

Autodesk® Scaleform®

Scaleform CLIK AS3 ユーザー ガイド

本書では Scaleform CLIK フレームワークと、付属のプリビルド コンポーネントの実装について詳しく説明しています。

著者: Prasad Silva、Matthew Doyle
バージョン: 2.0
最終更新日: 2010 年 8 月 19 日

Autodesk®
GAMEWARE 

Copyright Notice

Autodesk® Scaleform® 4.4

© 2014 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, AutoCAD LT, AutoCAD, Autodesk, the Autodesk logo, Autodesk 123D, Autodesk CAM 360, Autodesk Homestyler, Autodesk Inventor, Autodesk MapGuide, Autodesk Streamline, AutoLISP, AutoSketch, AutoSnap, AutoTrack, Backburner, Backdraft, Beast, BIM 360, Burn, Buzzsaw, CADmep, CAiCE, CAMduct, CFdesign, Civil 3D, Cleaner, Combustion, Communication Specification, Configurator 360™, Constructware, Content Explorer, Creative Bridge, Dancing Baby (image), DesignCenter, DesignKids, DesignStudio, Discreet, DWF, DWG, DWG (design/logo), DWG Extreme, DWG TrueConvert, DWG TrueView, DWGX, DXF, Ecotect, ESTmep, Evolver, FABmep, Face Robot, FBX, Fempro, Fire, Flame, Flare, Flint, FMDesktop, ForceEffect, FormIt, Freewheel, Fusion 360, Glue, Green Building Studio, Heidi, Homestyler, HumanIK, i-drop, ImageModeler, Incinerator, Inferno, InfraWorks, InfraWorks 360, Instructables, Instructables (stylized robot design/logo), Inventor, Inventor HSM, Inventor LT, Kynapse, Kynogon, LandXplorer, Lustre, MatchMover, Maya, Maya LT, Mechanical Desktop, MIMI, Mockup 360, Moldflow Plastics Advisers, Moldflow Plastics Insight, Moldflow, Moondust, MotionBuilder, Movimento, MPA (design/logo), MPA, MPI (design/logo), MPX (design/logo), MPX, Mudbox, Navisworks, ObjectARX, ObjectDBX, Opticore, Pipeplus, Pixlr, Pixlr-o-matic, Productstream, Publisher 360, RasterDWG, RealDWG, ReCap, ReCap 360, Remote, Revit LT, Revit, RiverCAD, Robot, Scaleform, Showcase, Showcase 360 ShowMotion, Sim 360, SketchBook, Smoke, Socialcam, Softimage, Sparks, SteeringWheels, Stitcher, Stone, StormNET, TinkerBox, ToolClip, Topobase, Toxik, TrustedDWG, T-Splines, ViewCube, Visual LISP, Visual, VRED, 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	
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

Table of contents

1 はじめに	1
1.1 はじめに.....	1
1.1.1 Scaleform CLIK の内容	1
1.2 UI についての考察	2
1.3 コンポーネントについて	3
1.4 フレームワークの基本.....	4
1.4.1 イベント	4
1.4.2 フォーカス.....	5
2 あらかじめビルトされたコンポーネント	6
2.1 基本的なボタンとテキストのタイプ	6
2.1.1 Button.....	7
2.1.2 CheckBox	12
2.1.3 Label	17
2.1.4 TextInput	19
2.1.5 TextArea.....	23
2.2 グループ タイプ	27
2.2.1 RadioButton.....	27
2.2.2 ButtonGroup	32
2.2.3 ButtonBar.....	34
2.3 スクロール タイプ	37
2.3.1 ScrollIndicator	37
2.3.2 ScrollBar.....	40

2.3.3	Slider	43
2.4	リスト タイプ	47
2.4.1	NumericStepper	48
2.4.2	OptionStepper	51
2.4.3	ListItemRenderer	54
2.4.4	ScrollingList.....	59
2.4.5	TileList	64
2.4.6	DropdownMenu	69
2.5	プログレス タイプ	75
2.5.1	StatusIndicator.....	75
2.6	その他のタイプ	78
2.6.1	Window	78
3	アートの詳細	82
3.1	ベスト プラクティス.....	82
3.1.1	ピクセル パーフェクトなイメージ	82
3.1.2	マスク.....	83
3.1.3	アニメーション	83
3.1.4	レイヤーと描画プリミティブ	84
3.1.5	複雑なスキン	84
3.1.6	2 のべき乗	84
3.2	既知の問題と推奨ワークフロー	84
3.2.1	コンポーネントの複製	84
3.3	スキニングの例	86
3.3.1	StatusIndicator のスキニング	86
3.4	フォントとローカライズ	89

3.4.1	はじめに	89
3.4.2	フォントの埋め込み	89
3.4.3	textField にフォントを埋め込む	90
3.4.4	Scaleform のローカライズ システム	90
4	プログラミングの詳細	91
4.1	UIComponent	91
4.1.1	初期化	92
4.2	コンポーネント ステート	92
4.2.1	ボタン コンポーネント	93
4.2.2	ボタン以外のインタラクティブ コンポーネント	95
4.2.3	非インタラクティブ コンポーネント	95
4.2.4	特殊なケース	95
4.3	イベント モデル	95
4.3.1	使用法とベスト プラクティス	96
4.4	ランタイムでのコンポーネントの生成	97
4.5	フォーカスの処理	97
4.5.1	使用法とベスト プラクティス	98
4.5.2	複合コンポーネントのフォーカスの取得	99
4.6	入力処理	99
4.6.1	使用法とベスト プラクティス	99
4.6.2	複数のマウス カーソル	101
4.7	無効化	102
4.7.1	使用法とベスト プラクティス	102
4.8	コンポーネントのスケーリング	103
4.8.1	Scale9Grid	104

4.8.2	Constraints.....	104
4.9	コンポーネントとデータのセット	105
4.9.1	使用法とベスト プラクティス	105
4.10	ダイナミック アニメーション	105
4.10.1	使用法とベスト プラクティス	106
4.11	レイアウトフレームワーク.....	107
4.12	ポップアップのサポート	110
4.12.1	使用法とベスト プラクティス	110
4.13	ドラッグ&ドロップ	111
4.13.1	使用法とベスト プラクティス	111
5	使用例	113
5.1	基礎編	113
5.1.1	2 つの textField を備えた ListItem Renderer.....	113
5.1.2	ピクセル単位のスクロール表示.....	115
6	よく寄せられる質問	117
7	潜在的な危険	119
8	CLIK AS3 vs. CLIK AS2	119

1 はじめに

本書では、Scaleform® Common Lightweight Interface Kit (CLIK™) のフレームワークとコンポーネントの実装について詳しく説明します。この CLIK User Guide を読み進める前に、「[Getting Started with CLIK](#)」と「[Getting Started with CLIK Buttons](#)」に目を通すことをお勧めします。この 2 つのマニュアルは Scaleform CLIK を起動して実行するまでに必要なステップの説明、基本的なコア コンセプトの紹介、さらに CLIK コンポーネントの作成とスキーリングに関する詳細なチュートリアルを提供しています。ただし、上級ユーザーは本書をすぐに熟読するか、または上記のマニュアルと並行して参照したい場合もあるでしょう。

1.1 はじめに

Scaleform CLIK は、ActionScript™ 2.0 (AS) ユーザー インターフェイス (UI) エレメント、ライブラリ、ワークフロー拡張ツールのセットです。Scaleform 4.0 以降のユーザーが、家庭用ゲーム機・PC・携帯ゲーム機アプリケーションに、リッチで効率的な UI をすぐに実装できるように用意しました。このフレームワークの主要目的は、(メモリと CPU の使用の点で) 軽量化し、簡単にスキーリングを行い、大幅にカスタマイズできるようにすることです。UI コア クラスとシステムの基本フレームワークの外、CLIK には 15 以上のプリビルド汎用インターフェイス エレメント (ボタン、スライダ、テキスト入力など) が添付されており、これらのエレメントを使うと開発者は直ちにユーザー インターフェイス画面を作成、イテレートすることができます。

1.1.1 Scaleform CLIK の内容

コンポーネント: 簡単で拡張可能なプリビルドの UI コントロール

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

クラス: コアのシステム API

InputManager	UIComponent
FocusManager	Tween
DragManager	DataProvider
PopUpManager	IDataProvider
IUIComponent	IList
Constraints	IListItemRenderer

マニュアルとサンプル ファイル:

- Getting Started with CLIK
- Getting Started with CLIK Buttons
- CLIK AS3 API Reference
- CLIK AS3 User Guide
- CLIK Flash Samples
- CLIK Video Tutorials

1.2 UI についての考察

優れた UI を作成するには、まず始めに紙や、Microsoft Visio®などの図形エディタでそのコンセプトを立案します。次に、UI の特定のグラフィック デザインを始める前に、すべてのインターフェイス オプションとその流れを考え出すために、Flash で UI のプロトタイプ化を始めます。Scaleform CLIK は特に、ユーザーが直ちにプロトタイプ化して、完成するまでイテレートすることができるよう設計されています。

フロント エンド UI を作成する方法はいくつもあります。Flash では、Visio や他のフローチャート プログラムと同じように、ページの概念は存在していません。キーフレームとムービー クリップがその役割を果たします。あなたが作ったすべてのページを同じ Flash ファイルに入れ込んだ場合、そのファイルはより多くのメモリを使用する可能性がありますが、ページ間の移動が簡単で速くなる場合もあります。各ページをそれぞれ個別のファイルにすると、全体のメモリ使用量は少なくなるかもしれません、ロード時間が長くなる場合があります。また、各ページを個別のファイルとして持つことは、複数のデザイナーやアーティストが同じインターフェイスを同時に作業できるという利点もあります。技術的な必要条件や設計上の必要条件に加えて、自身の UI プロジェクトの構造を決めるときにワークフローを必ず考慮に入れてください。

従来のデスクトップや Web アプリケーションとは異なり、優れたマルチメディア ゲーム インターフェイスを作成する方法はいくつもあるということを理解しておくことが大切です。それぞれのエンジン、さらに数えるほどしかない各プラットフォームに対しても、少しずつ異なるアプローチがあると言えます。選択が、結果として良く出ることもあれば、悪く出てしまうこともあるでしょう。たとえば、複数ページのインターフェイス デザインを 1 つの Flash ファイルに入れると仮定します。こうすると管理が楽になりトランジションの作成が簡単になりますが、メモリの使用量も増えるので、旧型の家庭用や携帯ゲーム機には悪影響を及ぼす場合もあります。1 つの Flash ファイルですべてを行う場合、複数のデザイナーが同時にそのインターフェイスの別々のパートで作業することができません。複雑なインターフェイスが求められるプロジェクトで、大人数のデザイン チームが存在する、またはその他の特定の技術上、あるいは設計上の必要条件がある場合、UI の各ページを個別の Flash ファイルに分けるほうが良い場合があります。多くの場合、両方のソリューションとも効果がありますが、そのプロジェクトに対しては、1 つのソリューションが他方のソリューションよりも大幅な利点を提供できる場合があります。たとえば、画面をオンデマンドでロード/アンロードしたり、動的に更新したり、ネットワーク経由でストリーミングすることが必要な場合もあります。つまり、UI のアセット 管理は、論理的なレイアウトから実装のワークフローとパフォーマンスの考察に至るまで、常に周到に行うべきです。

Scaleform は開発者が UI の動作の大部分に Flash と ActionScript を使用することを強く推奨していますが、完璧な答えはないので、アプリケーション エンジンのスクリプト言語 (Lua や UnrealScript など) を使って、難しい作業の大部分を行いたいと思うチームもあるかもしれません。このような場合、Flash はどちらかというと、ファイル自体に少量の ActionScript とインターラクティビティを備えたアニメーション プрезентーション ツールとして使われる必要があります。

プロジェクトを先の方まで進行させてしまう前に、技術的、設計上、さらにワークフローの必要条件を早い段階で理解し、次にそのプロセスを通した全体的な手法を継続的に評価して、円滑で良好な結果が出ることを確実にしておくことが、とりわけ重要です。

1.3 コンポーネントについて

始める前に、開発者は Flash コンポーネントについて正確に理解することが重要です。Flash は一連のデフォルトのインターフェイス作成ツールと、コンポーネントと呼ばれる構成ブロックを内蔵しています。しかし、本書の「コンポーネント」とは、ワールドワイドで知られている gskinner.com チームと共に、Scaleform が開発した Scaleform CLIK フレームワークを使って作成されたプリビルドのコンポーネントを指しています。

gskinner.com はグラント・スキナー氏が率いており、彼は Adobe の命によって Flash Creative Suite® 4 (CS4) のコンポーネントを作成しました。gskinner.com は優れた Flash チームの 1 つとして世界的に知られています。gskinner.com の詳細については、<http://gskinner.com/blog> を参照してください。

プリビルドの CLIK コンポーネントについてさらに詳しく理解するため、以下のデフォルトの CLIK ファイルパレットを Flash Studio で開きます：

Scaleform SDK¥Resources¥AS3¥CLIK¥components¥CLIK_Components_AS3.fla

1.3.1.1 検証可能なプロパティ

コンポーネントの重要なプロパティは、Flash IDE の [コンポーネント インスペクタ] パネル、または [パラメータ] タブで構成されています。このパネルを CS4 で開くには、[ウィンドウ] メニューの [コンポーネント インスペクタ] を選択するか、またはキーボードで Shift + F7 キーを押します。これで [コンポーネント インスペクタ] パネルが表示されます。このようなプロパティは「検証可能なプロパティ」と呼ばれています。このパネルでは、AS プログラミングに慣れていないアーティストやデザイナーに、コンポーネントの動作や機能を構成するための便利な方法を提示しています。

