移。如果沒有自動生成捲軸示例，則這一屬性不會產生影響。

thumbSizeFactor	捲軸thumb頁面尺寸因素。如果某一指定因素的數值大於1.0，則thumb的尺寸將會增加。小於1.0的正值將會縮小thumb的尺寸。如果捲軸未附在列表中，則這一數值不會產生影響。
offsetX	下拉清單從下拉按鈕位置進行水平偏移。正值將列表移動到下拉按鈕水平位置右側。
offset	下拉清單從下拉按鈕發生的垂直偏移。正值將列表從按鈕中移除。
Extent	用於連接 offsetX 的偏移寬度。如果 dropdownWidth 屬性的設置值不等於-1，則該數值不會產生影響。
Direction	列表的開啓方向。有效值為“上”和“下”。

2.4.6.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

DropdownMenu元件產生的事件列表如下所示。除**change**事件意外，與Button元件類似，事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行時已設置為false
focusIn	元件獲得焦點
focusOut	元件失去焦點
Change	選擇序號發生變化。 <ul style="list-style-type: none"> • <i>index</i>：新選擇序號，數值類型，值為0到列表項數量最小為1。 • <i>data</i>：選擇序號中與列表項關聯的資料，AS2物件類型
Select	所選屬性已改變。 <ul style="list-style-type: none"> • <i>Selected</i>：按鈕的選擇狀態，true為被選擇，為布林類型。
stateChange	按鈕狀態已改變 State：按鈕新狀態。String類型。值為“up”、“over”、“down”等。 參考 CLIK按鈕入門 文檔獲取詳細的狀態列表資訊。
rollover	滑鼠箭頭在按鈕上方滾動。 <ul style="list-style-type: none"> • <i>mouseIndex</i>：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠游標環境）。數值類型。值為0-3。
rollout	滑鼠箭頭從按鈕移開。 <ul style="list-style-type: none"> • <i>mouseIndex</i>：用戶產生事件的滑鼠箭頭索引（只應用在多滑鼠箭頭環境）數值類型。值為0-3。

Press	按鈕被點擊。
	<ul style="list-style-type: none"> • <i>mouseIndex</i>：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠游標環境）。數值類型。值為0-3。
doubleClick	按鈕被雙擊。只在 <i>doubleClickEnabled</i> 屬性為true時被觸發。
	<ul style="list-style-type: none"> • <i>mouseIndex</i>：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠游標環境）。數值類型。值為0-3。
Click	按鈕被點擊。
	<ul style="list-style-type: none"> • <i>mouseIndex</i>：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠游標環境）。數值類型。值為0-3。
dragOver	滑鼠箭頭拖動到按鈕上方（滑鼠左鍵被按下）
	<ul style="list-style-type: none"> • <i>mouseIndex</i>：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠游標環境）。數值類型。值為0-3。
dragOut	滑鼠箭頭從按鈕拖開（滑鼠左鍵被按下）。
	<ul style="list-style-type: none"> • <i>mouseIndex</i>：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠游標環境）。數值類型。值為0-3。
releaseOutside	滑鼠箭頭從按鈕拖開滑鼠左鍵鬆開。
	<ul style="list-style-type: none"> • <i>mouseIndex</i>：用來產生事件的滑鼠箭頭索引（只應用在多滑鼠箭頭環境）。數值類型。值為0-3。

2.4.6.6 提示和技巧

判斷下拉功能表開打和關閉：

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

在運行時動態創建下拉功能表DropdownMenu：

```
//確保代碼中具有鏈結;
//意味著設計的符號/元件存在與FLA庫中。
//例如：下列中代碼需要用到DropdownMenu、ScrollingList
//和 ScrollBar元件。
attachMovie("DropdownMenu", "dd", this.getNextHighestDepth(),
            {dropdown: "ScrollingList"});
dddataProvider = ["one", "two", "three", "four", "five", "six"];
```

```
// 安裝一個捲軸或滾動指示器到下拉清單比較複雜，  
//因為需要一個延時設置下面代碼中有所描述。  
//使下拉實例在附加捲軸到列表前先顯示列表，這裏就需要一個延時：  
onEnterFrame = function() {  
    dd.dropdown.scrollBar = "ScrollBar";  
    onEnterFrame = null;  
}
```

顯示一個複雜物件集中的標簽：

```
// ScrollingList將自動使用屬性名稱 'label'  
// 如果在專案物件中找到：  
list.dataProvider = [{label: "one", data:1}, {label: "two", data:2}];  
  
// 但是如果專案物件具有不同的標簽屬性，如下所示，  
// 則列表可以配置成使用該屬性來代替：  
  
list.labelField = "name";  
list.dataProvider = [{name: "one", data:1}, {name: "two", data:2}];  
// 如果從專案物件上構建一個標簽邏輯上比較複雜並需要一個函數，  
//則設置labelFunction屬性  
list.labelFunction = function(itemObj:Object):String {  
    // 邏輯上構建一個標簽  
}  
list.dataProvider = [{p1: "foo", p2: 1}, {p1: "bar", p2: 2}];
```

2.5 進度類型

進度類型用來顯示狀態或事件或動作的進度。The Scaleform

CLIK框架包含了兩個元件屬於此分類，狀態指示器StatusIndicator

和進度條ProgressBar。狀態指示器StatusIndicator元件用來顯示事件或動作的狀態。而進度條元件ProgressBar與狀態指示器擁有相同的語義，但是包含了一些附加功能，可以監聽其他產生進度事件的元件或動作。

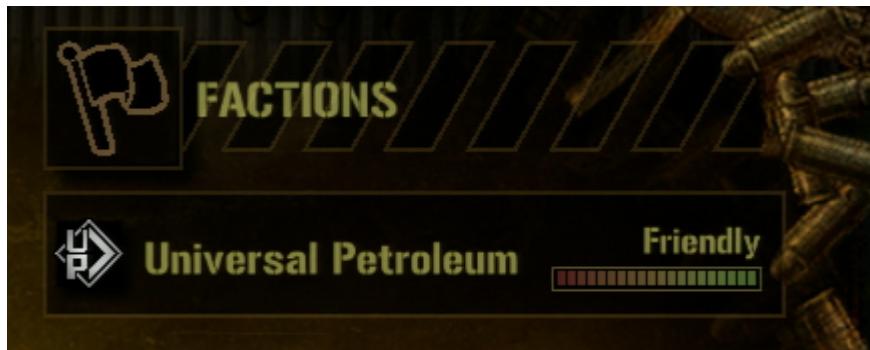


圖 45:來自*Mercenaries 2*的Fraction狀態指示器實例

2.5.1 StatusIndicator

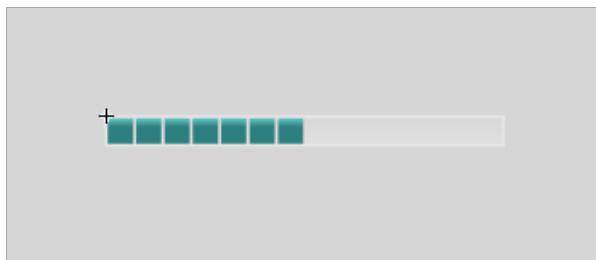


圖 46: 無皮膚狀態指示器StatusIndicator.

狀態指示器StatusIndicator component

(gfx.controls.StatusIndicator)顯示事件或動作狀態，使用時間軸作為視覺指示器。狀態指示器的值將為一個最大和最小值之間的值用來產生一個幀的序號，以在元件時間軸上播放。由於元件時間軸用來顯示狀態，為創建創新的視覺指示器提供了絕對自由的發揮空間。

2.5.1.1 用戶交互

狀態指示器StatusIndicator無用戶交互。

2.5.1.2 元件設置

使用CLIK

狀態指示器StatusIndicator的視頻剪輯MovieClip不需要任何子單元。但是狀態指示器需要至少兩個幀以正確操作。確保在第一幀插入stop()命令避免播放該幀。狀態指示器元件在值屬性產生的對應幀執行gotoAndStop()命令。

2.5.1.3 狀態

狀態指示器StatusIndicator元件無狀態，元件幀用來顯示事件或動作

2.5.1.4 檢查屬性

來自狀態指示器StatusIndicator元件的視頻剪輯MovieClip具有以下檢查屬性：

visible	如果設置為false則隱藏元件
disabled	如果設置為false則禁止元件
Value	一個事件或動作的狀態值，為一個最大值到最小值之間的播放的幀序號。
minimum	插入目標幀的最小值
maximum	插入目標幀的最大值

2.5.1.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

StatusIndicator元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行時已設置為false

2.5.2 ProgressBar

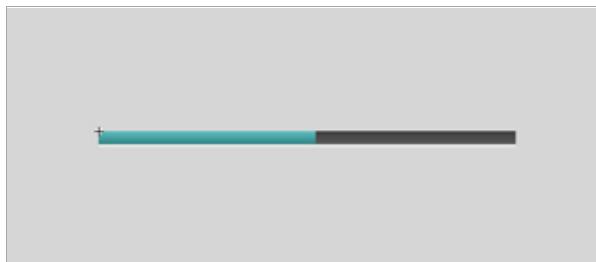


圖 47: 無皮膚進度條 ProgressBar.

進度條ProgressBar

(gfx.controls.ProgressBar)與狀態指示器StatusIndicator類似，也用來通過時間軸顯示事件或動作狀態，但是，與其他元件或產生的進度事件協作。正確指派目標並設置其模式，進度條元件將根據目標導入值（bytesLoaded 和 bytesTotal）自動改變視覺狀態。

2.5.2.1 用戶交互

進度條ProgressBar無任何用戶交互。

2.5.2.2 元件設置

與狀態指示器StatusIndicator類似，使用CLIK進度條StatusIndicator類的視頻剪輯MovieClip不需要任何子單元。但是進度條ProgressBar需要至少兩個幀以正確操作。確保在第一幀插入stop()命令避免播放該幀。進度條ProgressBar將在值屬性產生的相關幀上執行gotoAndStop()。

2.5.2.3 狀態

進度條ProgressBar元件無狀態，元件幀用來顯示事件或動作狀態。

2.5.2.4 檢查屬性

來自進度條ProgressBar元件的視頻剪輯MovieClip具有以下檢查屬性：

visible	如果設置為false則隱藏元件
disabled	如果設置為false則禁止元件
target	進度條ProgressBar將進行“監聽”的目標物件，判斷bytesLoaded 和 bytesTotal值
mode	properties.進度條ProgressBar監聽模式。在“manual”模式，進度條值必須使用setProgress方法進行設置，在“

“polled” 模式，目標必須為 bytesLoaded 和 bytesTotal 屬性，而在 “event” 模式，目標必須分派 “progress” 事件，包括 bytesLoaded 和 bytesTotal 屬性。

2.5.2.5 事件

所有的事件調用都接收一個 Object 參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

ProgressBar 元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為 true
Hide	元件可視屬性在運行時已設置為 false
progress	當進度條 ProgressBar 值改變時產生
complete	當進度條值達到最大值時產生

下例中顯示了進度條事件的監聽：

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

2.6 其他類型

Scaleform

CLIK框架頁包括若干個元件且不能簡單區分，但是為UI開發提供了寶貴的功能。它們為Dialog、UILoader和ViewStack元件。對話方塊Dialog元件可以為模式和非模式對話方塊，UI導入器UILoader提供一個方便的介面以導入內容，視圖堆疊ViewStack能用來管理表單，