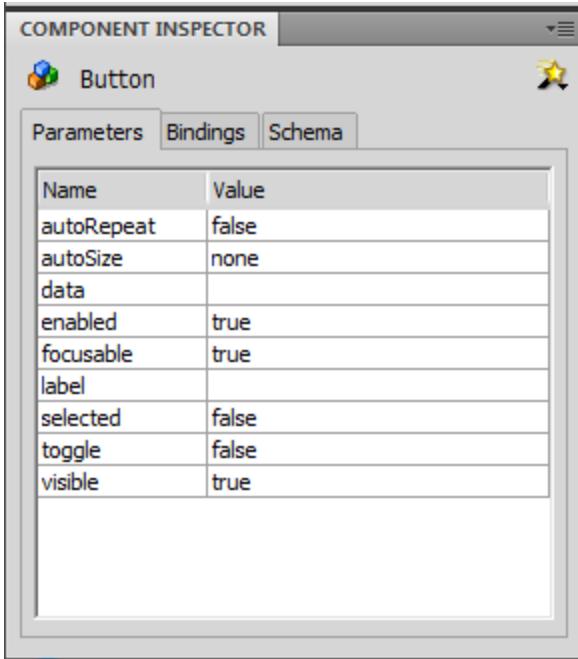


図 1: CS4 の [コンポーネント インスペクタ] に表示された Button コンポーネントの検証可能なプロパティ

これらのプロパティを変更しても、そのコンポーネントを含む SWF ファイルをパブリッシュするまではその変更は確認できません。CLIK コンポーネントはコンパイルされたクリップではないため、Flash IDE は設計時のステージ上に変更を表示しません。。これは、コンポーネントが簡単にアクセスでき、スキニング可能であるために必要なことです。

1.4 フレームワークの基本

1.4.1 イベント

ほとんどのコンポーネントはユーザーとの相互作用、ステートの変更、フォーカスの管理のためにイベントを生成します。これらのイベントは Clik コンポーネントを使用するユーザーインターフェースがうまく機能するために重要です。

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベントサブクラス パラメーターを受け取ります。次のプロパティはすべてのイベントに対して共通です。

- ***type***: イベントのタイプ。
- ***target***: イベントのターゲット。
- ***currentTarget***: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- ***eventPhase***: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE) 。
- ***bubbles***: イベントがバーリングイベントかどうかを示す。

- **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。
- ActionScript 3 Event クラスに関する詳細は、次の Adobe の ActionScript 3 参照文書にあります。
http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/events/Event.html

各コンポーネントの生成するすべてのイベントのリストはこのユーザーガイドの「あらかじめビルトされたコンポーネント」のセクションに、各イベント独自のプロパティと共に掲げています。

Scaleform では、ネイティブのイベントタイプによっては Scaleform のエクステンションで置き換えられるものもあります。たとえば、Extensions.enabled プロパティが true に設定されている場合、Scaleform は常に標準の MouseEvent の代わりに、MouseEventEx を拡張するクラスである MouseEventEx イベントを生成します。MouseEventEx は複数のコントローラーと右/中のマウスのボタンへのサポート用のプロパティを付加します。受け取ったイベントが MouseEventEx のインスタンスかどうかをチェックして、その通りならばイベントオブジェクトをエクステンションタイプに型変換してください。

1.4.2 フォーカス

Flash にはフォーカスという概念があります。デフォルトでは、ステージフォーカスはマウスまたはキーボードで最後に選択された InteractiveObject に設定されます。

一般的に、フォーカスが適用されたコンポーネントだけがキーボード イベントを受け取ります。コンポーネント上にフォーカスを設定する方法はいくつもあります。その方法の一部は、本書の「[プログラミングの詳細](#)」で説明しています。ほとんどの CLIK コンポーネントは、操作されているとき、特にマウスの左ボタン、あるいは同等のコントローラがそのコンポーネント上で押された（クリックされた）ときにもフォーカスが適用されます。Tab キーや Shift+Tab キー（または同等のナビゲーション コントロール）は、表示されているフォーカス可能なコンポーネント間でフォーカス移動します。これはほとんどのデスクトップ アプリケーションや Web サイトで目にするものと同じ動作です。CLIK 以外のエレメントで使用されるフォーカスは、CLIK コンポーネントでも使用されます。つまり、シーン内で Tab キーと Shift+Tab キーを使ってフォーカスを移動するような場合に、Flash 開発者は CLIK エレメントと非 CLIK エレメントをうまく組み合わせて、意図したとおりにフォーカスを動かすことができます。

デフォルトでは Button は Enter キーとスペースバーに応答します。マウス カーソルが Button 上を移動したり、Button から離れるとき、さらに Button にドラッグイン/ドラッグアウトするときもコンポーネントに影響します。

コンソールや携帯ゲーム機では、開発者は簡単にゲームパッドのコントロールを、キーボードやマウスのコントロールにマップすることができます。たとえば、Enter キーは通常、Xbox360 や PS3 のコンソール コントローラの A ボタンと X ボタンにそれぞれマップされます。このマッピングによって CLIK で作成した UI は、各種のプラットフォームで動作できます。

2 あらかじめビルドされたコンポーネント

一見すると Scaleform CLIK はプリビルド UI コンポーネントの基本セットに見えますが、CLIK の真意はより豊富なコンポーネントとインターフェイスを作成するためのフレームワークを提供することです。開発者は自由に、とはいっても、ある程度は外部から求められて、ニーズに合わせてカスタムのコンポーネントを作成し、この CLIK フレームワークに積み重ねることができます。

プリビルド CLIK コンポーネントは、基本的なボタンやチェックボックスから、リスト ボックス、ドロップダウン メニュー、モーダルなダイアログ ボックスなどの複雑な複合コンポーネントまでの標準の UI 機能を備えています。開発者は簡単に、これらの標準のコンポーネントを拡張して機能を追加することができます。一方、カスタム コンポーネントを最初から作成するためのリファレンスとして使うこともできます。

これ以降の章では、各プリビルド コンポーネントについて詳細に説明しています。コンポーネントは複雑さと機能ごとにグループ化されています。それぞれのコンポーネントの説明には、以下のサブセクションを伴っています。

- **ユーザーの操作** : ユーザーがそのコンポーネントを操作する方法。
- **コンポーネントのセットアップ** : Flash オーサリング環境でコンポーネントを作成するときに必要なエレメント。
- **ステート** : コンポーネントが動作するために必要な各種のビジュアル ステート (キーフレーム)。
- **検証可能なプロパティ** : コードを使わずにコンポーネントの特定の状態を簡単に作成するために、Flash オーサリング環境で公開されているプロパティ。
- **イベント** : UI の他のオブジェクトがリッスン可能なコンポーネントによって発生するイベントのリスト。

2.1 基本的なボタンとテキストのタイプ

基本タイプには Button、CheckBox、Label、TextInput、TextArea コンポーネントが含まれています。ほとんどのユーザー インターフェイスにおいて Button は重要な要素であり、CheckBox の機能はこの Button から派生しています。Label は静的なラベル タイプですが、TextInput と TextArea は一行と複数行の textField タイプを表しています。



図 2: Free Realms - メインメニューのボタンの例

Brawler: The Growler Report
Gerold has asked you to deliver his report on the Growler situation to Assistant Mayor Tevin in Snowhill.

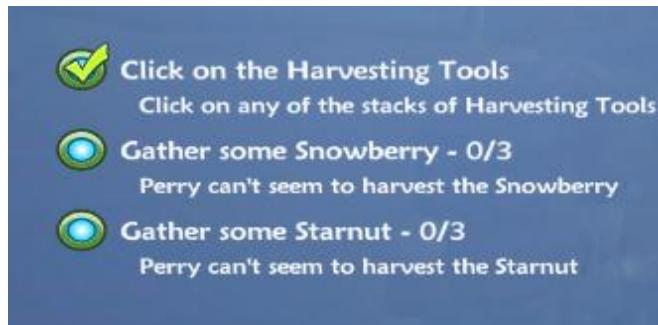


図 3: *Free Realms* - チェックボックスの例

図 4: *Free Realms* - ラベルの例

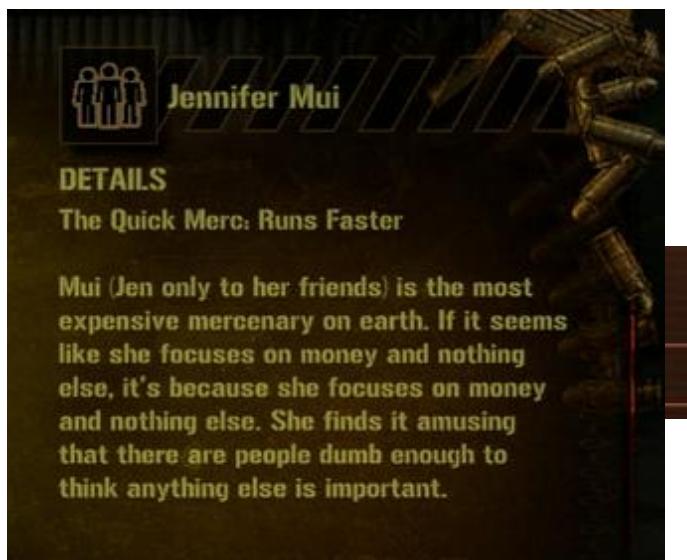


図 5: *Crysis Warhead* - テキスト入力 (TextInput) の例

図 6: *Mercenaries 2* - テキスト領域 (TextArea) の例

2.1.1 Button



図 7: スキンング前のボタン

Button は CLIK フレームワークの基礎コンポーネントです。クリック可能なインターフェイス コントロールが必要な箇所であればどこでも使えます。デフォルトの Button クラス

(scaleform.clik.controls.Button) はラベルを表示する textField と、視覚的にユーザーの操作を示すステートをサポートしています。Button は単独でも、ScrollBar の矢印や Slider のサムなど複合コンポーネントの一部としても使用することができます。クリックでアクティブになるインタラクティブコンポーネントのほとんどは、Button で構成、または拡張しています。

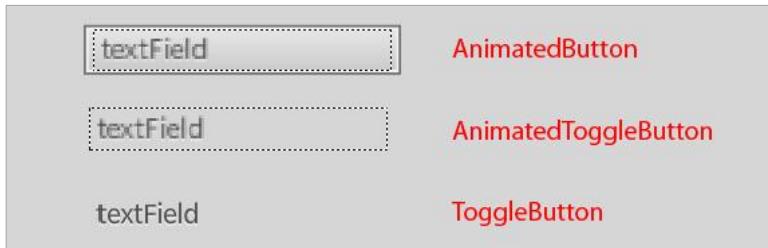


図 8: AnimatedButton、AnimatedToggleButton、ToggleButton

CLIK Button は汎用のボタン コンポーネントです。マウス操作、キーボード操作、ステート、さらに他の機能をサポートしていますので、多くのユーザー インターフェイスでの使用が可能です。また、切り替え機能やアニメーション ステートもサポートします。Button.fla コンポーネント ソース ファイルで提供される ToggleButton、AnimatedButton、AnimatedToggleButton はすべて、同じ 基本となるコンポーネント クラスを使用しています。

2.1.1.1 ユーザーの操作

Button コンポーネントはマウスやその他の同等のコントローラを使って押すことができます。また、フォーカスが適用されたときには、キーボードを使って押すこともできます。

2.1.1.2 コンポーネントのセットアップ

CLIK Button クラスを使用する MovieClip には、以下の指定されたサブエレメントが必須です。オプションのエレメントである場合には、以下に表記されています。

- *textField* : (オプション) TextField タイプ。ボタンのラベルです。
- *focusIndicator* : (オプション) MovieClip タイプ。フォーカスが適用されたステートの表示に 使用される別の MovieClip です。存在する場合、この MovieClip は "show" と "hide" の 2 つ の指定されたフレームを備えている必要があります。デフォルトでは over ステートを使って、 フォーカスされた Button コンポーネントを表しています。ただし場合によっては、この動作 はあまりに限定的ともいえます。デザイナーはフォーカスが適用されたステートと、マウス オーバー ステートを区別したい場合があるからです。focusIndicator サブエレメントはこのような場合に便利です。このサブエレメントを追加すると、Button ステートはフォーカスが適用されたステートに影響を受けません。コンポーネント ステートの詳細については、次の章を参照してください。

2.1.1.3 ステート

CLIK Button コンポーネントはユーザーの操作に基づいた様々なステートをサポートします。これらのステートには以下のものが含まれます:

- *up* またはデフォルトのステート
- マウス カーソルがコンポーネント上にあるとき、またはフォーカスが適用されているときの *over* ステート
- ボタンが押下されたときの *down* ステート
- *disabled* ステート

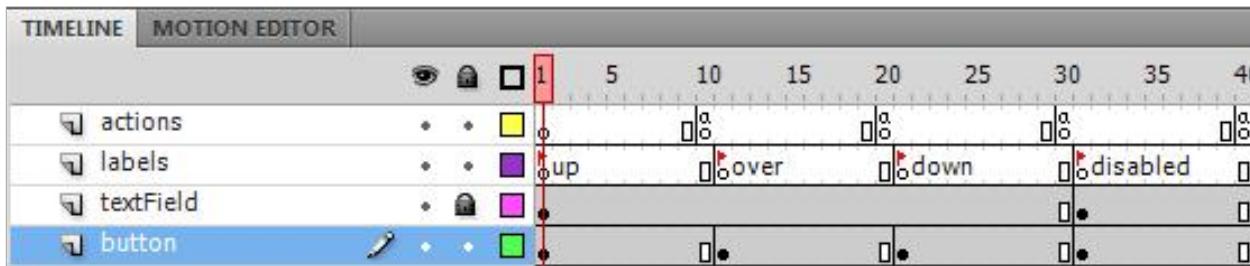


図 9: Button のタイムライン

これらのステートは Flash のタイムラインではキーフレームとして表現され、Button コンポーネントが正しく動作するために必要な最小限のキーフレーム セットです。コンポーネントの機能を拡張して、複雑なユーザー操作やアニメーション トランジションをサポートするその他のステートもあります。このようなステートについては、「[Getting Started with CLIK Buttons](#)」を参照してください。