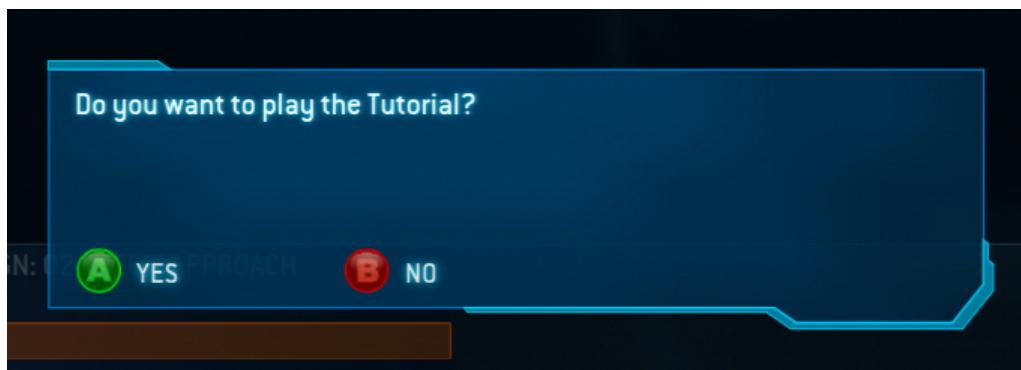


圖 48:來自 *Halo Wars*的對話方塊 Dialog 實例

2.6.1 Dialog

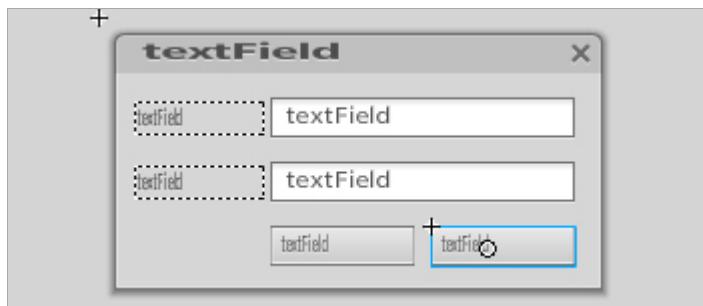


圖 49:無皮膚對話方塊Dialog實例

CLIK對話方塊 Dialog元件

(gfx.controls.Dialog)顯示一個對話方塊視圖，如一個警告對話方塊，在應用程式的頂部。提供示例的靜態介面，通過對話方塊顯示和隱藏任何視頻剪輯MovieClip，也為一個基本類可以作為實際對話方塊視頻剪輯MovieClip使用（擴展）。為確保一次隻打開一個對話方塊，新的Dialog.show()調用將關閉當前開打的對話方塊。

注意內置元件沒有任何內容，因為設計成完全由用戶定義。但是，通過簡單的編輯元件符號可以添加內容。

2.6.1.1 用戶交互

對話方塊Dialog的用戶交互在創建的對話方塊視圖裏定義。

2.6.1.2 元件設置

使用CLIK Dialog類的MovieClip必須具備下面所列的子單元。也列出了可選元素：

- **closeBtn**：(可選) CLIK按鈕 Button類型，與視窗關閉按鈕類似。
- **cancelBtn**：(可選) CLIK按鈕 Button類型，取消cancel按鈕用來關閉對話方塊無確認。
- **submitBtn**：(可選) CLIK按鈕 Button類型，OK按鈕在確認後關閉對話方塊。
- **dragBar**：(可選)視頻剪輯MovieClip 類型，對話方塊的拖動標題欄。

2.6.1.3 狀態

對話方塊Dialog元件無狀態。視頻剪輯MovieClip作為對話方塊視圖顯示，或許有或許沒有自身狀態。

2.6.1.4 檢查屬性

來自對話方塊Dialog元件的視頻剪輯MovieClip具有以下檢查屬性：

Visible	如果設置為false則隱藏元件
Disabled	如果設置為false則禁止元件

2.6.1.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

Dialog元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行時已設置為false
submit	對話方塊Dialog確認按鈕被點擊。

- **data**：對話方塊Dialog提交資料，對話方塊元件的getSubmitData方法調用

於此方法，應該被自定義對話方塊覆蓋。默認返回值為true（布林型）。getSubmitData的返回值為AS2物件。

Close 點擊了Dialog的閉關按鈕

以下例子顯示如何處理對話方塊提交事件：

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

2.6.2 UILoader

CLIK UILoader

(gfx.controls.UILoader)通過唯一路徑導入一個擴展的SWF/GFX或者圖像。UILoaders支援導入資源自動縮放以適合邊框。如果Scaleform和平臺支援線程運行則非同步導入資源。

2.6.2.1 用戶交互

UILoader元件無用戶交互，如果一個SWF/GFX文件導入到UILoader，則可能有自身用戶交互。

2.6.2.2 元件設置

使用CLIK UILoader類的MovieClip必須具備下面所列的子單元。也列出了可選元素：

- **bg**：(可選)視頻剪輯 MovieClip
類型，UILoader背景，無函數支援，不同於場景中的上級元件的可是外觀，背景可以在運行時取消。

2.6.2.3 狀態

UILoader元件無狀態，如果SWF/GFX文件導入到UILoader，可能有自身狀態。

2.6.2.4 檢查屬性

來自UI導入器UILoader元件的視頻剪輯MovieClip具有以下檢查屬性：

Visible	如果設置為false則隱藏元件
autoSize	如果設置為true，導入內容自動縮放以適合UILoader邊框
maintainAspectRatio	如果為true，導入內容將在UILoader邊距內縱橫比進行合適縮放。如果為false，內容伸展鋪滿UILoader邊框。
Source	導入的SWF/GFX或圖像檔案名

Timeout	導入等待超時時間為毫秒級，如果達到了超時時間內容還未導入完畢，UILoader將產生一個‘ioError’事件
----------------	---

2.6.2.5 事件

所有的事件調用都接收一個Object參數，包含事件相關資訊。以下為通用事件的屬性。

- **type:** 事件類型。
- **target:** 事件產生的目標。

UILoader元件產生的事件列表如下所示。事件旁列出的屬性為通用屬性的補充。

Show	元件可視屬性在運行時已設置為true
Hide	元件可視屬性在運行時已設置為false
progress	導入過程不考慮是否可以被導入，事件在以下情況下觸發a) 內容導入或者b) 導入超時
loaded	導入資料比例，屬性值在0和100之間
complete	內容導入完畢
ioError	指定內容不能被導入

下例展示了如何通知導入錯誤：

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

2.6.3 ViewStack

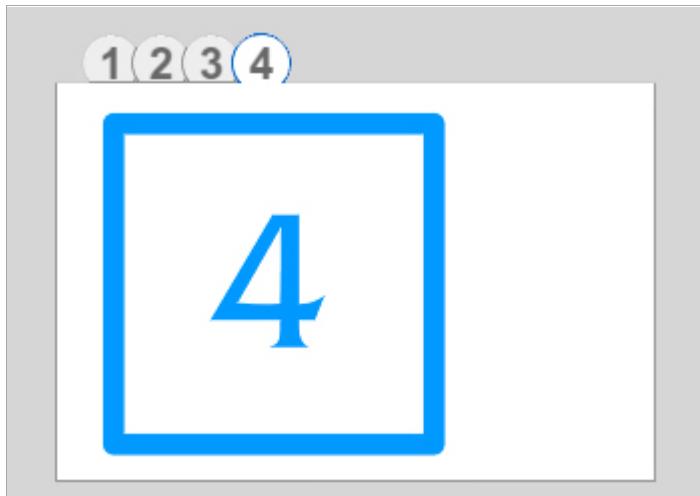


圖 50: 無皮膚視圖堆疊ViewStack

CLIK視圖堆疊ViewStack

(gfx.controls.ViewStack)顯示一個視圖，來自一個導入到緩存中的集。該元件可以用於多視圖元件如Tab Box或

Accordion（這些例子的樣本都可以在演示文件夾裏找到）。一個視圖堆疊ViewStack能夠指向另外一個元件如選擇按鈕RadioButton組以當元件改變時自動改變視圖。

2.6.3.1 用戶交互

視圖堆疊ViewStack元件無用戶交互，通過ViewStack導入的視圖並顯示可能有自身用戶交互。

2.6.3.2 元件設置

視頻剪輯MovieClip使用CLIK視圖堆疊ViewStack類不需要任何命名子單元。但必須根據導入內容縮放尺寸。

2.6.3.3 狀態

視圖堆疊ViewStack元件無狀態，由ViewStack導入並顯示的視圖可能有自身狀態。

2.6.3.4 檢查屬性

來自視圖堆疊ViewStack元件的視頻剪輯MovieClip具有以下檢查屬性：

Visible	如果設置為false則隱藏元件
Cache	如果為true，導入視圖將存放在緩存中，這樣節省創建視圖的處理時間，但是需要一個固定的ViewStack目標組targetGroup（見下面內容）。

targetGroup 有效組物件名稱，如ButtonGroup，產生 ‘change’ 事件，組物件中的當前元素必須具有一個資料屬性包含一個導入並顯示的視圖的鏈結ID。例如，選擇按鈕RadioButtons，擁有一個資料屬性可以通過Flash IDE元件檢查其指派一個鏈結ID。

2.6.3.5 事件

視圖堆疊ViewStack不產生任何事件。

3 藝術細節

本章將幫助美工設計師開發Scaleform

CLIK元件的皮膚，包括為小的元件實例繪製皮膚的詳細過程和最優方法，以及動畫和內置字體。

3.1 最優方法

本章包括了為Scaleform Clik元件創建皮膚的最優方法。

3.1.1 圖元圖像

當開發基於CLIK元件繪製向量皮膚，建議所有資源都用圖元圖像。一個完美的圖元資源在Flash柵格化中完美排列。

為使得能夠改變柵格：

1. 從Flash頂部功能表選擇視圖；
2. 選擇柵格，點擊使柵格顯示；
3. 從Flash頂部功能表再次選擇視圖；
4. 選擇柵格然後點擊編輯柵格；
5. 在垂直和水平位置輸入1px；
6. 點擊OK。

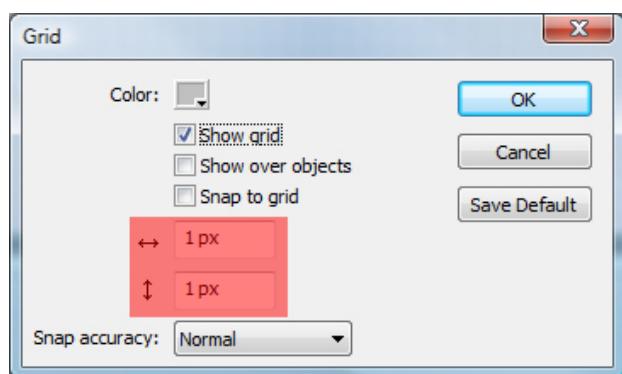


圖 51: 編輯柵格 Grid 視窗

現在應該可以看到覆蓋在場景上的一個柵格。每個柵格表示一個圖元。確保當創建美術資源時對齊柵格。這回確保最終的SWF中圖像不會模糊。注意：保持所有圖像尺寸為2的指數倍；否則可能會導致模糊，見下面的2的指數倍標題。

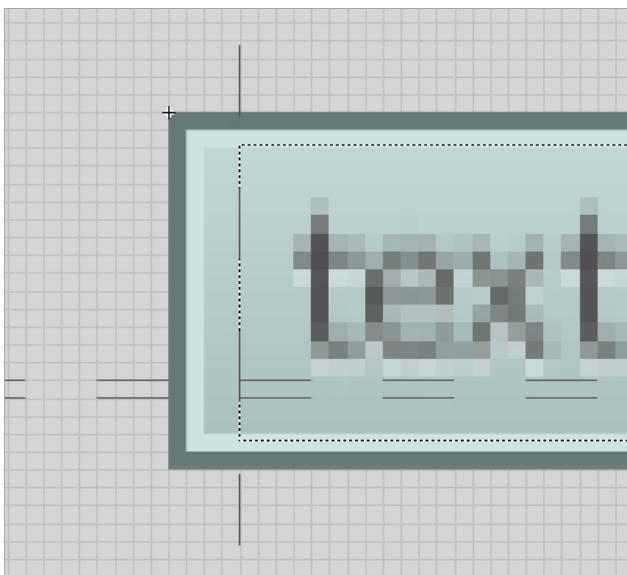


圖 52: 一個圖元的向量圖像

3.1.2 蒙版

在Flash中蒙板可以使設計師在運行時隱藏部分圖像。蒙板經常在動畫效果中使用，但是，蒙板消耗大量資源。Scaleform建議儘量避免蒙板的使用。如果蒙板必須使用，保持在個位數之內並測試添加蒙板前後動畫性能。

在Flash中使用蒙板的一個可能選擇為在Photoshop®中創建PNG，用透明混合處理需要遮蓋的區域。但是，這只能用於非動畫圖像的透明區域。

3.1.3 動畫

最好避免動畫中向量圖形的轉換，如將一個正方形轉換成圓形。這些動畫類型開銷非常大，因為這些形狀在每一個幀都需要重新進行評估。

如果可以避免在向量圖形中縮放動畫，額外的柵格化影響記憶體和性能 –

將向量圖形轉換為三角元素圖像的過程。開銷最小的動畫為使用平移和旋轉，不需要額外的柵格化。

避免使用可編程映射動畫，選擇以映射動畫為基礎的時間軸選擇，因為時間軸映射動畫在性能上更加有優勢。使用時間軸映射動畫，保持在既定幀速率下實現一個平滑的動畫所需要的最少幀數量。

3.1.4 圖層和繪製元素

在Flash中創建皮膚使用盡可能少的圖層。每使用一個圖層至少增加一個繪製元素。繪製元素使用越多，記憶體需求就越大，性能也將受到影響。

3.1.5 複雜皮膚介面

在複雜皮膚中最好使用點陣圖；但是，在簡單皮膚中使用向量圖可以節省更多記憶體並可以在任何解析度下縮放而不失真（模糊）。

3.1.6 2的指數倍

確保點陣圖在尺寸上為2的指數倍，2的指數倍點陣圖尺寸如下所示：

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

3.2 已知問題和推薦工作流程

本節包括了已知藝術相關Flash問題列表和在Scaleform CLIK下的推薦工作流程。

3.2.1 複製元件

在Flash中複製元件存在幾個問題，複製元件導致原來元件的鏈結資訊和元件定義資訊不拷貝到新的元件中去。同樣的，有些元件沒有這些鏈結資訊將無法工作。本節描述了兩種方法來解決這個問題。

3.2.1.1 複製元件（方法1）

從外部FLA文件複製一個無更改（無皮膚）CLIK元件，如按鈕，到目標FLA文件最快的方如下：

1. 打開CLIK_Components.fla文件。
2. 從文件的庫中拷貝元件（例如，按鈕）到目標FLA文件，在原始檔案庫中點擊並選擇Copy，然後點擊目標FLA中的庫並選擇Paste。
3. 點擊目標FLA庫中的元件並選擇Rename對其重新命名，不要與‘Button’重名。
4. 再次點擊目標FLA庫中的元件，選擇Properties。
5. 改變Identifier區域匹配步驟3中選擇元件的新名稱。

6. 點擊庫視窗中的空白區域並選擇*Paste*，一個新的原始元件拷貝將粘貼進來包括所有的完整的鏈結資訊。

3.2.1.2 複製元件（方法2）

當在相同的庫中複製符號（元件），鏈結資訊將不會被拷貝到新的複製符號中去。元件功能執行需要這些資訊，實現的方法為：

1. 點擊*library*面板中需要複製的元件並選擇*Properties*。
2. 在*Properties*視窗，雙擊文本（例如`gfx.controlsButton`）突出顯示*Class*文本區域並按下(**CTRL+C**)鍵進行拷貝。
3. 按下*Cancel*。
4. 再次點擊需要拷貝的元件，並選擇*Duplicate*。
5. 在*Duplicate Symbol* 視窗中，點擊*Export*使*ActionScript*核取方塊使能。
6. 點擊*Class*區域並按下(**CTRL+V**)來粘貼鏈結資訊到該文本區域*textField*。
7. 按下OK。
8. 點擊*library*面板中新拷貝的元件並選擇*Component Definition*。
9. 點擊*Class textField*空白區域並按下(**CTRL+V**)來粘貼鏈結資訊到文本區域*textField*。
10. 按下OK。

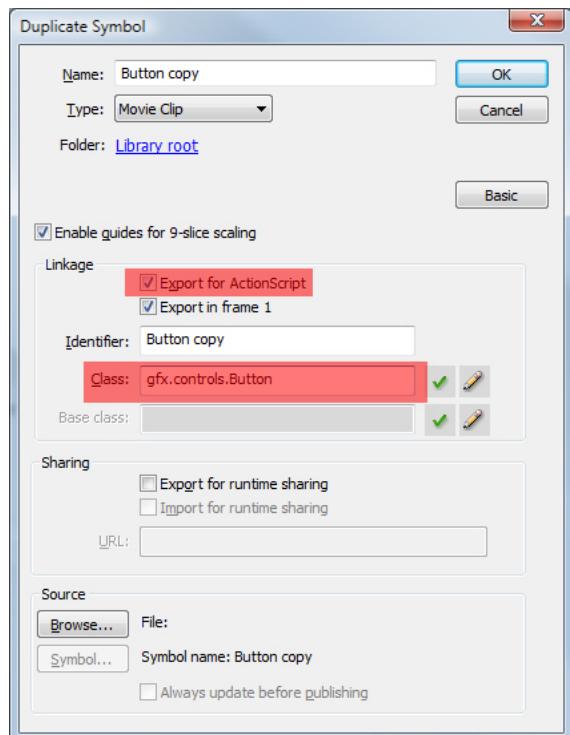


圖 53: 複製符號鏈結（必須填充紅色顯亮區域）

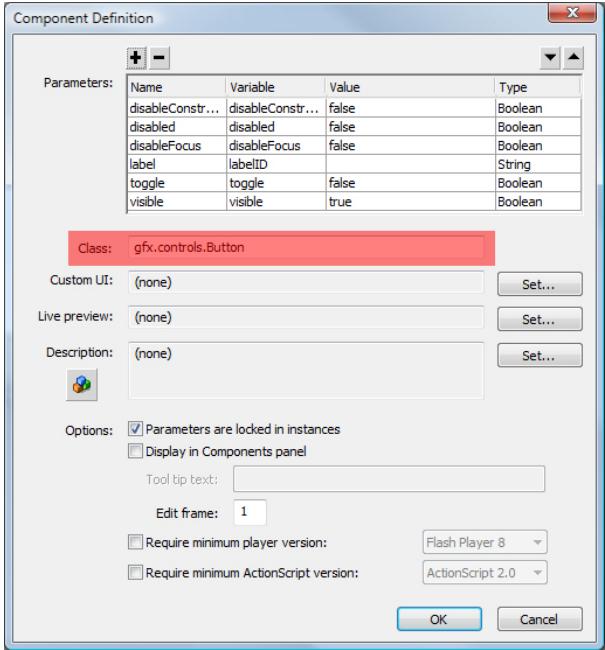


圖 54: 元件定義（必須填入類）

3.3 皮膚繪製實例

Scaleform

CLIK的主要優勢為視覺和函數層的分離。分離能夠使大部分部件中編程人員和藝術設計師可以各自獨立工作。輕鬆自定義外觀和風格為這種分離的主要優勢之一。

儘管內置CLIK元件具有一個標準的外觀和風格，但有時候需要自定義以滿足客戶自身需求。以下部分即描述了自定義內置元件的方法。

CLIK元件設置皮膚與任何Flash標準符號相同，雙擊FLA文件庫中的一個元件獲得元件任意一個時間軸：

- 在Flash中修改默認皮膚；
- 在Flash中從頭開始創建自定義皮膚；或者
- 導入Photoshop或Illustrator®中創建的藝術資源到Flash。

請瀏覽文檔[CLIK入門](#)獲得關於皮膚繪製指南的詳細資訊。

3.3.1 StatusIndicator皮膚繪製

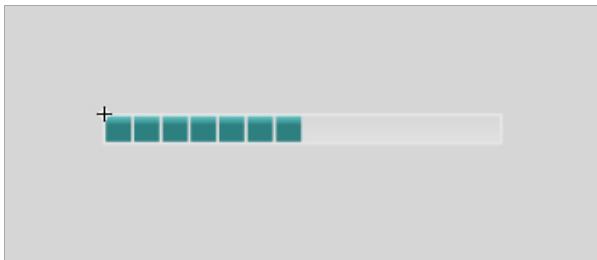


圖 55: 無皮膚StatusIndicator。

狀態指示器StatusIndicator為一個唯一的元件在繪製皮膚時與其他大多數基於按鈕的元件需要略微不同的方法，打開狀態指示器元件，記錄時間軸，具有兩個圖層：*indicator*和*track*。*Track*用來顯示背景圖像；無其他功能。*Indicator*圖層使用若干關鍵幀包括點陣圖或向量圖來從最低到最高的逐步遞增狀態。

指南中使用Photoshop文件中創建的若干個圖層的點陣圖來繪製狀態指示器。這為狀態指示器皮膚繪製的方法之一，但是，還有其他的方法可以在場景中創建和裝配圖形。最終的結果應該都相同，能夠使狀態指示器正確工作。



圖 56: 狀態指示器StatusIndicator PSD 文件

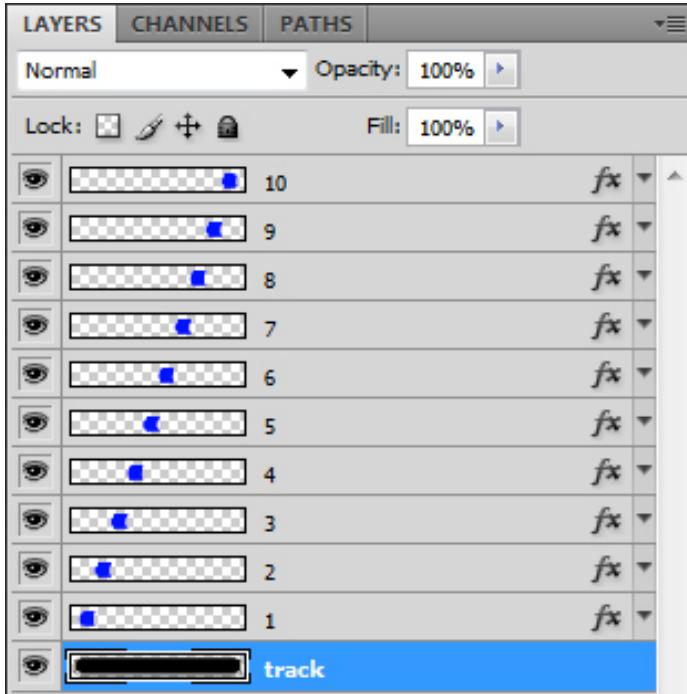


圖 57：在Photoshop中為StatusIndicator PSD 文件設置的圖層

1. 在Photoshop中創建狀態指示器StatusIndicator點陣圖與上面描述類似。注意圖層順序和編號以及在圖層中每個圖像的位置。確保背景透明，為本使用手冊創建一個類似文件；
2. 保存文件為PSD格式；
3. 在Flash中，從功能表選擇File，然後選擇Import -> Import to Stage；
4. 瀏覽並選擇狀態指示器StatusIndicator皮膚PSD文件；
5. 在Import窗口，確保Convert layers to:下拉功能表設置為‘Layers’；
6. 按下OK；
7. 一個新的Flash時間軸圖層應該在PSD文件中的每個層中創建，在上面圖像例子中，需要創建10個圖層，因為PSD文件內部擁有10個圖層（每個渲染為一個層）。每個圖層標簽分別從1到10，1為最底部圖層10為最頂部圖層，選擇layer 1；
8. 在場景中的點陣圖圖像周圍繪製一個核取方塊；
9. 將圖像移動到老的無皮膚軌道位置，根據要求進行縮放；
10. 在layer 1上選擇第一個關鍵幀並拖動到第6幀；
11. 點擊幀選擇Insert關鍵幀添加新的幀到layer 1圖層的第11幀；
12. 在layer 2圖層選擇點陣圖並按下(Ctrl+X)進行剪切；
13. 選擇layer 1圖層中第11幀位置為新關鍵幀並按下(Ctrl+Shift+V)放置點陣圖到layer 1圖層；
14. 重複此過程直到10個點陣圖都在同一個圖層(layer
 - 1)，位於正確的關鍵幀。每個並排的點陣圖應該拷貝到最後關鍵幀第5幀後的一個關鍵幀中。圖層layer 2中的點陣圖應該被拷貝到關鍵幀11；圖層layer 3中的點陣圖應該被拷貝到第16幀；圖層layer 4中的點陣圖應該被拷貝到第21幀，等等。使用10個PSD圖像，最後關鍵幀應該位於第51幀；
15. 刪除空圖層（2-10）進行清理；