2.1.1.4 検証可能なプロパティ

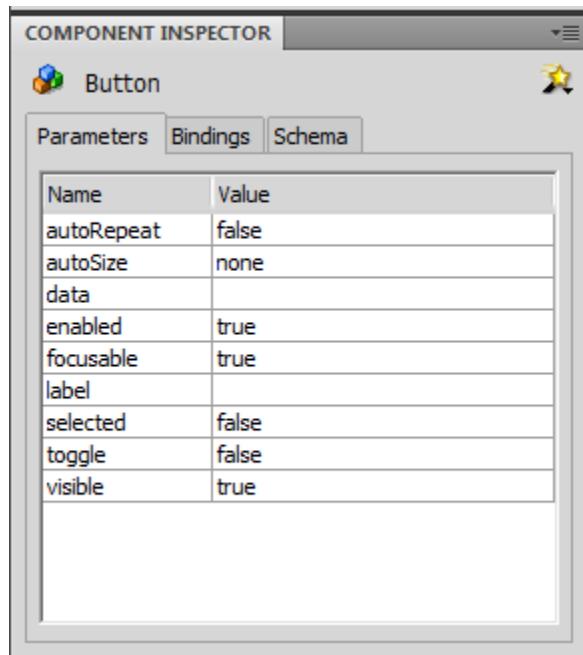


Figure 10: Button component inspectable properties in the CS4 component inspector.
Button コンポーネントの検証可能なプロパティは以下のとおりです:

autoRepeat	ボタンが押し続けられたときに「クリック」イベントを出すかどうかを決定します。
autoSize	ボタンが、その中のテキストサイズに合うように拡大縮小して、サイズの変わったボタンがどの方向に整列されるかを決定します。autoSize プロパティーを TextFieldAutoSize.NONE に設定するとサイズは変わりません。
data	ボタン関連のデータ。このプロパティーは、特に ButtonGroup でボタンを使用する場合に役に立ちます。
enabled	false に設定しておくとボタンを無効にします。無効にされたコンポーネントはユーザーの入力を受け取りません。
focusable	コンポーネントのフォーカス管理を有効化/無効化します。フォーカス可能なプロパティーを false に設定すると Tab キー、方向キー、マウスボタンに基づいたフォーカス変更へのサポートを削除します。
label	ボタンのラベルを設定します。
selected	ボタンの選択されたステートを設定します。ボタンは選択された、非選択の 2 つのマウスステートを持ち得ます。ボタンの toggle プロパティーが true の場合、選択されたステートはそのボタンがクリックされたときに変更されますが、toggle プロパティーが false でも ActionScript を用いて選択されたステートを設定可能です。
toggle	Button のトグル (切り替え) モードを設定します。true に設定すると、この Button はトグル ボタンとして動作します。
Visible	false に設定した場合、ボタンを非表示にします。

2.1.1.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベントサブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE) 。
- **bubbles**: イベントがバーリングイベントかどうかを示す。
- **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

Button コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	visible プロパティーが、実行時に true に設定されました。
ComponentEvent.HIDE	visible プロパティーが、実行時に false に設定されました。
ComponentEvent.STATE_CHANGE	コンポーネントのステートが変更された。

NGE	
FocusHandlerEvent.FOCUS_I N	コンポーネントがフォーカスを受け取った。
FocusHandlerEvent.FOCUS_O UT	コンポーネントがフォーカスを失った。
Event.SELECT	選択されたプロパティーが変更された。
MouseEvent.ROLL_OVER	<p>マウスのカーソルがボタンの上に来た。</p> <p><i>mouseIdx</i>: イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p> <p><i>buttonIdx</i>: イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p>
MouseEvent.ROLL_OUT	<p>マウスのカーソルがボタンの上から離れた。</p> <p><i>mouseIdx</i>: イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p> <p><i>buttonIdx</i>: イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p>
ButtonEvent.PRESS	<p>ボタンが押された。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p> <p><i>isKeyboard</i>: イベントがキーボード/ゲームパッドで生成されたときは true、イベントがマウスで生成されたときは false。</p> <p><i>isRepeat</i>: イベントが autoRepeating ボタンを押し続けることで生成されたときは true、ボタンが現在反復していないときは false。</p>
MouseEvent.DOUBLE_CLICK	<p>ボタンがダブルクリックされた。<i>doubleClickEnabled</i> プロパティーが true のときだけ発せられる。</p> <p><i>mouseIdx</i>: イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p> <p><i>buttonIdx</i>: イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p>
ButtonEvent.CLICK	<p>ボタンがクリックされた。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラー</p>

	<p>ーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p> <p><i>isKeyboard</i>: イベントがキーボード/ゲームパッドで生成されたときは true、イベントがマウスで生成されたときは false。</p> <p><i>isRepeat</i>: イベントが autoRepeating ボタンを押し続けることで生成されたときは true、ボタンが現在反復していないときは false。</p>
ButtonEvent.DRAG_OVER	<p>マウス カーソルが（マウスの左ボタンを押したままの状態で）、ボタン上にドラッグされました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p>
ButtonEvent.DRAG_OUT	<p>マウス カーソルが（マウスの左ボタンを押したままの状態で）、ボタンから離れました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p>
ButtonEvent.RELEASE_OUTS IDE	<p>マウス カーソルが、ボタンから離れ、マウスの左ボタンが、解放されました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p>

ActionScript コード スニペットがこれらのイベントのキャッチや処理に必要となります。以下の例ではボタンのクリックを処理する方法を示しています

```
:myButton.addEventListener( ButtonEvent.PRESS, onButtonPress );
    function onButtonPress( e:ButtonEvent ):void {
        // Do something
    }
```

一行目は 'myButton' という名前のボタンに、ButtonEvent.PRESS イベントのイベントリスナーを追加して、そのイベントが起きたときに、onButtonPress 関数を呼び出すように指示しています。イベントハンドラーに与えられている ButtonEvent パラメーター（例では'e'と名付けられている）は、そのイベントに関する情報を含みます。パラメーター中の Event のタイプは、リスナーを加えるときに使用されるタイプと同じクラスか、先祖である必要があります。

2.1.2 CheckBox



図 11: スキニング前の CheckBox

CheckBox (`scaleform.clik.controls.CheckBox`) は、クリックされたときに選択状態を切り替えるように設定された Button コンポーネントです。Checkboxes は `true/false` (ブール) 値の表示と変更に使用されます。これは機能的には ToggleButton と同じですが、意識することなく `toggle` プロパティを設定します。

2.1.2.1 ユーザーの操作

マウス、または同等のキーボード コントロールを使って CheckBox コンポーネントをクリックすると、連続してそのコンポーネントを選択/選択解除します。それ以外の点では、この CheckBox は Button と同じように動作します。

2.1.2.2 コンポーネントのセットアップ

CLIK CheckBox クラスを使用する MovieClip は以下の指定されたサブエレメントが必須です。オプションのエレメントである場合には、以下に表記されています:

- *textField* : (オプション) TextField タイプ。ボタンのラベルです。
- *focusIndicator* : (オプション) MovieClip タイプ。フォーカスが適用されたステートの表示に使用される別の MovieClip です。存在する場合、この MovieClip は "show" と "hide" の 2 つの指定されたフレームを備えている必要があります。

2.1.2.3 ステート

自身の `toggle` プロパティのために、CheckBox は選択状態を表す別のキーフレーム セットが必要となります。このようなステートには以下が含まれます:

- *up* またはデフォルトのステート
- マウス カーソルがコンポーネント上にあるとき、またはフォーカスが適用されているときの *over* ステート
- ボタンが押下されたときの *down* ステート
- *disabled* ステート
- *selected_up* またはデフォルトのステート
- マウス カーソルがコンポーネント上にあるとき、またはフォーカスが適用されているときの *selected_over* ステート

- ボタンが押下されたときの *selected_down* ステート
- *selected_disabled* ステート

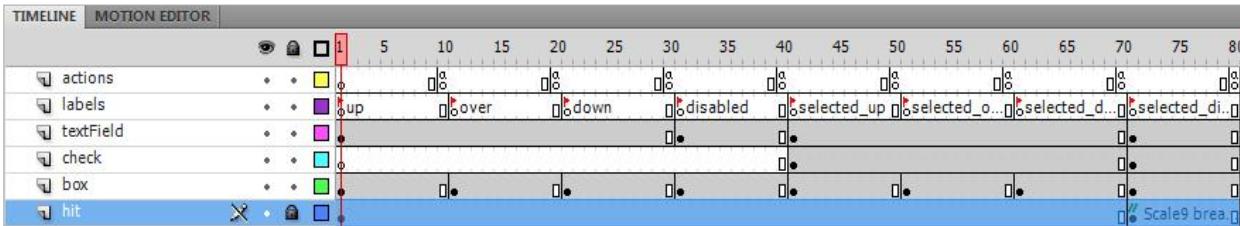
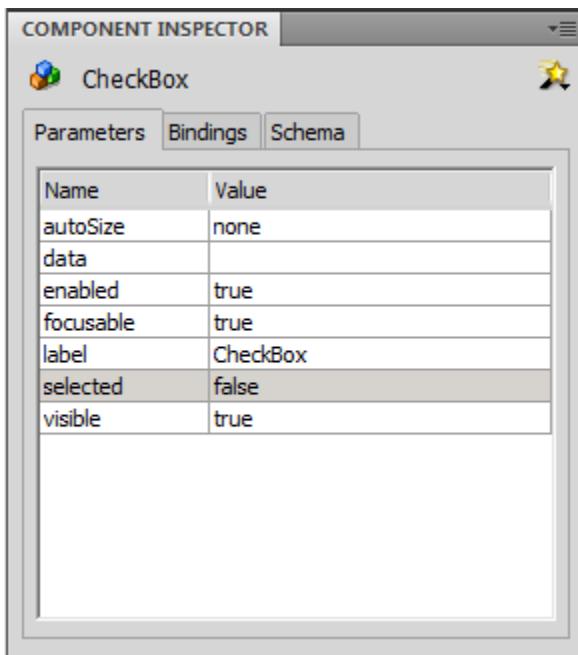


図 12: CheckBox のタイムライン

これは CheckBox に必要な最小限のキーフレーム セットです。Button コンポーネントがサポートするステートとキーフレームの拡張セットと、それに伴う CheckBox コンポーネントについては、「[Getting Started with CLIK Buttons](#)」を参照してください。

2.1.2.4 検証可能なプロパティ



CheckBox は Button コントロールから生じているので、このコンポーネントには Button と同じ検証可能なプロパティが含まれていますが、toggle と disableFocus プロパティは除外されます。

autoSize	ボタンが、その中のテキストサイズに合うように拡大縮小して、サイズの変わったボタンがどの方向に整列されるかを決定します。 autoSize プロパティを autoSize=TextFieldAutoSize.NONE に設定すると、その現在のサイズが変わりません。
Data	ボタン関連のデータ。このプロパティは、特に ButtonGroup でボタンを使用する場合に役に立ちます。
Enabled	false に設定しておくとボタンを無効にします。無効にされたコンポーネントはユーザーの入力を受け取りません。
Focusable	コンポーネントのフォーカス管理を有効化/無効化します。フォーカ

	ス可能なプロパティを <code>false</code> に設定すると Tab キー、方向キー、マウスボタンに基づいたフォーカス変更へのサポートを削除します。
Label	Button のラベルを設定します。
Selected	ボタンの選択されたステートを設定します。ボタンは選択された、非選択の 2 つのマウスステートを持ち得ます。ボタンの <code>toggle</code> プロパティが <code>true</code> の場合、選択されたステートはそのボタンがクリックされたときに変更されますが、 <code>toggle</code> プロパティが <code>false</code> でも ActionScript を用いて選択されたステートを設定可能です。
Visible	<code>false</code> に設定した場合、ボタンを非表示にします。

2.1.2.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベントサブクラス パラメーターを受け取ります。次のプロパティはすべてのイベントに対して共通です。

- **`type`**: イベントのタイプ。
- **`target`**: イベントのターゲット。
- **`currentTarget`**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **`eventPhase`**: イベントフローでの現在のフェーズ。（`EventPhase.CAPTURING_PHASE`、`EventPhase.AT_TARGET`、`EventPhase.BUBBLING_PHASE`）。
- **`bubbles`**: イベントがバーリングイベントかどうかを示す。
- **`cancelable`**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

CheckBox コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	visible プロパティが、実行時に <code>true</code> に設定されました。
ComponentEvent.HIDE	visible プロパティが、実行時に <code>false</code> に設定されました。
FocusHandlerEvent.FOCUS_IN	ボタンがフォーカスを受け取った。
FocusHandlerEvent.FOCUS_OUT	ボタンがフォーカスを失った。
Event.SELECT	選択されたプロパティが変更された。
ComponentEvent.STATE_CHANGE	ボタンのステートが、変更されました。
E	
MouseEvent.ROLL_OVER	マウス カーソルが、ボタン上に移動しました。 <code>mouseIdx</code> : イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleform のみ、イベントを <code>MouseEventEx</code> にキャストすることが必要。 <code>buttonIdx</code> : イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleform のみ、イベントを <code>MouseEventEx</code> にキャストすることが必要。
MouseEvent.ROLL_OUT	マウス カーソルが、ボタンから離れました。 <code>mouseIdx</code> : イベントの生成に使用されたマウスのカーソル

	<p>のインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleformのみ、イベントを MouseEventEx にキャストすることが必要。</p> <p><i>buttonIdx</i>: イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleformのみ、イベントを MouseEventEx にキャストすることが必要。</p>
ButtonEvent.PRESS	<p>ボタンが、押下されました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p> <p><i>isKeyboard</i>: イベントがキーボード/ゲームパッドで生成されたときは true、イベントがマウスで生成されたときは false。</p> <p><i>isRepeat</i>: イベントが autoRepeating ボタンを押し続けることで生成されたときは true、ボタンが現在反復していないときは false。</p>
MouseEvent.DOUBLE_CLICK	<p>ボタンが、ダブルクリックされました。</p> <p><i>doubleClickEnabled</i> プロパティが true の場合に限って発生します。</p> <p><i>mouseIdx</i>: イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleformのみ、イベントを MouseEventEx にキャストすることが必要。</p> <p><i>buttonIdx</i>: イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleformのみ、イベントを MouseEventEx にキャストすることが必要。</p>
ButtonEvent.CLICK	<p>ボタンが、クリックされました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p> <p><i>isKeyboard</i>: イベントがキーボード/ゲームパッドで生成されたときは true、イベントがマウスで生成されたときは false。</p> <p><i>isRepeat</i>: イベントが autoRepeating ボタンを押し続けることで生成されたときは true、ボタンが現在反復していないときは false。</p>
ButtonEvent.DRAG_OVER	<p>マウス カーソルが（マウスの左ボタンを押したままの状態で）、ボタン上にドラッグされました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p>
ButtonEvent.DRAG_OUT	<p>マウス カーソルが（マウスの左ボタンを押したままの状態で）、ボタンから離れました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該</p>

当)。ユニットタイプ。

ButtonEvent.RELEASE_OUTSIDE	マウス カーソルが、ボタンから離れ、マウスの左ボタンが解放されました。 <i>controllerIdx</i> :イベントの生成に使用されたコントローラーのインデックス (複数のマウスカーソル環境下でのみ該当)。ユニットタイプ。
------------------------------------	--

以下のサンプルコードは CheckBox の切り替え方法を示しています。

```
myCheckBox.addEventListener(Event.SELECT, onCheckBoxToggle);
function onCheckBoxToggle(e:Event):void {
    // Do something
}
```

2.1.3 Label



図 13: スキニング前の Label

CLIK Label コンポーネント (scaleform.clik.controls.Label) は簡単に言うと、便利な機能が少し追加された MovieClip シンボルでラップされた編集不可の標準の textField です。内部では、この Label は標準の textField と同じプロパティと動作をサポートしますが、このコンポーネント自体は一般的に使用されるほんの一握りの機能しか公開していません。ユーザーがプロパティを直接変更する必要がある場合、Label の実際の textField にアクセスすることができます。後述するような特定のケースでは、開発者はこの Label コンポーネントの代わりに標準の textField を使用することもできます。Label は MovieClip シンボルなので、標準の textField では不可能な、グラフィック エレメントで装飾することができます。シンボルなので、textField インスタンスのように、インスタンスごとに構成する必要はありません。また、Label はタイムラインで定義できる disabled ステートも提供します。一方で、標準の textField でこの動作を模倣するには、複雑な AS2 コードが必要となります。Label コンポーネントはデフォルトで制限があります。つまり、ステージで Label インスタンスのサイズを変更しても、実行時に目に見える効果はないということです。textField のサイズ変更が必要な場合、ほとんどのケースで開発者は、Label ではなく標準の textField を使用する必要があります。一般的に、テキスト エレメントに一貫した再利用性が必要ではない場合、標準の textField の方が Label コンポーネントよりも軽量です。

2.1.3.1 ユーザーの操作

Label ではユーザーの操作は一切できません。

2.1.3.2 コンポーネントのセットアップ

CLIK Label クラスを使用する MovieClip には、以下の指定されたサブエレメントが必須です。オプションのエレメントである場合には、その旨表記されています：

- **textField** : TextField タイプ。Label のテキストです。

2.1.3.3 ステート

CLIK Label コンポーネントは disabled プロパティに基づいた 2 つのステートをサポートします：

- *default* または有効のステート
- *disabled* ステート

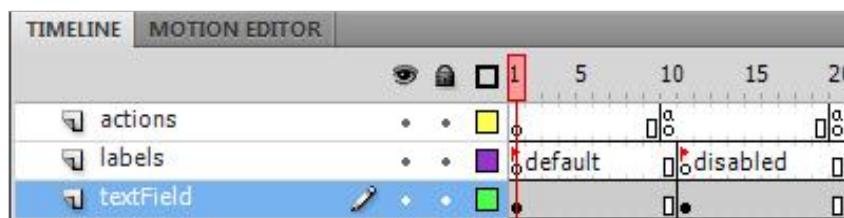
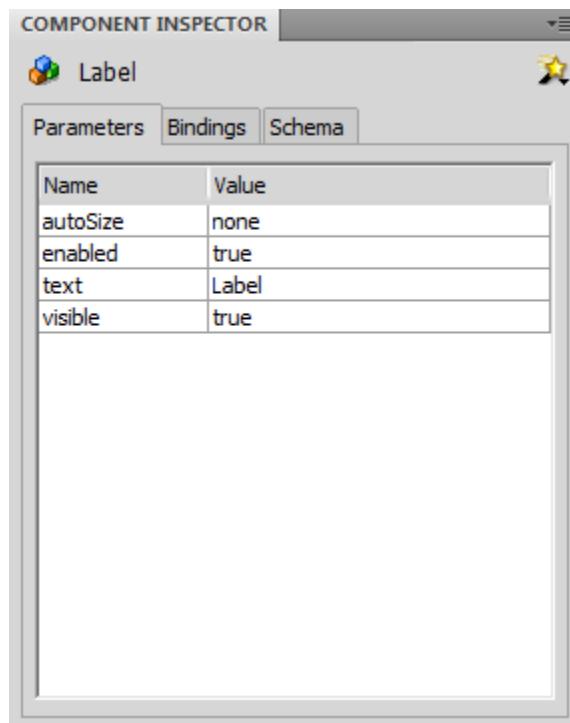


図 14: Label のタイムライン

2.1.3.4 検証可能なプロパティ



Label コンポーネントの検証可能なプロパティは以下のとおりです:

Text	ラベルのテキストを設定します。
visible	false に設定しておくとコンポーネントを非表示にします。
enabled	false に設定しておくとコンポーネントを無効にします。
autoSize	ラベルが、その中のテキストサイズに合うように拡大縮小して、サイズの変わったボタンがどの方向に整列されるかを決定します。autoSize プロパティを autoSize=TextFieldAutoSize.NONE に設定すると、その現在のサイズが変わりません。

2.1.3.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベント サブクラス パラメーターを受け取ります。次のプロパティはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE) 。
- **bubbles**: イベントがバーブリングイベントかどうかを示す。
- **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

Label コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティがランタイムで false に設定された。
ComponentEvent.STATE_CHANGE	コンポーネントのステートが変更された。

2.1.4 TextInput



図 15: スキンning前の TextInput

TextInput (scaleform.clik.controls.TextInput) は、テキストによるユーザー入力の取得に使用される編集可能な textField コンポーネントです。Label と同様に、このコンポーネントは標準の textField のラッパーに過ぎないので、パスワード モード、最大文字数、HTML テキストなど

`textField` 同じ機能をサポートします。このコンポーネント自体はこれらのプロパティのほんの一部しか公開していませんが、残りは直接 `TextInput` の `textField` インスタンスにアクセスすることで変更することができます。

`TextInput` コンポーネントは入力のために使用します。編集不可のテキストは `Label` を使って表示できるからです。`Label` と同様に、開発者は必要に応じて `TextInput` コンポーネントの代わりに標準の `textField` を使用することができます。ただし、特に PC アプリケーション向けの高度な UI を開発している場合、この `TextInput` コンポーネントは標準の `textField` に勝る価値のある拡張機能を提供します。

第一に、`TextInput` は `focused` と `disabled` ステートをサポートします。これらのステートは標準の `textField` では簡単に実現できません。フォーカス ステートが分けられているために、`TextInput` はカスタムのフォーカス インジケータをサポートすることができます。これは標準の `textField` には含まれていません。標準の `textField` の表示スタイルを変更するには複雑な AS2 コードが必要ですが、`TextInput` の表示スタイルはタイムライン上で簡単に構成することができます。`TextInput` の検証可能なプロパティは、Flash Studio に詳しくないデザイナーやプログラマに簡単なワークフローを提供します。また、開発者は `TextInput` が発生させるイベントを簡単にリッスンして、カスタムの動作を作成することもできます。

`TextInput` は複数段落の HTML フォーマット テキストも含め、`textField` が提供する標準の選択、さらにカット、コピー、ペースト機能もサポートします。デフォルトでは、キーボード コマンドは選択 (`Shift+矢印キー`)、カット (`Shift+Delete キー`)、コピー (`Ctrl+Insert キー`)、ペースト (`Shift+Insert`) です。

`TextInput` は、マウスカーソルのロールオーバーとロールアウト イベントのハイライト表示をサポート可能です。特別な `actAsButton inspectable` (検証可能な) プロパティが、このモードをサポートします。このモードは、これら二つのマウスイベントを再生する 2 個の追加キーフレームをサポートします。ここでの二つのキーフレームは、“over”と“out”と名付けます。これらは、それぞれ `rollOver` と `rollOut` ステートを表します。`actAsButton` モードがセットされても、“over”または“out”フレームが存在していない場合には、`TextInput` は“default”的キーフレームをどちらのイベントでも再生します。これらのフレームは、CLIK に用意してある `TextInput` コンポーネントには、デフォルトで存在していない、というポイントに留意してください。開発者の方が必要であれば、ご自分で追加していくことになります。

このコンポーネントは、また、全く値がセットされていない場合、あるいは、ユーザーが値を入力していない場合に、表示されるデフォルトのテキストをサポートしています。デフォルト `Text` プロパティは、いかなる `String` にもセット可能です。デフォルトテキストの色とスタイルのテーマは、明灰色 (0xAAAAAA) で、イタリックです。新たな `Text` フォーマット オブジェクトを、デフォルト `TextFormat` プロパティにアサインすることで、カスタムのスタイル設定を行います。

2.1.4.1 ユーザーの操作

`TextInput` をクリックするとこのコンポーネントにフォーカスが適用され、次にカーソルが `textField` に表示されます。カーソルが表示されると、ユーザーはキーボードや同等のコントローラで文字を入力することができます。左右の矢印キーを押すとカーソルが該当する方向に移動します。すでにカーソルが `textField` の左端にあるときに左の矢印キーが使用された場合、フォーカスは左側のコントロールに移動します。右矢印の場合も同じです。

2.1.4.2 コンポーネントのセットアップ

CLIK TextInput クラスを使用する MovieClip には、以下の指定されたサブエレメントが必須です。オプションのエレメントである場合には、その旨表記されています：

- **textField** : textField タイプ

2.1.4.3 ステート

CLIK TextInput コンポーネントは、その focused と disabled プロパティに基づいて、3 つのステートをサポートします：

- *default* または有効のステート
- *focused* ステート、通常、textField の周囲の強調表示された境界線で表現されます。
- *disabled* ステート

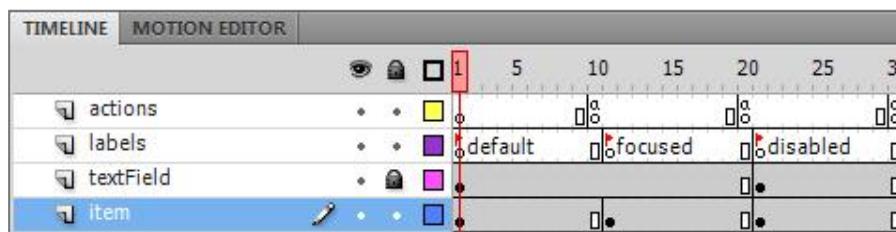
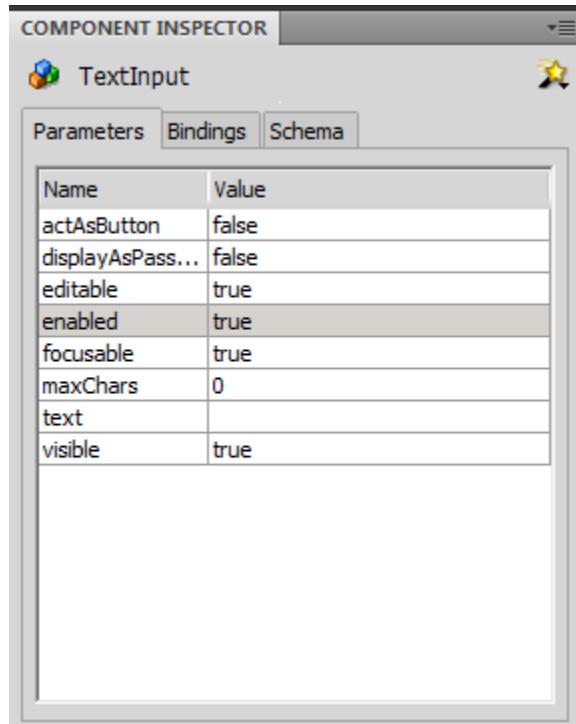


図 16: TextInput のタイムライン

2.1.4.4 検証可能なプロパティ



TextInputコンポーネントの検証可能なプロパティは以下のとおりです:

actAsButton	true の場合、TextInput は、フォーカスされていなければ、Button と同様に動作します。rollOver と rollOut ステートをサポートします。マウス クリックまたは、タブキーの押下によって、フォーカスされると、TextInput は、フォーカスを失うまで、ノーマルの状態に戻ります。
displayAsPassword	これを true に設定すると、textField を実際の文字の代わりに '*' 文字を表示するようにします。textField の値は、ユーザーが入力し、テキストプロパティーで返される実際の文字です。
Editable	false に設定した場合、TextInput が編集不可になります。
Enabled	false に設定しておくとコンポーネントを無効にします。
Focusable	コンポーネントのフォーカス管理を有効化/無効化します。フォーカス可能なプロパティーを false に設定すると Tab キー、方向キー、マウスボタンに基づいたフォーカス変更へのサポートを削除します。
maxChars	ゼロより大きな数値は、textField に入力できる文字数を制限します。
Text	textField の初期テキストを設定します。
Visible	false に設定した場合、このコンポーネントを非表示にします。

2.1.4.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベント サブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE)。
- **bubbles**: イベントがバーリングイベントかどうかを示す。
- **cancelable**: イベントに関連づけられているビヘイビューを回避できるかどうかを示す。

TextInputコンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。
FocusHandlerEvent.FOCUS_IN	このコンポーネントにフォーカスが、適用されました。
FocusHandlerEvent.FOCUS_OUT	このコンポーネントのフォーカスが、解除されました。
Event.CHANGE	textField の内容が、変更されました。
ComponentEvent.STATE_CHANGE	コンポーネントのステートが変更された。
MouseEvent.ROLL_OVER	マウスのカーソルがボタンの上に来た。 <i>mouseIdx</i> : イベントの生成に使用されたマウスのカーソ