16. 如果在最後的點陣圖關鍵幀之後時間軸上有附加的幀，加上另外五個關鍵幀，然後選擇剩餘幀並刪除，刪除操作為首先選擇然後右鍵點擊並選擇Remove Frames即可。在本使用手冊中，最後幀應該位於第55幀；
17. 選擇原來的*indicator layer*並刪除；
18. 選擇原來的*track* 圖層並刪除，確保不刪除新導入的*track*圖層；
19. 選擇*layer 1*並重命名為 ‘*indicator*’ ；
20. 拖動*indicator*圖層和*track*圖層到*actions layer*圖層下方，確保*track*圖層在*indicator*圖層下面；

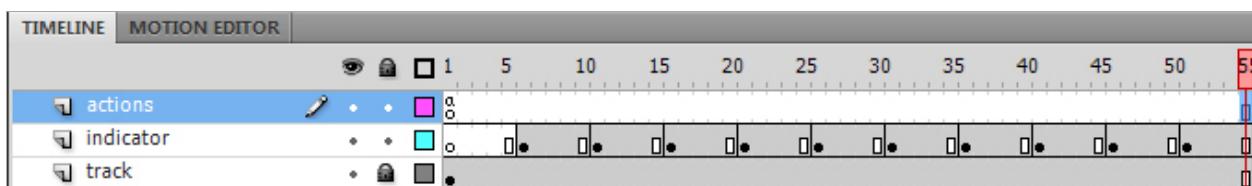


圖 58: 最終時間軸（注意關鍵幀位置和最終幀位置）

21. 推出狀態指示器StatusIndicator時間軸；
22. 設置檢查參數值為1到10之間的任何值；
23. 保存文件；
24. 發佈文件查看新繪製皮膚的指示器。

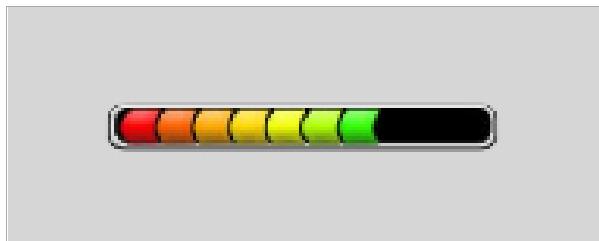


圖 59: StatusIndicator皮膚繪製

3.4 字體和本地化

本節詳細描述了在Scaleform CLIK元件中字體使用。

3.4.1 概要

為了在Scaleform

中字體能被正確渲染，所需字形（字元）必須內置到SWF或使用Scaleform本地化系統進行設置。以下為探究在Flash UI介面中管理字體的方法。

3.4.2 內置字體

和Flash播放器不同，Scaleform

需要內置字體以便顯示。Scaleform播放器在未嵌入字體情況下將顯示空的進行矩形，甚至這些字體在系統中已存在也是如此。這個效果的優勢為方便判斷在Scaleform

中字體是否正確嵌入。在內置元件設置中，在每個文本區域textFidld實例中都嵌入了Slate Mobile。注意Slate

Mobile不包含任何中文、日文或韓文（CJK）字元或字形，內置元件只包含了ASCII字形。這可以通過將Slate Mobile替換為包含CJK字形並設置適當的嵌入選項進行改變。

注意直接內置字體到文本區域textFields與Scaleform本地化系統不相容（在3.4.4小節有所描述）。Scaleform實際上推薦使用Scaleform本地化系統來設置字體，比直接嵌入具有很多優勢。但是在某些情況下，如不需要本地化，內置字體最好為可選項。與UI介面管理類似，字體管理也需要幾點考慮如本地化和記憶體管理。

3.4.3 在textField嵌入字體

只需在動態或輸入文本區域textFields中嵌入字體，靜態文本在編譯時自動轉換為字形輪廓（原始向量圖形）。在動態文本區域textField中嵌入字體，在場景中選擇動態文本區域的屬性檢查(Window > Property Inspector)框中點擊Embed按鈕。選擇應該嵌入的字元或字元集並選擇OK。一旦完成了這些步驟，在你的FLA文件中就可以使用該字體。在相同的FLA文件的多行文本區域中如果需要使用相同的字體，這些步驟只需要執行一次。同樣，多次嵌入相同字體不會增加記憶體使用量。注意字元集包含了大量的字形如中文在導入時佔用大量的記憶體。

3.4.4 本地化系統

Scaleform本地化系統使用熱交換機制無論在當前位置是否發生變化均可導入和導出字體庫。這些字體庫本身為SWF文件包括內置字體字形可以被SWF文本使用。Scaleform能夠使用來自字體庫的字形，為根據需求動態改變字體提供了強大的方法。

由於Scaleform本地化系統在文本區域textField直接嵌入字形時不能交換字體，文本區域必須使用一個導入的字形符號。通常字體符號在一個名為gxfontlib.fla的文件中創建並導入到swf文本中。這個導入的字體設置到所有支援字體交換的文本區域中。Scaleform當前可以截留該字體符號鏈結並基於當前位置導入不同的字體。

字體庫不需要導出任何字體，只需要插入適當的字體（見前面小節的如何插入字體）。Scaleform本地化系統使用fontconfig.txt

文件來定義每個位置的字體庫，以及文本區域使用的字體符號之間的字體映射關係和插入到字體庫中的實體字體。

fontconfig.txt文件也包含轉換對應關係。**Scaleform**本地化系統可以自行文本的快速轉換，這些文本顯示在每個場景中的動態或輸入文本區域。例如，如果一個文本區域包含文本‘\$TITLE’，轉換映射具有一個對照表如\$TITLE=Scaleform，然後文本區域將自動顯示‘Scaleform’來代替。

本章中描述的資訊作為**Scaleform**字體和本地化系統的一個概括。為深入理解**Scaleform**中字體和本地化，請參考文檔[字體概述](#)。

4 編程詳述

本章描述了子系統框架和亮點的具體細節，提供了Scaleform CLIK元件架構的上層理解。

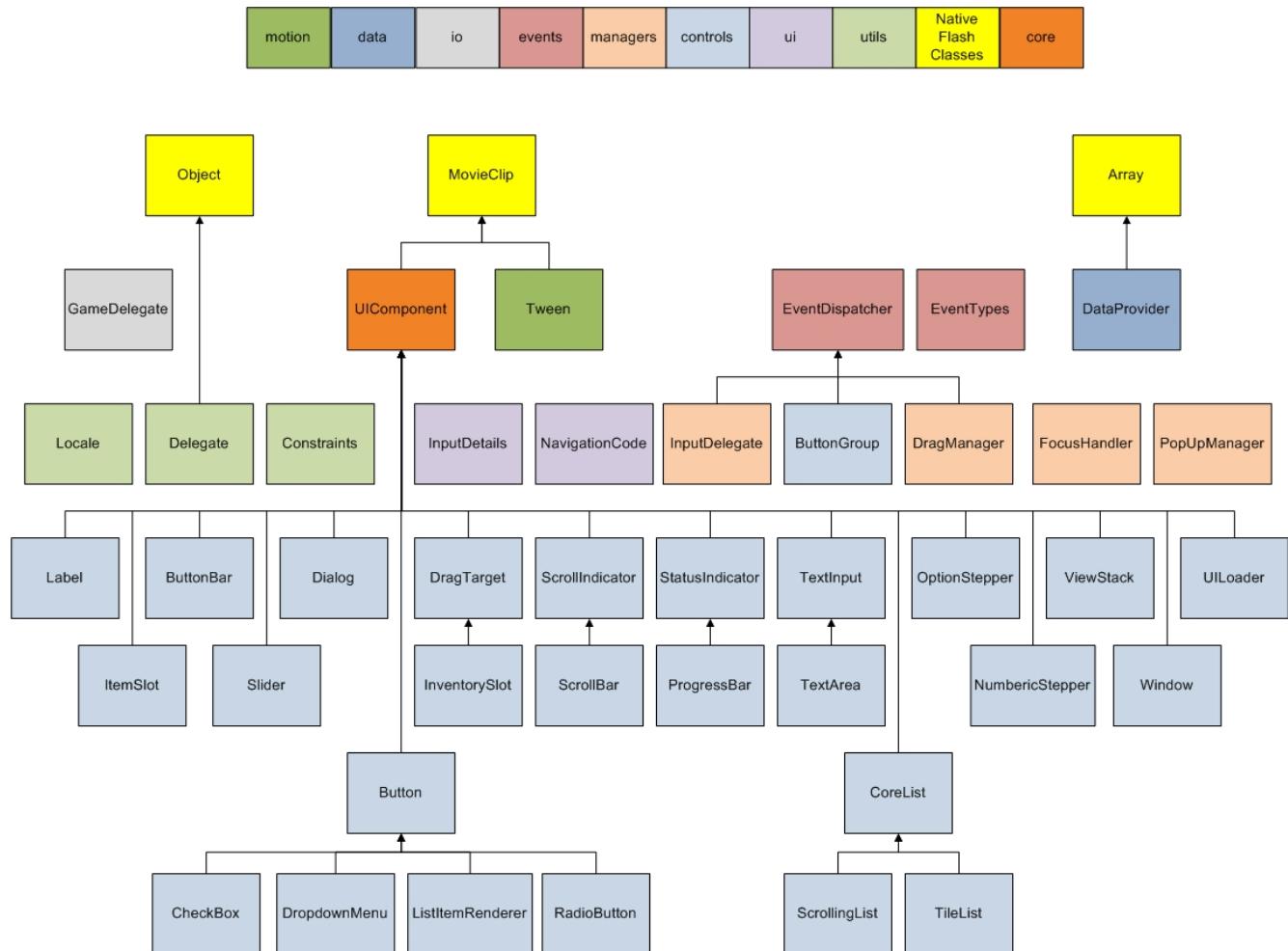


圖 60: Scaleform CLIK 類層次結構

4.1 UI元件UIComponent

內置元件章描述了與Scaleform

CLIK捆綁的元件使用的類。所有這些元件都繼承了UIComponent類(gfx.core.UIComponent).的核心功能。這些類為所有CLIK元件的基礎，Scaleform推薦自定義元件為UIComponent的子類。