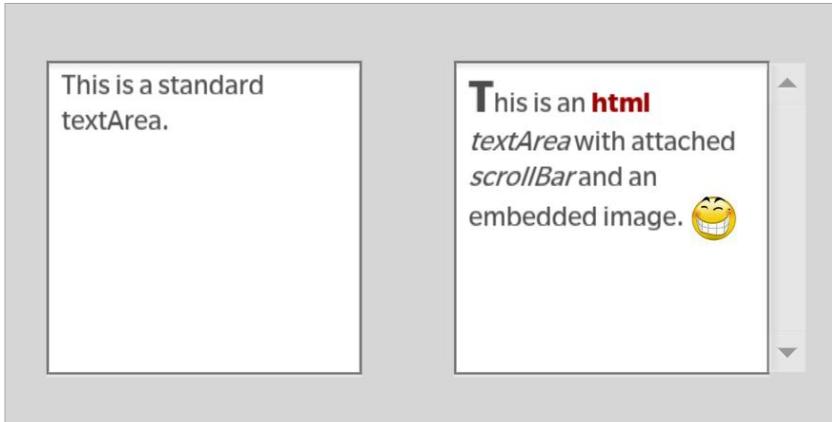
	ルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleformのみ、イベントをMouseEventExにキャストすることが必要。 <i>buttonIdx</i> :イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleformのみ、イベントをMouseEventExにキャストすることが必要。
MouseEvent.ROLL_OUT	マウスのカーソルがボタンの上から離れた。 <i>mouseIdx</i> :イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleformのみ、イベントをMouseEventExにキャストすることが必要。 <i>buttonIdx</i> :イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleformのみ、イベントをMouseEventExにキャストすることが必要。

以下のコード サンプルは textField の内容への変更をリッスンする方法を示しています:

```
myTextInput.addEventListener(Event.CHANGE, onTextChange);
function onTextChange(e:Event):void {
    trace("Latest text: " + e.target.text);
}
```

2.1.5 TextArea

図 17: スキンning前後の TextArea



TextArea (scaleform.clik.controls.TextArea) は CLIK TextInput から派生したものであり、同じ機能、プロパティ、ステートを共有していますが、オプションとして、複数行の編集可能で、スクロール可能なテキスト入力のための ScrollBar を備えています。TextInput と TextArea で共有される特別な機能について、TextInput の項での記述を参考にしてください。

Label や TextInput と同様に、TextArea は標準の複数行 textField のラッパーでもあるので、textField のプロパティや、HTML テキスト、テキストの折り返し、選択、カット、コピー、ペーストなどの動作もサポートします。開発者は自由に TextArea を標準の textField に置き換えることができますが、拡張機能、ステート、検証可能なプロパティ、イベント等を考慮すると、このコンポーネントを使用することを強くお勧めします。

標準の textField は ScrollIndicator や ScrollBar に組み合わせることができますが、TextArea はこれらのコンポーネントとより密接につながります。標準の textField とは異なり、編集不可であっても、フォーカスが適用されたときに TextArea はキーボードや同等のコントローラを使ってスクロールすることができます。スクロール関連のコンポーネントはフォーカスが適用できないので、TextArea は、自分自身とフォーカス ステートのスクロール コンポーネントの両方を包む、さらに洗練されたフォーカスが適用されたグラフィック ステートを表示することができます。

2.1.5.1 ユーザーの操作

TextArea をクリックするとこのコンポーネントにフォーカスが適用され、次にカーソルが textField に表示されます。カーソルが表示されると、ユーザーはキーボードや同等のコントローラで文字を入力することができます。4 つの矢印キーを押すとカーソルが該当する方向に移動します。カーソルがすでに端にあるときに矢印キーが使用された場合、フォーカスはその矢印キーの方向の隣接するコントロールに移動します。

2.1.5.2 コンポーネントのセットアップ

CLIK TextArea クラスを使用する MovieClip には、以下の指定されたサブエレメントが必須です。オプションのエレメントであれば、その旨表記されています:

- *textField* : TextField タイプ

2.1.5.3 ステート

その親コンポーネントである TextInput と同様に、TextArea コンポーネントは、その focused と disabled プロパティに基づいて、3 つのステートをサポートします:

- **default** または有効のステート
- **focused** ステート、通常、textField の周囲の強調表示された境界線で表現されます。
- **disabled** ステート

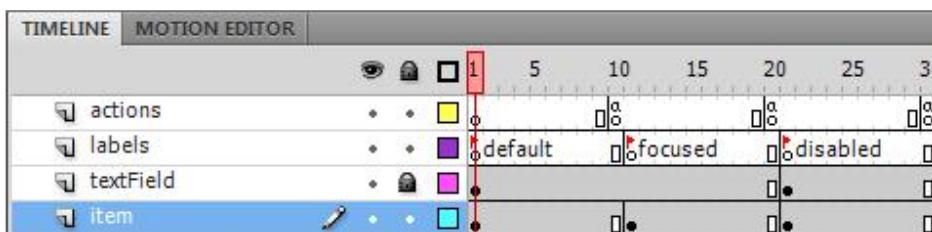
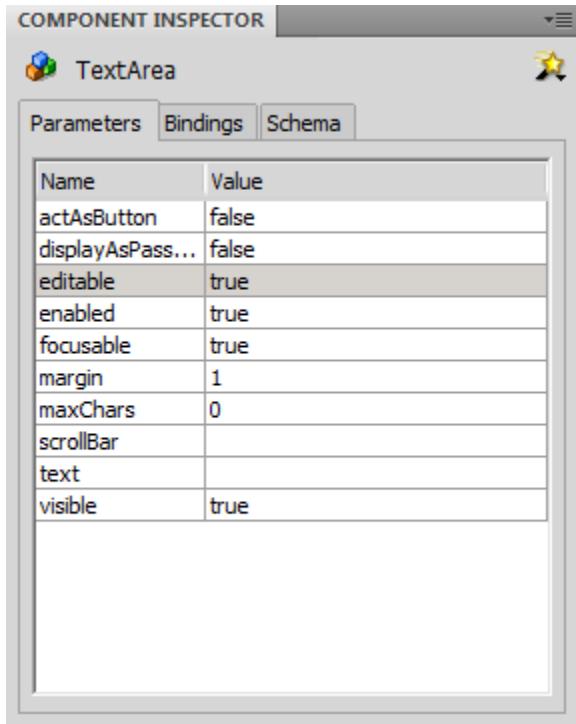


図 18: TextArea のタイムライン

2.1.5.4 検証可能なプロパティ



TextArea コンポーネントの検証可能なプロパティは TextInput に似ていますが、若干の追加があります。password プロパティは含まれていません。追加されたプロパティは CLIK ScrollBar コンポーネントに関係します。これは 2.4 章で説明します：

actAsButton	これを trueにしておくと、TextInput はフォーカスされていない場合には Button と同様に動作し、rollOver、rollOut ステートをサポートします。マウスを押す、または Tab の使用でフォーカスされると、TextInput はフォーカスを失うまで元の通常のモードに戻ります。
displayAsPassword	これを true に設定すると、textField を実際の文字の代わりに '*' 文字を表示するようにします。textField の値は、ユーザーが入力し、テキストプロパティーで返される実際の文字です。
Editable	false に設定しておくと TextInput の編集ができなくなります。
enabled	false に設定しておくとコンポーネントを無効にします。
focusable	コンポーネントのフォーカス管理を有効化/無効化します。フォーカス可能なプロパティーを false に設定すると Tab キー、方向キー、マウスボタンに基づいたフォーカス変更へのサポートを削除します。
maxChars	ゼロより大きい数にしておくと、textField に入れられる文字数が制限されます。
scrollBar	CLIK ScrollBar コンポーネントが使用するインスタンス名、または ScrollBar シンボルへのリンクエージ ID (この場合インスタンスは

scrollPolicy	TextArea が作成します) です。 "auto" に設定された場合、スクロールバーはスクロールする十分なテキストがある場合に限って表示されます。"on" に設定されると ScrollBar は常に表示され、"off" に設定されると表示されません。このプロパティは ScrollBar が割り当てられている場合に限って、そのコンポーネントに影響します (ScrollBar プロパティを参照してください)。
text	textField の初期テキストを設定します。
visible	false に設定しておくとコンポーネントを非表示にします。

2.1.5.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベントサブクラス パラメーターを受け取ります。次のプロパティはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE)。
- **bubbles**: イベントがバーブリングイベントかどうかを示す。
- **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

TextArea コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。
FocusHandlerEvent.FOCUS_IN	このコンポーネントにフォーカスが、適用されました。
FocusHandlerEvent.FOCUS_OUT	このコンポーネントのフォーカスが、解除されました。
T	
Event.CHANGE	textField の内容が、変更されました。
Event.SCROLL	テキスト領域が、スクロールされました。
ComponentEvent.STATE_CHANGE	コンポーネントのステートが変更された。
GE	
MouseEvent.ROLL_OVER	マウスのカーソルがボタンの上に来た。 <i>mouseIdx</i> : イベントの生成に使用されたマウスのカーソルのインデックス (複数のマウスカーソル環境下でのみ該当)。 ユニットタイプ。 Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。 <i>buttonIdx</i> : イベントがどのボタンに対して生成されたかを示します (ゼロ基準のインデックス)。 Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。
MouseEvent.ROLL_OUT	マウスのカーソルがボタンの上から離れた。 <i>mouseIdx</i> : イベントの生成に使用されたマウスのカーソルのインデックス (複数のマウスカーソル環境下でのみ該当)。

ユニットタイプ。Scaleformのみ、イベントを MouseEventEx にキャストすることが必要。

buttonIdx: イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleformのみ、イベントを MouseEventEx にキャストすることが必要。

以下の例は TextArea のスクロールイベントをリッスンする方法を示しています:

```
myTextArea.addEventListener(Event.SCROLL, onTextScroll);
function onTextScroll(e:Event):void {
    // Do something
}
```

2.2 グループ タイプ

グループ タイプには、RadioButton、ButtonGroup、そして ButtonBar コンポーネントが含まれています。ButtonGroup は、Button コンポーネントのグループを管理する特殊なロジックを備えたマネージャ タイプです。これは視覚的な表現を持たず、ステージ上には存在しません。ただし、ButtonBar はステージ上にあり、Button のグループの管理も行います。RadioButton は自動的にその兄弟と ButtonGroup にグループ化される特殊な Button です。

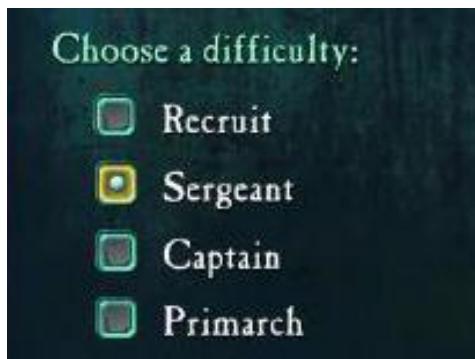


図 19: Dawn of War II - ラジオ ボタン グループの例

2.2.1 RadioButton



図 20: スキンning前の RadioButton

RadioButton (scaleform.clik.controls.RadioButton) は、通常 1 つの値を表示し変更するセットに属するボタン コンポーネントです。セット内の RadioButton は 1 つだけしか選択できず、そのセットの別の RadioButton をクリックするとその新規のコンポーネントが選択され、前回選択されていたコンポーネントの選択は解除されます。

CLIK RadioButton は CheckBox コンポーネントに非常に良く似ており、その機能、ステート、動作を共有します。大きな違いは RadioButton がグループ プロパティをサポートすることです。そのプロパティにカスタムの ButtonGroup (次の章を参照してください) を割り当てることができます。また、RadioButton は toggle プロパティを設定しません。切り替え (トグル) はそれを管理する ButtonGroup インスタンスが行うからです。

2.2.1.1 ユーザーの操作

マウス、または同等のコントロールを使って RadioButton コンポーネントをクリックすると、まだ選択されていなければ、連続してそのコンポーネントを選択します。RadioButton が選択されていて、たった今クリックされた別の RadioButton と同じ ButtonGroup に属している場合、最初の RadioButton が選択解除されます。それ以外の点では、この RadioButton は Button と同じように動作します。

2.2.1.2 コンポーネントのセットアップ

CLIK RadioButton クラスを使用する MovieClip には、以下の指定されたサブエレメントが必須です。オプションのエレメントであれば、その旨表記されています:

- **textField** : (オプション) TextField タイプ。ボタンのラベルです。
- **focusIndicator** : (オプション) MovieClip タイプ。フォーカスが適用されたステートの表示に使用される別の MovieClip です。存在する場合、この MovieClip は "show" と "hide" の 2 つの指定されたフレームを備えている必要があります。

2.2.1.3 ステート

RadioButton は選択したステートと選択解除されたステートを切り替えることができるので、CheckBox と同様に、最低でも以下のステートが必要となります:

- *up* またはデフォルトのステート
- マウス カーソルがコンポーネント上にあるとき、またはフォーカスが適用されているときの *over* ステート
- ボタンが押下されたときの *down* ステート
- *disabled* ステート
- *selected_up* またはデフォルトのステート
- マウス カーソルがコンポーネント上にあるとき、またはフォーカスが適用されているときの *selected_over* ステート
- ボタンが押下されたときの *selected_down* ステート
- *selected_disabled* ステート