UIComponent類本身從Flash 8

視頻剪輯MovieClip類繼承而來，因此繼承了標準Flash視頻剪輯MovieClip的所有屬性和方法。一些自定義讀/寫屬性也被UIComponent類所支援。

- **disabled;**
- **visible;**
- **focused;**
- **width;**
- **height;**
- **displayFocus:** 如果元件應該顯示焦點狀態則設置為true，更多資訊請參考[焦點處理](#) 小節。

UIComponent擁有幾個需要子類實現的空方法。這些方法包括：

- **configUI:** 元件配置調用；
- **draw:** 但元件無效時候調用，更多資訊，請參考[失效](#) 小節；
- **changeFocus:** 當元件接收或失去焦點時被調用；
- **scrollWheel:** 當滑鼠游標在元件上方滾動滾輪時被調用。

UIComponent混合到EventDispatcher類用來支援事件預訂和指派，因此，所有的子類都支援事件預訂和指派。更多資訊，請參考[事件模型](#) 小節。

4.1.1 初始化

改類在onLoad()事件控制碼內執行下列初始化步驟：

- 設置MovieClip默認尺寸大小；
- 執行元件配置，改步驟調用configUI()方法；
- 繪製內容，該步驟調用validateNow()方法，立即執行畫面刷新，例如調用draw()函數。

4.2 元件狀態

幾乎所有的Scaleform

CLIK元件支援視覺狀態，狀態通過元件時間軸上的一個特殊關鍵幀的導航來設置，或者傳遞狀態資訊到子單元。具有三個常用的狀態設置，詳細內容見下面內容。

4.2.1 按鈕元件

任何行爲類似按鈕的元件，回應滑鼠動作，可以被選到此分類。例如使用此類的CLIK元件為Button及其變數、ListItemRenderer、RadioButton和CheckBox。複雜元件的子元素如捲軸ScrollBar也屬於按鈕Button元件。歸到此類的CLIK元件可以直接使用按鈕Button類(gfx.controls.Button)或者使用從按鈕Button類繼承而來的類。

按鈕元件支援的基本狀態為：

- **up** 或默認狀態；
- 當滑鼠箭頭在元件上方或者獲得焦點時為**over** 狀態；
- 當按鈕被點擊時候為**down** 狀態；
- **disabled** 狀態；

按鈕Button元件也支援字首狀態，可以根據其他屬性的值進行設置。默認情況下，核心CLIK按鈕Button元件只支援一個“**selected_**”字首，但元件處於選中狀態時追加到幀的標簽。

按鈕元件支援的基本狀態，包括選中狀態，為以下所示：

- **up** 或默認狀態；
- 當滑鼠箭頭在元件上方或者獲得焦點時為**over** 狀態；
- 當按鈕被點擊時候為**down** 狀態；
- **disabled** 狀態；
- **selected_up** 或者默認狀態；
- 當滑鼠箭頭位於元件上方或獲得焦點時為**selected_over** 狀態；
- 當按鈕被按下時為 **selected_down** 狀態；
- **selected_disabled** 狀態；

注釋：本節中涉及的狀態只是CLIK按鈕元件支援的狀態類型裏的一部分。詳細的狀態列表請參考[CLIK按鈕入門](#)文檔。

CLIK按鈕類提供了一個getStatePrefixes()方法，使開發者能夠根據元件屬性改變列表字首。該方法定義如下：

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

如之前提到的，CLIK按鈕默認情況下只支援“**selected_**”字首。getStatePrefixes()方法根據選擇屬性返回不同字首序列。該字首序列將於適當的狀態標簽協同使用來確定幀的播放。

當狀態在內部進行設置，例如滑鼠滾動，出現一個幀標簽列表的查找表。按鈕類中的stateMap屬性定義了幀標簽映射的狀態。以下為CLIK按鈕類中定義的狀態映射關係：

```
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" ]
}
```

每個狀態可能有多個目標標簽，從狀態表返回的值與getStatePrefixes()返回的字首結合產生一個播放的目標幀列表。下圖描述了用來決定正確幀播放的完整過程：

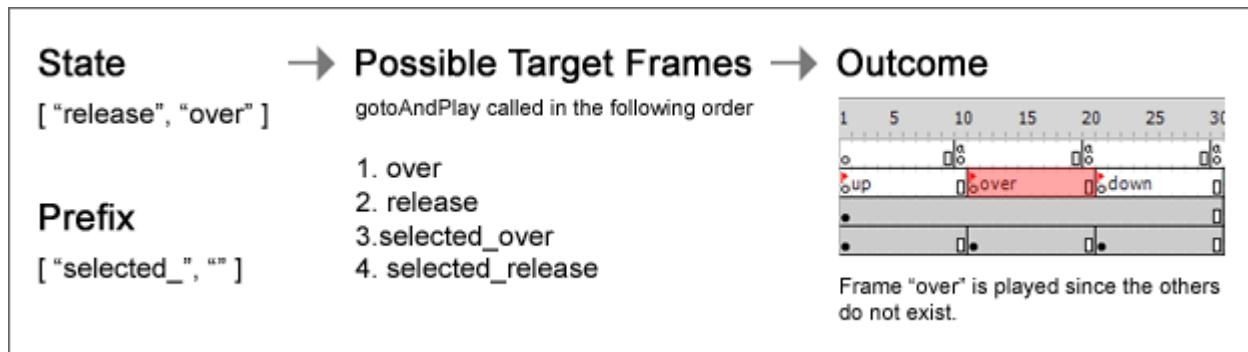


圖 61:決定正確幀播放的過程

如果不能找到特定字首的幀，播放位置總是跳轉到最後的幀，元件為之前請求幀的默認狀態。開發者可以忽略狀態映射而創建自定義行為。

4.2.2 非按鈕交互元件

這些涉及任何元件具有交互和接收焦點功能，但不回應滑鼠事件。例如使用這類方法的CLIK元件為ScrollingList、OptionStepper、Slider

和TextArea。這些元件可能包含子單元可以回應滑鼠事件。無按鈕交互元件支援的狀態為：

- **default** 狀態；
- **focused** 狀態；
- **disabled** 狀態。

4.2.3 非交互元件

這些指向任何元件為非交互性質，但是可以被禁止。標簽元件為狀態所支援的默認設置元件中唯一的非交互元件。狀態支援的非交互元件為：

- **default** 狀態；
- **disabled** 狀態。

4.2.4 特殊案例

有幾個元件不遵守上面描述的狀態規則，狀態指示器StatusIndicator及其子類進度條ProgressBar使用時間軸顯示元件值。播放位置將設置到幀的位置表示元件值的比例。例如，狀態指示器中的一個50幀時間軸最小值為0，最大值為10，當前值設置為5（50%）將gotoAndStop()到第25幀（50幀的50%處）。擴展這些元件管理這些顯示程式非常容易。`updateValue()`方法可以被修改或忽略以改變其行為。

在某些情況下，某些元件除默認行為外還具有特殊模式，支援額外元件狀態。“文本輸入”和“文本區域”元件在設置了“動作按鈕”的情況下，能夠支援“滾動”和“滑出”狀態，以支援滑鼠滾動和滑出事件。

4.3 事件模型

Scaleform CLIK元件框架使用一個通信範例稱之為事件模型*event model*。元件在改變或交互時“分派”事件，容器元件可以訪問不同的事件。使得多個物件可以被通知所發生的改變，而不是只有單個物件，這也是ActionScript 2調用的工作方法。

EventDispatcher類(gfx.events.EventDispatcher)提供了易於使用的API函數，支援事件模型。一個CLIK元件能夠擴展該類或者使用EventDispatcher.initialize()方法將行為混合。但是，如果CLIK元件從UIComponent繼承而來，則只需要在其構造器中調用super()，因為UIComponent元件已經與EventDispatcher相混合。一個EventDispatcher的子類或混合元件支援事件預訂和指派。

4.3.1 最佳使用方法

4.3.1.1 預訂一個事件

預訂一個事件，可以使用addEventListener()方法，包括一個類型參數指定監聽交互事件的類型。反過來使用removeEventListener()方法將取消事件預訂。如果多個監聽器包含相同的參數，只能觸發一個。這些方法中的每一個都需要一個範圍值參數，作為監聽物件和回收信號，其為一個String類型指派事件時調用的函數名稱。

`EventTypes`類(`gfx.events.EventTypes`)包含一個常用事件的枚舉。可以用來替代字串表示事件的類型(`EventTypes.SHOW` 替代 “`show`”)。

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

在本例中按鈕實例設置為監聽 “`itemClick`” 事件並在接收到事件時執行函數 “`callback`” 。回收函數移除事件監聽器。

事件物件的屬性基於原始類型但各不相同。`CLIK`元件產生的詳細事件物件（及其屬性）列表，請參考[內置元件](#)相關章節。

4.3.1.2 指派一個事件

`CLIK`元件希望使用`dispatchEvent()`方法通知預訂監聽器發生的變化或交互動作。該方法需要一個參數：一個包含相關資料的物件，包括一個強制類型屬性指定指派事件類型。元件的框架自動添加一個目標屬性，作為事件指派物件的索引，但是能夠手動設置為用自定義目標進行替換。

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

4.4 焦點處理

Scaleform

`CLIK`元件使用一個自定義焦點處理框架，在多數元件中執行，並且應該在無框架元件和符號中正常工作。所有的焦點變化發生在`Scaleform`播放器層，通過滑鼠或鍵盤（遊戲控制器）改變焦點，或者通過調用`ActionScript`

中的`Selection.setFocus`（實例）來實現。`FocusHandler`管理器(`gfx.managers.FocusHandler`)在一個元件類創建後立即可以顯現，無需直接顯示焦點控制碼`FocusHandler`。

4.4.1 最佳使用方法

當前焦點需要設置在`CLIK`元件實例上，否則焦點為播放器默認狀態。如果未進行設置，焦點管理系統將不能正常工作（如`Tab`鍵不切換焦點等）。可以通過在任何元件上設置焦點屬性為`true`使得焦點得以應用。另外一個應用焦點的方法，也是應用在非元件元素中，如下所示：

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

視頻剪輯MovieClips中無滑鼠控制碼（例如，`onRelease`、`onRollOver`）不能由Scaleform或Flash播放器獲得焦點，也不能產生焦點變化事件。按鈕元件可以自動添加滑鼠控制碼，其他元件將設置為標準視頻剪輯MovieClip為`focusEnabled`和`tabEnabled`屬性。

具有可獲得焦點的子元素元件，如捲軸ScrollBar，使用焦點目標屬性傳遞焦點到它們自身元件。當元件焦點發生變化時，焦點控制碼FocusHandler將遞迴搜索焦點目標鏈，將焦點傳給上一個焦點不返回一個焦點目標。

有時當一個元件不是播放器或引用程式的實際焦點時需要出現一個焦點。`displayFocus`屬性可以設置為`true`使元件表現為獲得焦點的樣子。例如，當滑動條Slider獲得焦點，滑動條軌道也需要獲得焦點。注意當焦點屬性發生改變時元件調用`UIComponent.changeFocus()`方法。

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

反過來，有時候元件需要可以被點擊，但不獲得焦點，如面板拖動，活任何其他只受滑鼠控制的元件。在這種情況下，設置`tabEnabled`屬性為`false`。

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

4.4.2 在複合附件捕獲焦點

複合元件為那些由其他元件組成，如`ScrollingList`、`OptionStepper`或`ButtonBar`。元件本身可能擁有子元件的滑鼠控制碼，但是不具有自身滑鼠控制碼。意思為在Flash和Scaleform中的Selection引擎不支援項和內置導航焦點將無法識別元件，而錯誤得去檢查其子元件。

使元件行為不受合成的約束作為一個獨立的入口，需要以下步驟：

1. 在所有具有滑鼠控制碼的子元件上設置`tabEnabled = false`（如`OptionStepper`中的箭頭按鈕）；
2. 在所有具有滑鼠控制碼的子元件上設置焦點目標屬性。確保在容器元件中設置`focusEnabled = true`；
3. 如果有必要設置子元件中的`displayFocus`屬性為`true`，該屬性位於容器元件`changeFocus`方法之內；

現在當子元件獲得焦點，焦點將轉遞給容器元件。

4.5 輸入處理

由於Scaleform CLIK元件從Flash 8

MovieClip類繼承而來，當接收到用戶輸入時與任何其他MovieClip實例具有相同的行為。如果安裝了滑鼠控制碼元件可以捕獲滑鼠事件。然而，在CLIK上相關鍵盤或類似空孩子氣事件處理具有概念上的區別。

4.5.1 最佳使用方法

所有的非滑鼠輸入都可以被InputDelegate管理類class

(gfx.managers.InputDelegate)截取，InputDelegate將輸入命令轉換為內部的任意一個InputDetails物件(gfx.ui.InputDetails)，或者從遊戲引擎請求輸入命令的值。後者包括了修改InputDelegate.readInput()方法以支援目標應用程式。一旦InputDetails被創建，一個輸入事件就從InputDelegate得到指派。

InputDetails包含了以下屬性：

- 類型，如“key”
- 編碼，如按下鍵的鍵值
- 值，按鈕向量等輸入資訊，或者鍵輸入類型。這是一個數值，它提供了諸如按鈕向量或按鍵類型等輸入的附加資訊。關鍵事件是由鍵的按下與彈起動作而產生的，相應的輸入細節的數值參數也分別為“keyUp”或“keyDown”；
- navEquivalent(navigation equivalent)，定義了可讀的導航方向如“up”、“down”、“left”或者“right”只要可以進行映射。NavigationCode類(gfx.ui.NavigationCode)提供了一個手動通用導航枚舉列表。

FocusHandler從InputDelegate監聽輸入事件並通過焦點路徑傳遞給元件。獲得焦點的元件用來決定焦點路徑，為一個自上而下的元件列表，位於實現the handleInput()方法的現實列表層次當中。

本方法被焦點鏈的最頂部元件調用，InputDetails和pathToFocus併列作為參數傳遞。

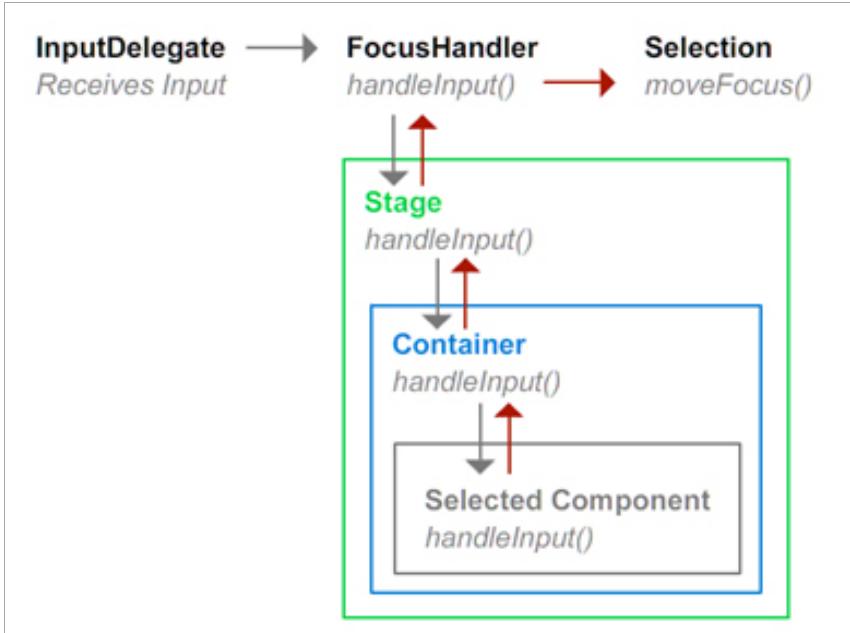


圖 62: handleInput() 函數鏈

FocusHandler控制碼預期從handleInput()函數調用返回一個布林值，用來表示元件或焦點路徑中的任何元件是否處理了輸入。如果接收到回應為false，輸入不為null具有一個navEquivalent，則輸入資訊傳遞給Scaleform播放器。

```

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

每個元件處理輸入將事件傳遞給pathToFocus序列作用的下一個元件，如果輸入資訊被處理則返回true或false。最好不要設想焦點路徑中的下一個元件將正確執行輸入處理，因此該方法應該進行確認，傳遞回一個布林值不僅僅返回輸入資訊傳遞的值。

```

function handleInput(details:InputDetails, pathToFocus:Array):Boolean {
    var nextItem:MovieClip = pathToFocus.shift();
    var handled:Boolean = nextItem.handleInput(details, pathToFocus);
    if (handled) { return true; }
    //自定義處理代碼
    return false; //如果被處理則為true
}
  
```

輸入事件也可以傳遞給元件而不通過焦點路徑。例如，在下拉功能表DropdownMenu中，`handleInput`傳遞到下拉清單元件，即使不在`pathToFocus`序列也是如此。這使得列表可以回應鍵盤命令。

```
function handleInput(details:InputDetails, pathToFocus:Array):Boolean {
    var handled:Boolean = dropdown.handleInput(details);
    if (handled) { return true; }
    // 自定義處理代碼
    return false; // 如果處理則為true
}
```

也可以添加一個事件監聽器到`InputDelegate.instance`中手動監聽輸入事件，並按照這種方法處理輸入資訊。注意這種情況下輸入資訊仍然可以被`FocusHandler`控制碼捕獲並傳遞到焦點層次。

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

`InputDelegate`應該在遊戲中用來管理預期輸入資訊。默認`InputDelegate`處理鍵盤控制、方向鍵轉換以及通用遊戲導航鍵W、A、S和D直接轉換到相等的導航鍵。

4.5.2 多滑鼠游標

在一些系統中，如Nintendo

Wii™，支援多游標設備。CLIK元件框架支援多個游標，但是不允許在單個SWF文件中多個元件獲得焦點。如果兩個用戶都點擊獨立的按鈕，最後的點擊項即為焦點項。

所有在框架中指派的滑鼠事件包含了一個滑鼠索引屬性，作為產生事件的輸入設備索引。

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

4.6 失效

失效為元件使用的一種機制，用來限制元件在多個屬性發生變化時候元件的刷新次數。當一個元件為失效狀態，在下一幀將重繪元件。這使開發者可以暫時忽略元件發生的改變，元件只要在特定時間更新一次即可。在有些情況下元件需要立即更新；但是大多數情況下失效性非常有用。

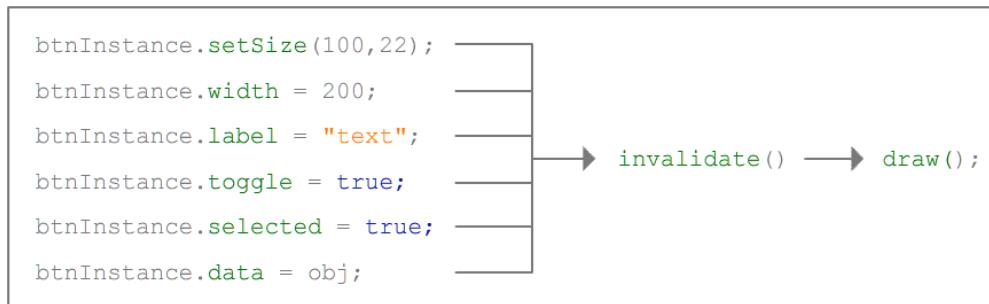


圖 63: CLIK 元件當特定屬性改變時自動變為無效

4.6.1 最佳使用方法

在任何內部元件發生改變後（通常由設置功能引起），則調用UIComponent.invalidate()函數，該函數最終調用UIComponent.draw()函數重繪元件。**invalidate()**方法基於計時器延時產生**draw()**調用避免不必要的更新。開發者使用的元件可能不需要失效屬相，但是至少也需要瞭解這項功能。

在需要立即重繪元件的情況下，開發者可以使用UIComponent.validateNow()方法函數。

4.7 元件縮放

Scaleform CLIK元件按照兩種方式進行縮放：

1. 使用重複繪製，重新安排元件尺寸，元件元素進行縮放適合原來的比例尺寸。具有子黨員並沒有背景的元件採用此方法。
2. 元素不進行縮放維持縱橫比例關係，元件進行縮放，但是元素不縮放。該方法是用在有圖形背景的元件中，**scale9grid**進行了伸展，內部的文本縮放而不扭曲。

由於受到Flash

scale9Grid的限制，元件通常使用回流方法。這要求開發者建立“皮膚”標識，類似于Flash/Flex元件。如果部件擁有可以放縮的子元素，則回流方法的效果最好，因此它應用於**container**和類似實體。

反縮放方法專門為CLIK創建，主要是由於Scaleform

中的**scale9Grid**擴展功能。它允許利用幀狀態創建一個單一資產元件，而無需使用設置其他元件時所用的

密集分層方法。基本的CLIK元件都具有簡約的特性，通常包括一個背景、一個標籤和一個可選圖示或子按鈕，因此非常適合反縮放方法。然而，反縮放並不打算進行類container設置（面板設計等等）。在這種情況下，我們建議使用回流方法，因為它具有約束其餘子元素的背景。

元件可以在Flash

IDE的場景中進行縮放，或者使用寬度和高度屬性進行動態縮放，或者使用setSize()方法函數。縮放後的元件外觀可能在Flash

IDE中不是那麼精確。這就是其局限性，元件作為未編譯的視頻剪輯MovieClips不能進行即時預覽LivePreviews。在Scaleform

Player中測試動畫為確認縮放元件在遊戲中外觀是否精確的唯一方法。CLIK擴展了一個啓動面板改進了這項工作流。

4.7.1 Scale9Grid

多數元件使用第二種縮放方法。Scaleform Player

中的Scale9Grid視頻剪輯MovieClip資源在多數部分都附帶了scale9grid，儘管Flash在包含了MovieClips的情況下將丟棄柵格。這意味著即使scale9grid在Flash

Player中不能工作，在Scaleform中可能可以很好發揮作用。

需要注意的一點是當視頻剪輯MovieClip使用了scale9grid，其子單元也將根據此規則進行縮放。增加Scale9grid到子單元將導致忽略其上級柵格，正常進行繪製。

4.7.2 強制

Constraints類(gfx.utils.Constraints)輔助縮放元件內部的資源縮放和定位。使開發者在場景中定位資源，使資源保持到上級元件邊緣的距離。例如，捲軸ScrollBar元件將根據在場景中的位置重新縮放軌道大小及其位置。Constraints在兩種元件縮放方法中都被用到。

以下代碼增加捲軸ScrollBar資源到configUI()方法的強制物件，向下方向鍵底部對齊，縮放軌道根據其上級元件進行延伸。draw()方法函數包含了代碼以更新強制物件，從而任何元素都用此進行登記。這項更新在draw()函數中完成，因為在元件無效時被調用，通常在元件尺寸發生改變後。

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

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

4.8 元件和資料設置

元件需要使用一個資料源**dataProvider**方法的列表資料。資料源**dataProvider**為一個資料存儲和物件檢索單元，開放了Scaleform CLIK

IDataProvider類(gfx.interfaces.IDataProvider)中定義的所有或部分API函數。

資料源**dataProvider**方法是用一個調用函數的請求模型，代替直接的屬性訪問。這允許資料源在需要時可以從遊戲引擎獲取資料。這具有存儲優勢，也可以將大塊資料設置為小塊，易於管理的資料。

元件使用資料源**dataProvider**包括任何擴展的CoreList (ScrollingList, TileList)、OptionStepper 和DropdownMenu。沒有CLIK框架內的元件需要使用資料源**IDataProvider**介面，只在其API函數中包含了方法。資料源**IDataProvider**類只提供索引。

4.8.1 最佳使用方法

資料源**DataProvider**

類(gfx.data.DataProvider)包含在框架中使用其靜態**initialize()**方法，將資料源**dataProvider**方法添加到任何ActionScript序列中去。元件使用一個資料源**dataProvider**將自動初始化序列使他們可以使用**dataProvider**方法進行訪問。這意味著下列語法初始化一個靜態聲明序列作為完全可操作的資料源**dataProvider**，以及在**IDataProvider**中描述的方法。

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

資料源**dataProvider**應該實現內容為：

- **length:** 可以作為一個屬性或獲取函數返回資料設置的長度；
- **requestItemAt:** 從資料源**dataProvider**請求一個指定項，通常被列表元件使用以在任何時間顯示一個項，如 OptionStepper；
- **requestItemRange:** 從資料源**dataProvider**請求一系列項包括一個開始和結束序號。通常由列表元件使用以顯示更多項，如ScrollingList；
- **indexOf:** 返回項的序號；
- **invalidate:** 標記資料源**dataProvider**變化，提供一個新的資料長度。該方法也應該指派一個“**dataChange**”事件來通知元件資料已更新。資料源**dataProvider**應該支援一個可以公開訪問和反應資料長度的長度屬性。

實例需要資料簡介列表能夠使用一個資料作為資料源**dataProvider**。開發者應該明白在ActionScript 2中存儲資料比在應用程式中自帶資料需要更多的存儲空間和性能開銷。在大量資料設置推薦綁定資料源**dataProvider**和**ExternalInterface.call**基於需求從遊戲引擎獲取資料。

4.9 動態動畫

Scaleform CLIK提供一個自定義Tween類 (`gfx.motion.Tween`)，與Flash 8

Tween類具有類似功能，但是與Scaleform完全相容。該兩種Tween類均支援easing函數，如在`mx.transitions.easing.*`套裝軟體下的函數。

但是，CLIK Tween類與其副本有明顯區別。首先，它安裝Tween方法到Flash 8

視頻剪輯MovieClip類的原型。這向所有的MovieClips開放了Tween功能，不僅僅是CLIK元件。其次，CLIK

Tween類使用`onEnterFrame`因此每個MovieClip每次只能出發一個Tween。如果一個已經出發一個Tween的MovieClip需要創建一個新的Tween，則新的Tween將停止之前的一個。然而Tween方法支援單個MovieClip的多個屬性，允許同一個物件的不同屬性在相同時間發生變化。下節中的例子展示了如何創建Tweens在同一時間影響多個屬性。

4.9.1 最佳使用方法

Tween類在MovieClip開放了兩個主要的方法：`tweenTo()` 和

`tweenFrom()`。`tweenTo()`方法將MovieClip的當前屬性值映射到在函數參數中指定的一個屬性。`tweenFrom()`方法從參數列表中指定的屬性值映射到當前值。

在使用Tween方法之前，必須隨MovieClip原型進行安裝。Tween類提供了一個幫助靜態函數執行函數：`Tween.init()`。