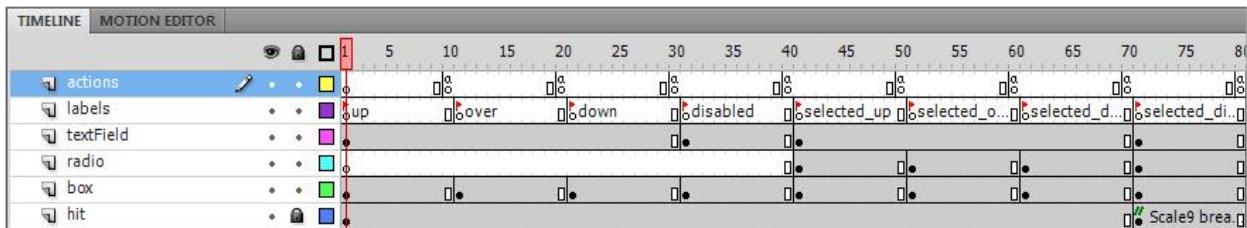
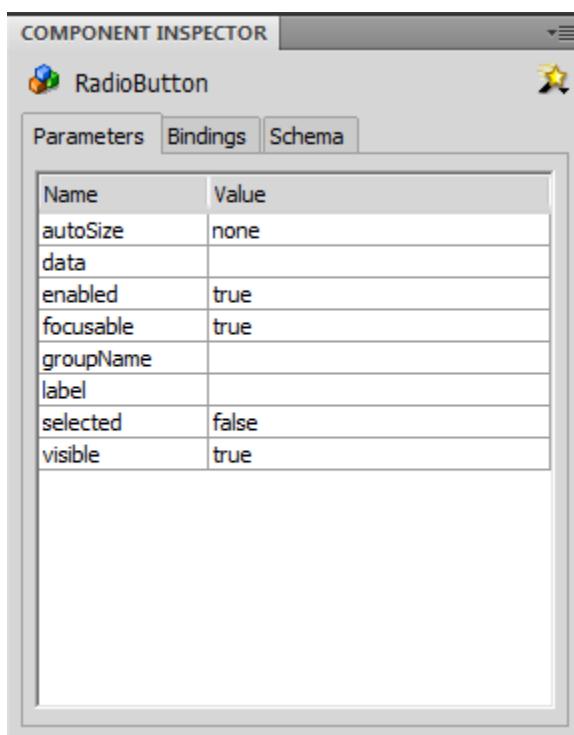


図 21: RadioButton のタイムライン

これは RadioButton に必要な最小限のキーフレーム セットです。Button コンポーネントがサポートするステートとキーフレームの拡張セット、それに伴う RadioButton コンポーネントについては、「[Getting Started with CLIK Buttons](#)」を参照してください。

2.2.1.4 検証可能なプロパティ



RadioButton は Button コントロールから派生したものなので、このコンポーネントには Button と同じ検証可能なプロパティが含まれていますが、toggle と disableFocus プロパティは除外されます。

autoSize	ボタンが、その中のテキストサイズに合うように拡大縮小して、サイズの変わったボタンがどの方向に整列されるかを決定します。 autoSize プロパティーを autoSize=TextFieldAutoSize.NONE に設定すると、その現在のサイズが変わりません。
data	ボタン関連のデータ。このプロパティーは、特に ButtonGroup でボタンを使用する場合に役に立ちます。
enabled	false に設定しておくとボタンを無効にします。無効にされたコンポ

	一ネントはユーザーの入力を受け取ません。
focusable	コンポーネントのフォーカス管理を有効化/無効化します。フォーカス可能なプロパティーを <code>false</code> に設定すると Tab キー、方向キー、マウスボタンに基づいたフォーカス変更へのサポートを削除します。
groupName	存在する、または RadioButton の生成すべき ButtonGroup インスタンスの名前。RadioButton が生成する場合、新しい ButtonGroup は RadioButton のコンテナーの内部に存在します。たとえば、RadioButton が <code>in_root</code> にある場合、その ButtonGroup オブジェクトは <code>in_root</code> に生成されます。同じグループを使用するすべての RadioButtons は 1 つのセットに属します。
label	ボタンのラベルを設定します。
selected	ボタンの選択されたステートを設定します。ボタンは選択された、非選択の 2 つのマウスステートを持ち得ます。ボタンの <code>toggle</code> プロパティーが <code>true</code> の場合、選択されたステートはそのボタンがクリックされたときに変更されますが、 <code>toggle</code> プロパティーが <code>false</code> でも ActionScript を用いて選択されたステートを設定可能です。

2.2.1.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベントサブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- **`type`**: イベントのタイプ。
- **`target`**: イベントのターゲット。
- **`currentTarget`**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **`eventPhase`**: イベントフローでの現在のフェーズ。 (`EventPhase.CAPTURING_PHASE`、`EventPhase.AT_TARGET`、`EventPhase.BUBBLING_PHASE`)。
- **`bubbles`**: イベントがバブルリングイベントかどうかを示す。
- **`cancelable`**: イベントに関連づけられているビヘイビューを回避できるかどうかを示す。

RadioButton コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティーがランタイムで <code>true</code> に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで <code>false</code> に設定された。
FocusHandlerEvent.FOCUS_IN	このコンポーネントにフォーカスが、適用されました。
FocusHandlerEvent.FOCUS_OUT	このコンポーネントのフォーカスが、解除されました。
Event.SELECT	選択されたプロパティーが変更された。
ComponentEvent.STATE_CHANGE	ボタンのステートが、変更されました。
MouseEvent.ROLL_OVER	マウス カーソルが、ボタン上に移動しました。 <code>mouseIdx</code> : イベントの生成に使用されたマウスのカ

	<p>ーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleformのみ、イベントを MouseEventEx にキャストすることが必要。</p> <p><i>buttonIdx</i>: イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。</p> <p>Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p>
MouseEvent.ROLL_OUT	<p>マウス カーソルが、ボタンから離れました。</p> <p><i>mouseIdx</i>: イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p> <p><i>buttonIdx</i>: イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。</p> <p>Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p>
ButtonEvent.PRESS	<p>ボタンが、押下されました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p> <p><i>isKeyboard</i>: イベントがキーボード/ゲームパッドで生成されたときは true、イベントがマウスで生成されたときは false。</p> <p><i>isRepeat</i>: イベントが autoRepeating ボタンを押し続けることで生成されたときは true、ボタンが現在反復していないときは false。</p>
MouseEvent.DOUBLE_CLICK	<p>ボタンが、ダブルクリックされました。</p> <p><i>doubleClickEnabled</i> プロパティが true の場合に限って発生します。</p> <p><i>mouseIdx</i>: イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p> <p><i>buttonIdx</i>: イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。</p> <p>Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p>
ButtonEvent.CLICK	<p>ボタンが、クリックされました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p> <p><i>isKeyboard</i>: イベントがキーボード/ゲームパッドで生成されたときは true、イベントがマウスで生成さ</p>

ButtonEvent.DRAG_OVER	れたときは false。 <i>isRepeat</i> : イベントが autoRepeating ボタンを押し続けることで生成されたときは true、ボタンが現在反復していないときは false。 マウス カーソルが (マウスの左ボタンを押したままの状態で)、ボタン上にドラッグされました。 <i>controllerIdx</i> : イベントの生成に使用されたコントローラーのインデックス (複数のマウスカーソル環境下でのみ該当)。ユニットタイプ。
ButtonEvent.DRAG_OUT	マウス カーソルが (マウスの左ボタンを押したままの状態で)、ボタンから離れました。 <i>controllerIdx</i> : イベントの生成に使用されたコントローラーのインデックス (複数のマウスカーソル環境下でのみ該当)。ユニットタイプ。
ButtonEvent.RELEASE_OUTSIDE	マウス カーソルが、ボタンから離れ、マウスの左ボタンが解放されました。 <i>controllerIdx</i> : イベントの生成に使用されたコントローラーのインデックス (複数のマウスカーソル環境下でのみ該当)。ユニットタイプ。

以下の例は RadioButton のトグルを処理する方法を示しています:

```
myRadio.addEventListener(Event.SELECT, onRadioToggle);
function onRadioToggle(e:Event):void {
    // Do something
}
```

2.2.2 ButtonGroup

図 22: スキンning前の ButtonGroup



CLIK ButtonGroup (scaleform.clik.controls.ButtonGroup) はそれ自体はコンポーネントではありませんが、それでも重要です。ボタン セットの管理に使用されるからです。ButtonGroup はセット内

のボタンを 1 つだけ選択させ、残りは未選択であることを保証します。ユーザーがセット内の別のボタンを選択した場合、現在選択されているボタンは選択解除されます。CLIK Button コンポーネント (CheckBox や RadioButton) から派生するコンポーネントはすべて、ButtonGroup インスタンスに割り当てることができます。

2.2.2.1 ユーザーの操作

ButtonGroup にはユーザーの操作はありません。視覚的な表示を持たないからです。ただし、そのグループに属する RadioButton がクリックされたときに、間接的な影響は受けます。

2.2.2.2 コンポーネントのセットアップ

CLIK ButtonGroup クラスを使用する MovieClip にはサブエレメントはいっさい必要ありません。視覚的な表示を持たないからです。

2.2.2.3 ステート

ButtonGroup はステージ上の視覚的な表示は備えていません。したがって、割り当てられるステートもありません。

2.2.2.4 検証可能なプロパティ

ButtonGroup はステージ上の視覚的な表示は備えていません。したがって、検証可能なプロパティは使用できません。

2.2.2.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベントサブクラス パラメーターを受け取ります。次のプロパティはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE)。
- **bubbles**: イベントがバーリングイベントかどうかを示す。
- **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

ButtonGroup コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

Event.CHANGE	グループの新規のボタンが、選択されました。
ButtonEvent.CLICK	グループ内のボタンが、クリックされました。 <i>target</i> : クリックされたボタンです。

以下の例は、ButtonGroup のどのボタンが選択されたかを判断する方法を示しています:

```
myGroup.addEventListener(Event.CHANGE, onNewSelection);
function onNewSelection(e:Event):void {
    if (e.item == myRadio) {
        // Do something
    }
}
```

2.2.3 ButtonBar

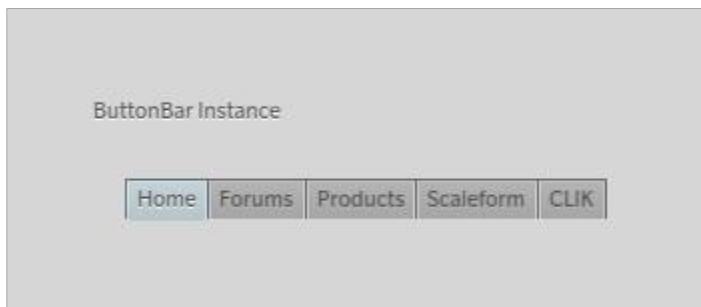


図 23: スキニング前の ButtonBar

ButtonBar (scaleform.clik.controls.ButtonBar) は ButtonGroup に似ていますが、視覚的な表示を備えています。また、dataProviderに基づいて、オンザフライで Button インスタンスを作成することができます (dataProviderについては、「[プログラミングの詳細](#)」を参照してください)。ButtonBar は、ダイナミックなタブバーのような UI エレメントの作成に便利です。このコンポーネントはdataProviderの割り当てによって設定されます:

```
buttonBar.dataProvider = new DataProvider(["item1", "item2", "item3", "item4",
"item5"]);
```

2.2.3.1 ユーザーの操作

ButtonBar は ButtonGroup と同じように動作します。ユーザーは直接そのバーを操作することはできませんが、ButtonBar は自らが管理する Button がクリックされたときに、間接的に影響も受けます。

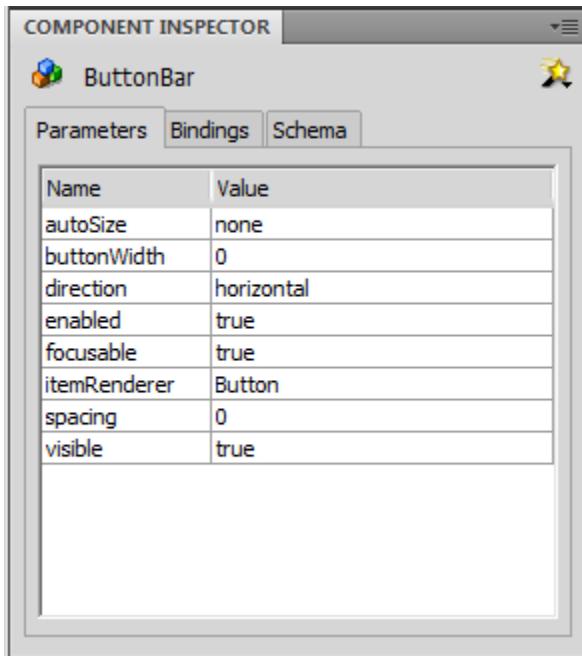
2.2.3.2 コンポーネントのセットアップ

CLIK ButtonBar クラスを使用する MovieClip にはサブエレメントはいっさい必要ありません。実行時にサブエレメントを作成するからです。

2.2.3.3 ステート

CLIK ButtonBar は視覚的なステートは備えていません。自らが管理する Button コンポーネントが、グループ ステートの表示に使用されるからです。

2.2.3.4 検証可能なプロパティ



ButtonBar コンポーネントにはコンテンツはありませんが (Flash IDE のステージ上では単に小さな丸として表示されます)、複数の検証可能なプロパティが含まれています。そのプロパティの大部分は、ButtonBar が作成した Button インスタンスの配置設定を処理します。

autoSize	これが true に設定されている場合、表示されているラベルに合うように Button インスタンスをサイズ変更します。
Direction	Button の配置。Horizontal は Button インスタンスを横に、vertical はこれらを上下に配置します。
buttonWidth	すべての Button インスタンスに共通の幅を設定します。autoSize が true に設定されている場合、このプロパティは無視されます。
enabled	false に設定しておくとボタンを無効にします。無効にされたコンポーネントはユーザーの入力を受け取りません。
focusable	コンポーネントのフォーカス管理を有効化/無効化します。フォーカス可能なプロパティーを false に設定すると Tab キー、方向キー、マウスボタンに基づいたフォーカス変更へのサポートを削除します。
itemRenderer	Button コンポーネントシンボルのリンクエージ ID です。このシンボルは ButtonBar に割り当てられたデータに基づいて、必要に応じてインスタンス化されます。
spacing	Button インスタンス間のスペースです。現在設定されている方向 (direction プロパティを参照してください) だけに影響します。
visible	これを false に設定しておくと ButtonBar を非表示にします。

2.2.3.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベント サブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- ***type***: イベントのタイプ。
- ***target***: イベントのターゲット。
- ***currentTarget***: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- ***eventPhase***: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE) 。
- ***bubbles***: イベントがバーリングイベントかどうかを示す。
- ***cancelable***: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

ButtonBar コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。
FocusHandlerEvent.FOCUS_IN	このコンポーネントにフォーカスが、適用されました。
FocusHandlerEvent.FOCUS_OUT	このコンポーネントのフォーカスが、解除されました。
IndexEvent.INDEX_CHANGE	<p>グループからの新しいボタンが選択された。</p> <p><i>index</i>: ButtonBar. int タイプの選択されたインデックス。 Values -1 (現在アイテムが選択されていない場合) からボタン数-1 まで。</p> <p><i>lastIndex</i>: ButtonBar. int タイプの以前に選択されたインデックス。 Values -1 (以前アイテムが選択されていなかった場合) からボタン数-1 まで。</p> <p><i>data</i>: 選択されたdataProvider のデータ値。 AS3 オブジェクト タイプ。</p>
ButtonBarEvent.BUTTON_SELECT	<p>グループからの新しいボタンが選択された。</p> <p><i>index</i>: ButtonBar の選択されたインデックス。 Int type。 Values -1 (現在アイテムが選択されていない場合) からボタン数-1 まで。</p> <p><i>renderer</i>: 現在選択されている ButotnBar 内の Button インスタンス。 Button タイプ。</p>

以下の例は、ButtonBar 内部で Button インスタンスがクリックされたときを検出する方法を示しています:

```
myBar.addEventListener(ButtonBarEvent.BUTTON_SELECT, onItemClick);
function onItemClick(e:ButtonBarEvent) {
    processData(e.renderer.data);
    // Do something
}
```

2.3 スクロール タイプ

スクロール タイプには ScrollIndicator、ScrollBar、Slider コンポーネントが含まれます。 ScrollIndicator は非インタラクティブで、単にサムを使ってターゲット コンポーネントのスクロールインデックスを表示するだけですが、 ScrollBar はスクロールの位置に影響するユーザーの操作を許可します。 ScrollBar は 4 つの Button (サムまたはツマミ、 トラック、 上下の矢印) で構成されています。 Slider は ScrollBar に似ていますが、 インタラクティブなサムと トラックしか含んでいません。 また、 ターゲット コンポーネント内のエレメント数に基づいたサムのサイズ変更は行いません。 Slider は ScrollBar に似ていますが、 インタラクティブなサムと トラックしか含んでいません。 また、 ターゲット コンポーネント内のエレメント数に基づいたサムのサイズ変更は行いません。

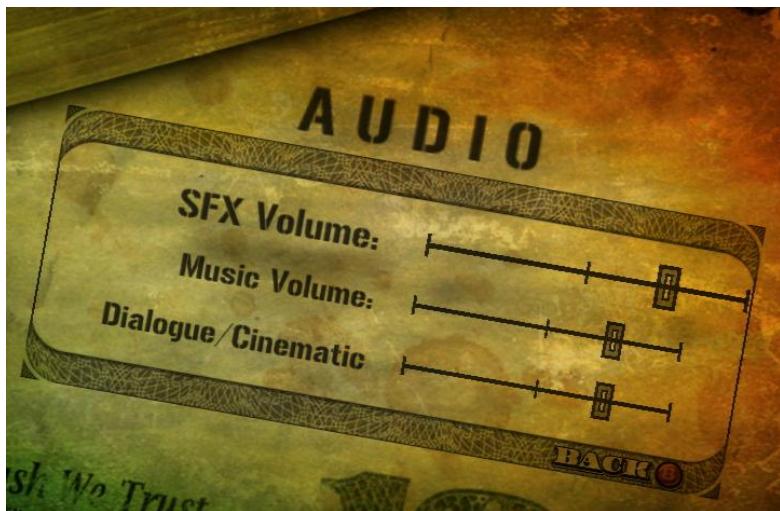


図 24: *Mercenaries 2* - オーディオ メニュー スライダの例

2.3.1 ScrollIndicator



図 25: スキンニング前の ScrollIndicator

CLIK ScrollIndicator (`scaleform.clik.controls.ScrollIndicator`) は、複数行の `textField` など、別のコンポーネントのスクロール位置を表示します。 ScrollIndicator は自動的にスクロール位置を指すように、 `textField` でポイントすることができます。 すべてのリストベースのコンポーネントは、 `TextArea` と同様に、 `ScrollIndicator` や `ScrollBar` (次の章を参照してください) インスタンス、あるいはリンクエージ ID を指す `scrollBar` プロパティを備えています。

2.3.1.1 ユーザーの操作

ScrollIndicator にはユーザーの操作はありません。表示のためだけに使用されます。

2.3.1.2 コンポーネントのセットアップ

CLIK ScrollIndicator クラスを使用する MovieClip には、以下の指定されたサブエレメントが必須です。オプションのエレメントであれば、その旨表記されています：

- *thumb* : CLIK Button のタイプ。スクロール インジケータのツマミです。
- *track* : MovieClip タイプ。スクロール インジケータのトラックです。このトラックの範囲は、ツマミが移動できる領域を決定します。

2.3.1.3 ステート

ScrollIndicator は明確なステートは備えていません。その子エレメントである thumb (サム) と track (トラック) の Button コンポーネントのステートを使用します。

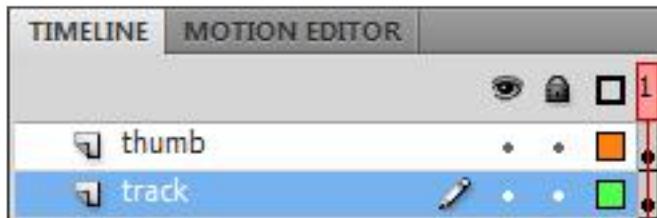
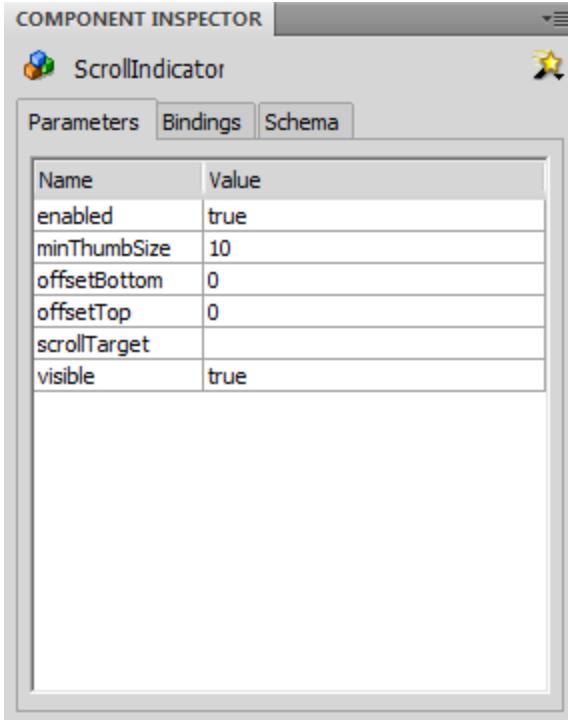


図 26: ScrollIndicator のタイムライン

2.3.1.4 検証可能なプロパティ



ScrollIndicator の検証可能なプロパティは以下のとおりです:

Enabled	false に設定しておくとコンポーネントを無効にします。無効にされたコンポーネントはユーザーの入力を受け取りません。
offsetTop	サム (thumb) を上端でオフセットします。正数値で、サムの最上端の位置を高くします。
offsetBottom	サム (thumb) を下端でオフセットします。正数値で、サムの最下端の位置を下げます。
scrollTarget	TextArea、または標準の複数行の textField をスクロール ターゲットとして設定し、自動的にスクロールのイベントに応答します。非 textField タイプは手動で ScrollIndicator のプロパティを更新する必要があります。
visible	false に設定した場合、このコンポーネントを非表示にします。

2.3.1.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベント サブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。

- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE、EventPhase.AT_TARGET、EventPhase.BUBBLING_PHASE)。
 - **bubbles**: イベントがバーピングイベントかどうかを示す。
 - **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。
- ScrollIndicator コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。
Event.SCROLL	ScrollIndicator がスクロールされた。

以下の例はスクロールイベントをリッスンする方法を示しています:

```
mySI.addEventListener(Event.SCROLL, onTextScroll);
function onTextScroll(e:Event) {
    trace("mySI.position: " + e.target.position);
    // Do something
}
```

2.3.2 ScrollBar



図 27: スキニング前の ScrollBar

CLIK ScrollBar (scaleform.clik.controls.ScrollBar) は、別のコンポーネントのスクロール位置を表示しコントロールします。ドラッグ可能なサム ボタンと、オプションの上下の矢印ボタン、さらにクリック可能なトラックと共に ScrollIndicator にインターラクティビティを追加します。

ユーザーの操作

ScrollBar のサムはマウスや同等のコントロールでつかみ、ScrollBar トラックの範囲をドラッグします。マウス カーソルが ScrollBar の上にあるときにマウス ホイールを動かすと、スクロール操作を行うことができます。上向き矢印をクリックするとサムは上に移動し、下向き矢印をクリックすると下に移動します。トラックのクリックには 2 つの動作があります。1 つはサムが継続してクリックした地点に向かってスクロールすることで、もう 1 つは直ちにその地点にジャンプして、ドラッグするよう設定されることです。このトラック モードは *trackMode inspectable* (検証可能な) プロパティが設定します (検証可能なプロパティの章を参照してください)。trackMode の設定に関係なく、Shift

キーを押しながらトラックをクリックすると、サムは直ちにそのカーソルまで移動して、ドラッグするように設定されます。

2.3.2.1 コンポーネントのセットアップ

CLIK ScrollBar クラスを使用する MovieClip には、以下の指定されたサブエレメントが必須です。オプションのエレメントであれば、表記されています：

- *upArrow* : CLIK Button のタイプです。スクロール アップする Button で、通常はスクロールバーの先頭に置かれます。
- *downArrow* : CLIK Button のタイプです。スクロール ダウンする Button で、通常はスクロールバーの最後に置かれます。
- *thumb* : CLIK Button のタイプ。スクロールバーのツマミです。
- *track* : CLIK Button のタイプです。ツマミの移動範囲を決める境界を持つスクロールバーのトラックです。

2.3.2.2 ステート

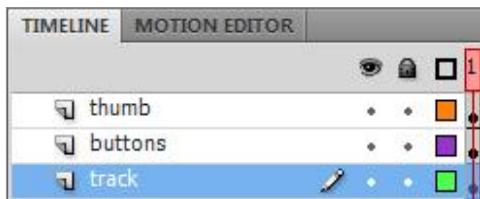
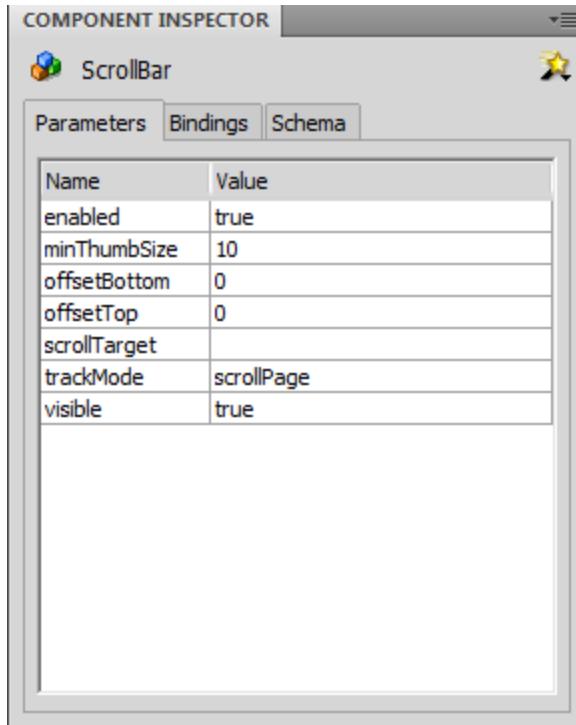


図 28: ScrollBar のタイムライン

ScrollIndicator と同様に、ScrollBar は明確なステートは備えていません。その子エレメントである thumb、up、down、track の Button コンポーネントのステートを使用します。

2.3.2.3 検証可能なプロパティ



ScrollBar の検証可能なプロパティは ScrollIndicator に似ていますが、1つ追加があります:

enabled	false に設定しておくとボタンを無効にします。無効にされたコンポーネントはユーザーの入力を受け取ません。
offsetTop	サム (thumb) を上端でオフセットします。正数値で、サムの最上端の位置を高くなります。
offsetBottom	サム (thumb) を下端でオフセットします。正数値で、サムの最下端の位置を下げます。
scrollTarget	TextArea、または標準の複数行の textField をスクロール ターゲットとして設定し、自動的にスクロールのイベントに応答します。非 textField タイプは手動で ScrollIndicator のプロパティを更新する必要があります。
trackMode	ユーザーがカーソルでトラックをクリックすると、カーソルが解放されるまで、scrollPage 設定によってサムはページ単位で継続してスクロールされます。scrollToCursor 設定によりサムは直ちにカーソルにジャンプし、さらにカーソルが解放されるまで、サムをドラッグ モードに移行します。
visible	false に設定した場合、このコンポーネントを非表示にします。

2.3.2.4 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベント サブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。

- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE、EventPhase.AT_TARGET、EventPhase.BUBBLING_PHASE)。
 - **bubbles**: イベントがバーピングイベントかどうかを示す。
 - **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。
- ScrollBar コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。
Event.SCROLL	ScrollIndicator がスクロールされた。

以下の例はスクロールイベントをリッスンする方法を示しています:

```
mySB.addEventListener(Event.SCROLL, onTextScroll);
function onTextScroll(e:Event) {
    trace("mySB.position: " + e.target.position);
    // Do something
}
```

2.3.3 Slider



図 29: スキンning前の Slider

Slider (scaleform.clik.controls.Slider) は範囲内の数値を表示します。その数値を表現するサムを含み、ドラッグによって変更します。