```
import mx.transitions.easing.*;
import gfx.motion.Tween;
Tween.init();           // 安裝tween方法到MovieClip原型
// 執行一個1秒鐘的Tween動畫從當前垂直位置和alpha值到指定的值
myMovieClip.tweenTo(1, {_x: 200, _alpha: 0}, Strong.easeIn);
```

當Tween動畫結束時被通知，只需在Tween物件中創建一個`onTweenComplete()`函數。

```
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 彈出式支援

Scaleform

CLIK包含了PopUpManager類(gfx.managers.PopUpManager)，支援彈出視窗如對話方塊和提示工具。對話方塊元件使用PopUpManager來顯示內容。

4.10.1 最佳使用方法

PopUpManager擁有幾個靜態方法輔助彈出視窗的創建和維護。`createPopUp()`方法能夠被用到任何視頻剪輯MovieClip符號（包括CLIK元件）的鏈結ID創建彈出視窗。文本參數被用來創建一個彈出視窗的實例，同時相關參數用來定位彈出功能表。這些參數在視頻剪輯MovieClips中都需要用到。

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

Scaleform擴展`topmostLevel`參數用來保證彈出視窗總是顯示在場景中所有其他部件的上面，由於與庫相關的問題，使用此方案替代在`root`層創建彈出窗口。如果導入一個子SWF文件到上級元件，試圖創建一個在上級內容所在的路徑的庫定義的符號實例，則將產生符號查找錯誤，因為上級元件無法訪問子元素的庫。不幸的是，沒有專門工作區域來解決這個問題，因此`topmostLevel`方案為最佳選擇。

注意`topmostLevel`也能應用到非彈出視窗，在場景中保持此類元素的Z軸序列。因此，在彈出功能表頂部繪製滑鼠游標，游標可以創建在最高處位置或層次並設置`topmostLevel`屬性為`true`。

4.11 拖放

DragManager 類

(gfx.managers.DragManager)支援初始化和拖放操作。該類是一單個元件，提供了一個實例屬性訪問唯一的物件。使用此物件，開發者可以開始和結束拖放操作。

`startDrag()`方法能夠被調用使用場景中的MovieClip或者一個符號ID開始拖放操作，該操作將創建一個用來拖動的符號實例。`stopDrag()`方法結束拖動操作並通知所有監聽器。DragManager與InputDelegate一同註冊，允許使用鍵盤或類似控制器退出拖放操作。

4.11.1 最佳使用方法



圖 64:動作中的DragDemo

DragManager只執行拖動管理。依賴于開發者創建元件利用DragManager來提供拖放功能。Scaleform CLIK包括一個例子為拖放目標類名為DragTarget (gfx.controls.DragTarget)。DragTarget擴展UIComponent因此分類為一個CLIK元件。擁有幾個唯一特性作為DragManager的補充。

DragTarget具有一個拖放類型的概念。這些拖動類型能夠按照每個DragTarget進行配置使元件允許或者禁止拖動操作。為實現這些拖動類型，DragTarget也需要隨DragManager 為dragBegin 和dragEnd安裝事件監聽器。這使得DragTarget可以顯示其是否支援拖放操作。

CLIK捆綁的演示文件中使用兩個元件，作為子類或者組成DragTarget來展示使用DragManager 和 DragTarget進行的拖放操作。這兩個元件為InventorySlot 和 ItemSlot，這些元件類作為CLIK框架的一部分提供，但是，注意他們只是作為一個例子表示可以實現的功能，並不是一個所有應用場合下的完整解決方案。

InventorySlot 類

(gfx.controls.InventorySlot)支援顯示一個圖示集，用來表示各個條目，通常在角色遊戲中看到。將DragTarget類分為子類提供放置支援，包含了代碼可以使用DragManager產生拖動操作。當拖動操作開始時，

伴隨游標創建一個圖示的拷貝。在一個有效的放置操作，目標InventorySlot將改變其拖動操作時的圖示。

ItemSlot 類

(gfx.controls.ItemSlot)與InventorySlot非常類似，但是包含了額外的功能以支援滑鼠事件，如滑鼠點擊。替代子類，ItemSlot包含了DragTarget的一個實例。也包含了CLIK按鈕的一個實例。這兩個子類為ItemSlot提供了必要的功能以支援拖動和放置操作以及按鈕操作。

4.12 雜項

4.12.1 代理Delegate

代理Delegate 類

(gfx.utils.Delegate)提供一個靜態方法在對應範圍內以創建一個函數代理。不被Scaleform CLIK元件使用，主要被DragManager使用，使用實例如下：

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

4.12.2 本地Locale

Locale類 (gfx.utils.Locale)

為自定義本地化方案提供了一個介面。Scaleform本地化方案執行內部的轉換查詢不需要來自ActionScript 的外部查詢。但是，自定義轉換可能需要此查詢。Button、Label 和 TextInput

元件以及它們的子類都需通過Locale.getTranslatedString()

方法查詢一個本地化字串。默認的實現方法為簡單返回相同的值。需要由開發者修改Locale類以支援自定義本地化方案。

5 實例

下面小節包括一些使用Scaleform CLIK元件的例子。

5.1 基礎

一下實例比較簡單，但展示了易於使用的Scaleform CLIK框架執行常規任務。

5.1.1 包含兩個textField的ListItem Renderer



圖 65: 滾動列表ScrollingList 展示包含兩個標簽的列表項

滾動列表元件默認使用ListItemRenderer來顯示行內容。然而ListItemRenderer只支援單個文本區域textField。很多情況下列表項需要多個文本區域textField用來顯示，或者甚至不需要文本區域如圖示。本例展示了如何添加兩個文本區域到列表項。

首先，定義需求，物件為一個自定義ListItemRenderer支援兩個文本區域。自定義ListItemRenderer應該與滾動列表ScrollingList相容。由於目的為使得每個列表項具有兩個文本區域textFields，資料也應該不止單個字串列表。本例中我們使用以下資料源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"}];
```

資料源包含的物件具有兩個屬性：fname 和 lname。該兩個屬性將顯示在兩個列表項的文本區域textField中。

由於默認的ListItemRenderer只支援一個文本區域textField，需要能夠支援兩個文本區域textField。最簡單的實現方法為將ListItemRenderer類作為子類。一下為調用MyItemRenderer類的源代碼，其中使用了ListItemRenderer子集並添加了兩個文本區域的基本功能支援。（代碼位於MyItemRenderer.as文件中，文件位於FLA工作目錄）：

```
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方法在本子類中沒有涉及。setData方法在列表項收到來自容器列表元件(ScrollingList、TileList等)項資料時被調用。在ListItemRenderer中，本方法設置一個textField的值，而MyItemRenderer用來設置textField的值，同時也存儲一個索引到內部專案物件。此專案物件在updateAfterStateChange方法中被重用，此方法在ListItemRenderer狀態發生改變時被調用，此狀態的變化需要文本區域textField刷新值。

到此，已經定義了具有複雜列表項支援列表項渲染的ListItemRenderer類的資料源dataProvider。要連接所有相關列表元件，必須創建一個符號以支援該型的ListItemRenderer類。本例中，最快的實現方法為修改ListItemRenderer符號使其具有兩個文本區域textField調用‘textField1’和

‘textField2’。下一步該符號標識和類必須修改為MyItemRenderer。為在列表中使用MyItemRenderer元件，修改列表實例的itemRenderer檢查屬性，從‘ListItemRenderer’改變到‘MyItemRenderer’。

現在運行FLA，列表應該顯示出來包含列表元素顯示兩個標簽：在資料源dataProvider中設置的fname和lname屬性。

5.1.2 圖元滾動視圖

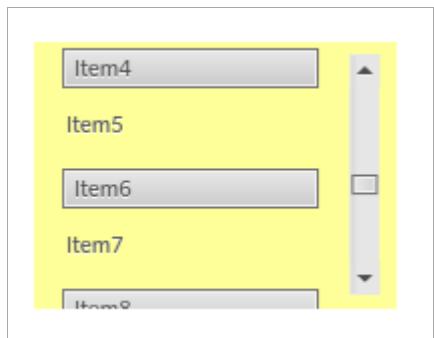


圖 66: 按圖元滾動包含CLIK元素的視圖

按圖元滾動通常用於複雜用戶介面。本例展現了如何使用CLIK捲軸ScrollBar元件簡單得實現此功能。

首先，創建一個新的符號用於滾動視圖，這提供了一個容器，簡化了所需的偏移量計算。在滾動視圖容器中，從頂部到底部順序設置一下圖層。這些圖層並不需要，但是用來作為說明：

- **actions**: 包含ActionScript代碼使得滾動視圖工作；
- **scrollbar**: 包含一個CLIK捲軸ScrollBar實例，該實例稱為‘sb’；
- **mask**：由矩形或MovieClip定義的蒙板圖層，設置圖層屬性為一個蒙板；
- **content**：包含需要滾動的內容；
- **background**：可選層包含了一個背景突出無蒙板區域。

添加相關元素到圖層並創建一個符號將其保存。通過創建一個符號保存所有的內容，滾動內容的任務變得十分簡單。調用這些內容實例‘content’並設置y軸為0。這確保滾動邏輯上不需要計算不同的偏移量。然而，這些偏移根據滾動視圖的複雜性也許需要用到。

到此，定義了所需創建的一個簡單滾動視圖，以及需要互相關聯的元素名稱結構。將以下ActionScript代碼放到code圖層的第一幀：

```
// 133 為視圖尺寸（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;
}
```

由於捲軸不與其他元件連接，需要手動配置。`setScrollProperties`方法就是用來使其在一個位置滾動，並設置完全顯示內容的最大值和最小值。本例中的捲軸位置為圖元格式。運行文件，滾動視圖可以通過與捲軸交互進行改變。

注意在Scaleform中的擴展蒙板使用將明顯降低性能，特別是HUD的應用。Scaleform建議開發者需要評估其性能指標。

5.2 合成

本節以下內容介紹了幾個Scaleform

CLIK捆綁的合成演示。只提供了上層實現細節。注意學習這些內容需要瞭解Flash和ActionScript 2。

5.2.1 使用ScrollingList 的TreeView

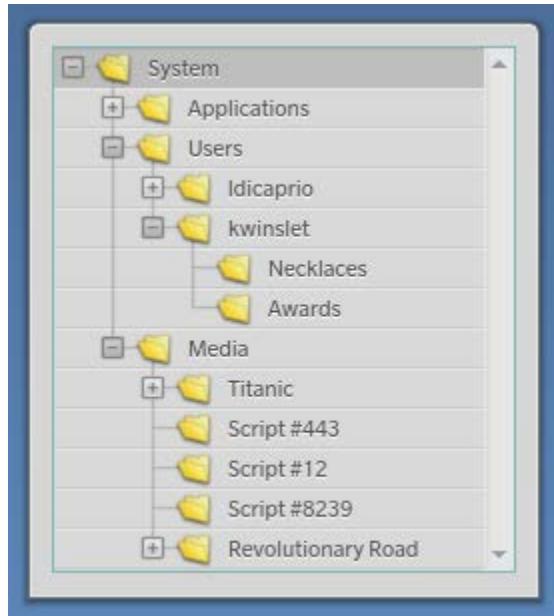


圖 67: TreeView 演示

樹狀視圖演示TreeView Demo

(*Resources/AS2/CLIK/demos/TreeViewDemo.fla*)使用ScrollingList作為基本元件實現一個樹狀視圖控制介面，並包含了自定義ListItemRenderer 和 DataProvider。

讓我們分析普通滾動視圖ScrollingList和樹狀視圖的區別：

- 樹狀視圖中的元素數量能夠根據專案層次狀態進行改變，但是仍然為元素的一個線性序列；
- 樹狀視圖項鄙視使用一個可視導航器顯示樹的層次深度；在多數情況下將提供一個“tabbed”查找功能。
只要改變ListItemRenderer為‘tab’其內容。

- 樹狀視圖必須包含一個機制允許用戶擴展和收縮每個項。同樣，可以修改**ListItemRenderer**以包含一個按鈕元素提供此項功能。

滾動列表為構建一個樹狀視圖的有效工具。在演示文件中，定義了一個自定義的**ListItemRenderer**名為**TreeViewItemRenderer**以及一個自定義的資料源名為**TreeViewDataProvider**。演示也使用了一個較小的工具類名為**TreeViewConstants**提供了常規列舉功能。所有這些類都可以在*Resources/AS2/CLIK/demos/com*目錄下找到。

第一項為資料描述決定何時實現一個樹狀視圖。**ActionScript**

物件本質上為一個哈希表，該演示使用了此項屬性。樹由一個物件樹構成，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"}
          ...
      ]}
```

樹中的每個項描述為帶有兩個屬性的一個物件：**label**

和**nodes**，通過陣列存放子節點。用滾動列表**ScrollingList**展現層次，需要一個自定義資料源用來解析和維護樹狀結構並開放相關資料到列表。

TreeViewDataProvider將樹狀物件收接受到其構造器並執行一個預處理步驟，用特殊屬性注釋每個項，只對資料源和自定義

ListItemRenderer有用。這些屬性包括項的類型、擴展/收縮狀態、上級/同級鏈結和一個線性圖形描述器陣列，描述了渲染時用到的線性圖像。**TreeViewDataProvider**實現了CLIK資料源需要的相關公共方法，但是服從多數工作的檢索幫助功能。例如，專案範圍檢索首先使用列表頂索引查找樹中物件，然後搜集一個項的線性序列，通過樹的遍曆開始於頂部項。

TreeViewItemRenderer通過滾動列表**ScrollingList**從**TreeViewDataProvider**接收資料。獲得的資料物件包括了所有渲染項需要的元資料，不需要知道樹的結構。未經更改的**ListItemRenderer**元件符號在此演示中被重新使用，其類鏈結設置為**TreeViewItemRenderer**。列表項的文件夾圖示和線性鏈結圖形可以被動態創建。這些圖形元素預先保存在緩存中以避免相同‘tab’中複製相同的符號。**ListItemRenderer**中的文本區域**textField**元素向右移動，以專案的深度為基礎。

TreeViewConstant類包含的列舉項如下所示：

- 層次中的節點類型：open node、closed node、leaf node;
- 文件夾圖示類型；
- 線性連接器類型。

後面的列舉項只用來顯示，該演示使用文件夾圖示和線性圖形提供了專案之間的無縫連接，不管其深度為多少。為實現這種效果，文件夾圖示和線性圖形必須覆蓋所有不同展開和搜索視圖設置。

由於該樹狀視圖基於運動列表**ScrollingList**，一個捲軸**ScrollBar**元件能夠被連接，捲軸根據樹的層次狀態改變其外觀，樹的展開/收縮/狀態影響捲軸的控制範圍。與捲軸交互可以根據預期改變滾動列表**ScrollingList**視圖。

5.2.2 可複用視窗

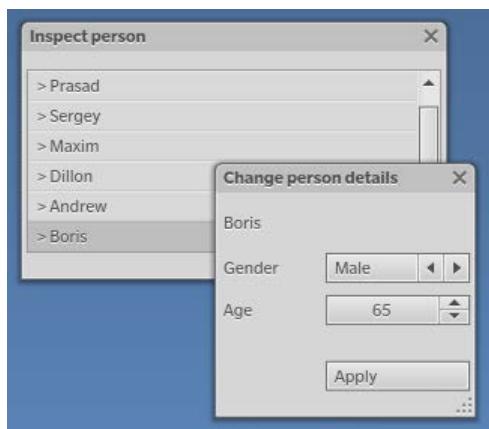


圖 68: 視窗演示

視窗演示Window Demo

(*Resources/AS2/CLIK/demos/WindowDemo.fla*)提供了一個簡單的可複用視窗元件，可以顯示任意內容。視窗元件類能夠在*Resources/AS2/CLIK/demos/com/scaleform/*目錄下找到。

該視窗元件以核心UIComponent類為子類因此規類為CLIK元件。不屬於核心框架類之下因通常視窗元件定義比較困難，單獨使用實現起來更加高效。元件添加更多的特性，將增加記憶體使用降低性能。

因此，本類只實現了視窗元件所需要的常用特性核心集，它們為：

- 可修改的標題欄；
- 可拖動的標題欄和窗口背景；
- 關閉按鈕；
- 伸縮邊界（只能向下和向右伸縮）；
- 最小和最大維數。

Window類開放了幾個檢查屬性：

Title	窗口標題
formPadding	視窗邊界和內容的填充數
formType	導入內容類型，該值可以為 ‘symbol’ 或 ‘swf’，如果為 ‘symbol’ 內容從褲導入，如果為 ‘swf’ 則導入一個擴展SWF文件。
formSource	內容的來源，可以為一個符號名稱（如果 formType 為 ‘symbol’ ）或者SWF檔案名（如果 formType 為 ‘swf’ ）。
allowResize	顯示或隱藏調整大小的按鈕。
minWidth, maxWidth, minHeight, maxHeight	窗口的調整尺寸，如果最大值設置為一個負值，則按照最小值來計算。將最大值都設為負值與隱藏調整按鈕效果相同，完全禁止了恢復。如果最大值設置為0，元件可以調整大小沒有最大值限制。最小值總是為內容的初始化大小。

添加一個檢查屬性到自定義元件類非常簡單。如果一個公共屬性在獲取或設置屬性時不需要執行任何代碼，可以如下聲明：

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

關於檢查元資料語法的更多資訊，請參考Flash文檔。注意檢查元資料支援實際屬性的混合，即為_formType屬性可以作為formType的別名以增強可讀性。

如果屬性在設置或獲取值後需要執行代碼，則屬性應該定義獲取和設置函數：

```
[Inspectable(defaultValue="Title")]
```

```

public function get title():String { return _title; }
public function set title(value:String):Void {
    _title = value;
    invalidate();
}

```

本視窗元件本身支援多個子單元，如標題按鈕、關閉按鈕和調整按鈕。這些按鈕每一個的都是一個**CLICK**按鈕元件。由於這些按鈕在類中需要用到，元件符號必須包含相應的子單元以反映這個需求。

由於窗口類以**UIComponent**類為其子類，包括了兩個主要的方法：**configUI()**和**draw()**。**configUI()**方法設置按鈕監聽器和調整大小的強制物件。**draw()**方法通常以手動或者強制物件形式執行元件排布繪製。然而，也包括使用一個符號名或者文件路徑導入內容。在導入一個私有方法後，調用**configForm()**設置內容。

使用Flash 8

MovieClipLoader執行文件導入。由於當線程支援有效時文件導入在**Scaleform**中非同步執行，元件從**MovieClipLoader.onLoadComplete()**內部調用**configForm()**函數，這在文件導入完畢時候被觸發。導入的形式內容，無論是通過符號還是文件路徑導入，都添加到內部強制物件。在視窗元件進行調整後，將為無效狀態。這將導致**draw()**方法的調用，從而在強制物件上執行一個更新操作使用新的尺寸。由於形式內容註冊到強制物件，通過**validateNow()**方法通知變化。

如果形式內容繼承自**UIComponent**類，則將調用自身的**draw()**方法。但調用**draw()**方法後可以進行形式內容的分佈管理，然後可以定義個**validateNow()**方法以重新繪製自身元素。視窗演示包括了幾個符號和**SWF**文件的例子，其中使用了**draw()**方法和**validateNow()**方法來維持形式分佈。

視窗元件也包含特殊的邏輯移動在前景中被滑鼠游標按下的視窗。確保所有視窗元件實例存在於相同的層次，使元件獲得焦點到下一個最高深入。

```

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

```

這為實現該行為的一種方法。另一種方法為創建**WindowManager**以維持場景中所有視窗元件的z軸。**WindowManager**方法實際上比本例中視窗元件具有更強的易用性。