2.3.3.1 ユーザーの操作

Slider のサムはマウス等のコントロールで、Slider トラックの範囲をドラッグできます。トランクをクリックすると直ちにカーソルの位置に移動し、ドラッグするようになります。フォーカスが適用されている間、左右の矢印はサムを該当する方向に移動しますが、Home キーや End キーはサムをトランクの先頭や最後に移動します。

2.3.3.2 コンポーネントのセットアップ

CLIK Slider クラスを使用する MovieClip には、以下の指定されたサブエレメントが必要です。オプションのエレメントであれば、その旨表記されています:

- *thumb* : CLIK Button のタイプ。スライダのツマミ（サム）です。
- *track* : CLIK Button のタイプです。スライダ トラックの範囲はツマミ（サム）が移動できる領域を決定します。

2.3.3.3 ステート

ScrollIndicator や ScrollBar と同様に、Slider は明確なステートは備えていません。その子エレメントである thumb と track の Button コンポーネントのステートを使用します。

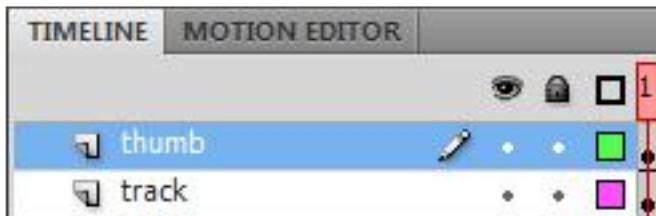
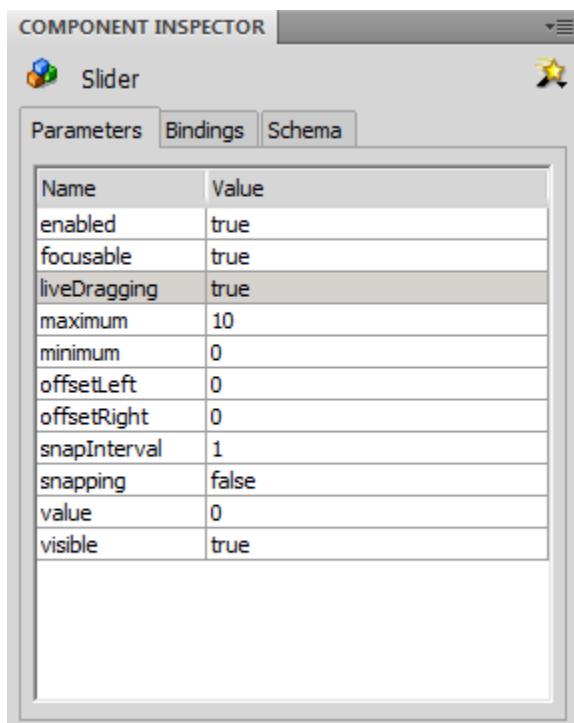


図 30: Slider のタイムライン

2.3.3.4 検証可能なプロパティ



Slider コンポーネントの検証可能なプロパティは以下のとおりです:

enabled	false に設定しておくとボタンを無効にします。無効にされたコンポーネントはユーザーの入力を受け取りません。
focusable	コンポーネントのフォーカス管理を有効化/無効化します。フォーカス可能なプロパティーを false に設定すると Tab キー、方向キー、マウスボタンに基づいたフォーカス変更へのサポートを削除します。
value	Slider で表示される数値です。
minimum	Slider の範囲の最小値です。

maximum	Slider の範囲の最大値です。
snapping	true に設定されると、サムは snapInterval の倍数の値にスナップします。
snapInterval	snapInterval はサムがスナップする値の倍数を決定します。snapping が false に設定されている場合は、効果はありません。
liveDragging	true に設定されると、Slider はサムをドラッグしているときに change イベントを作成します。false の場合、Slider はドラッグが終わるまで change イベントを作成しません。
offsetLeft	サム (thumb) を左端でオフセットします。正数値で、サムを内側に移動します。
offsetRight	サム (thumb) を右端でオフセットします。正数値で、サムを内側に移動します。
visible	false に設定しておくとコンポーネントを非表示にします。

2.3.3.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベントサブクラス パラメーターを受け取ります。次のプロパティはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE、EventPhase.AT_TARGET、EventPhase.BUBBLING_PHASE)。
- **bubbles**: イベントがバーブリングイベントかどうかを示す。
- **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

Slider コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティがランタイムで false に設定された。
ComponentEvent.STATE_CHANGE	コンポーネントのステートが変更された。
FocusHandlerEvent.FOCUS_IN	ボタンがフォーカスを受け取った。
FocusHandlerEvent.FOCUS_OUT	ボタンがフォーカスを失った。
SliderEvent.VALUE_CHANGE	Slider's 値が、変更されました。

以下の例は Slider 値の変更をリッスンする方法を示しています:

```
mySlider.addEventListener(SliderEvent.VALUE_CHANGE, onValueChange);
function onValueChange(e:SliderEvent) {
    trace("mySlider.value: " + e.target.value);
    // Do something
}
```

}

2.4 リスト タイプ

CLIK のリスト タイプには、NumericStepper、OptionStepper、ScrollingList、TileList、DropdownMenu コンポーネントが含まれます。NumericStepper を除き、これらのコンポーネントはすべて DataProvider と連携して、表示される情報を管理します。また、ListItemRenderer コンポーネントもこのカテゴリーに含まれます。ScrollingList や TileList コンポーネントがこれを使用して、リストの項目を表示するためです。

NumericStepper と OptionStepper は一度に 1 つのエレメントしか表示できませんが、ScrollingList と TileList は複数のエレメントを表示することができます。後者の 2 つのエレメントは ScrollIndicator または ScrollBar コンポーネントのいずれかをサポートすることができます。DropdownMenu コンポーネントは、アイドル状態では 1 つのエレメントしか表示しませんが、展開されると ScrollingList または TileList のいずれかを使ってさらに多くのエレメントを表示します。



図 31: *Mercenaries 2* - スクロール インジケータ付きのスクロール リストの例

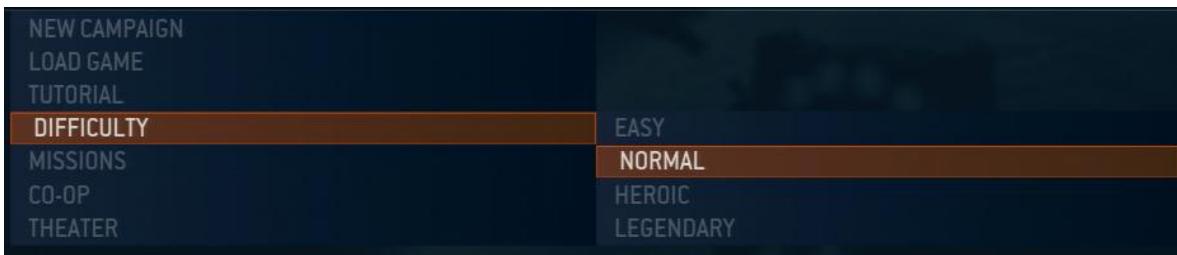


図 32: *Halo Wars* - 「難易度設定」 ドロップダウン メニューの例

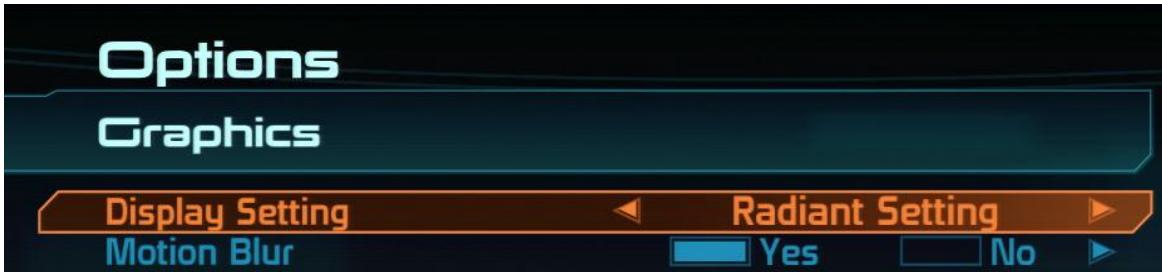


図 33: *Mass Effect* - 「表示設定」 オプション コントロールの例

2.4.1 NumericStepper



図 34: スキニング前の NumericStepper

NumericStepper (`scaleform.clik.controls.NumericStepper`) は、割り当てられた範囲内の 1 つの数値を表示し、任意の刻み幅単位で、その値を増減する機能をサポートします。

2.4.1.1 ユーザーの操作

NumericStepper には 2 つの矢印ボタンが含まれており、これらのボタンをマウスや同等のコントローラでクリックして、その数値を変更することができます。フォーカスが適用されると、この数値は左右の矢印キーや同等のコントロールを使って、キーボードで変更することもできます。これらのキーは刻み幅単位で現在の値を増減します。Home キーや End キー、または同等のコントロールを押すと、この数値がそれぞれ最小値と最大値に変更されます。

2.4.1.2 コンポーネントのセットアップ

NumericStepper クラスを使用する MovieClip には、以下の指定されたサブエレメントが必須です。オプションのエレメントであれば、その旨表記されています：

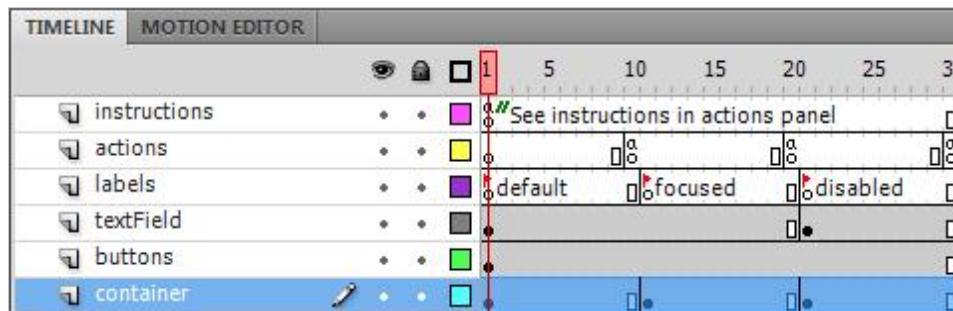
- *textField* : TextField タイプ。現在の値の表示に使用されます。
- *nextBtn* : CLIK Button のタイプです。これをクリックすると、刻み幅単位で現在の値を増加させます。
- *prevBtn* : CLIK Button のタイプです。これをクリックすると、刻み幅単位で現在の値を減少させます。

2.4.1.3 ステート

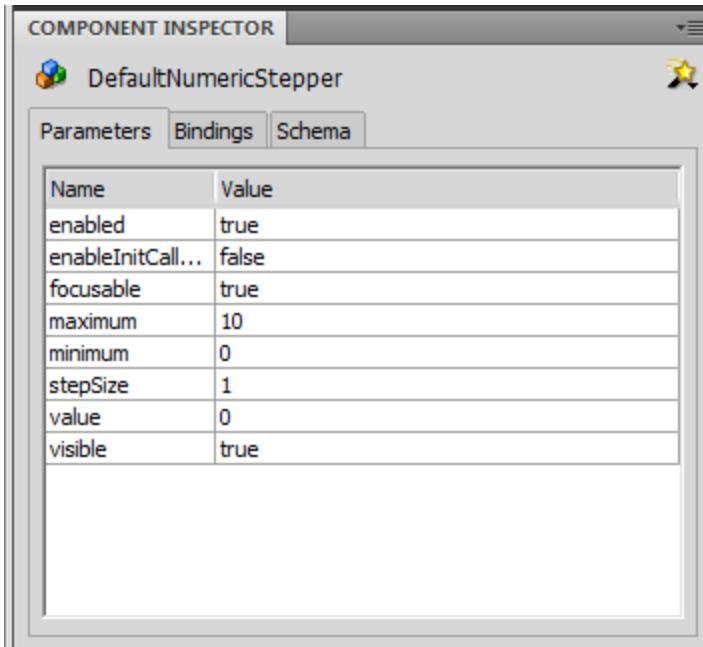
NumericStepper コンポーネントは、その *focused* と *disabled* プロパティに基づいて、3 つのステートをサポートします：

- *default* または有効のステート
- *focused* ステート、*textField* 領域を強調表示します。
- *disabled* ステート

図 35: NumericStepper のタイムライン



2.4.1.4 検証可能なプロパティ



NumericStepper コンポーネントから生じる MovieClip は、以下の検証可能なプロパティを備えています：

enabled	false に設定しておくとボタンを無効にします。無効にされたコンポーネントはユーザーの入力を受け取りません。
focusable	コンポーネントのフォーカス管理を有効化/無効化します。フォーカス可能なプロパティーを false に設定すると Tab キー、方向キー、マウスボタンに基づいたフォーカス変更へのサポートを削除します。
minimum	NumericStepper の範囲の最小値です。
maximum	NumericStepper の範囲の最大値です。
stepSize	NumericStepper の Button の 1 つがクリックされるごとに NumericStepper の値がどれだけ増加/減少させられるべきか。
Value	NumericStepper で表示される数値です。
visible	false に設定しておくとコンポーネントを非表示にします。

2.4.1.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベントサブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE)。
- **bubbles**: イベントがバーブリングイベントかどうかを示す。
- **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

NumericStepper コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。
ComponentEvent.STATE_CHANGE	コンポーネントのステートが変更された。
FocusHandlerEvent.FOCUS_IN	コンポーネントがフォーカスを受け取った。
FocusHandlerEvent.FOCUS_OUT	コンポーネントがフォーカスを失った。
ComponentEvent.STATE_CHANGE	コンポーネントのステートが変更された。
IndexEvent.INDEX_CHANGE	NumericStepper の変化した値。 <i>index</i> : NumericStepper. int タイプの選択されたインデックス。Values -1 (現在アイテムが選択されていない場合) からボタン数-1 まで。 <i>lastIndex</i> : NumericStepper. int タイプの以前に選択されたインデックス。Values -1 (以前アイテムが選択されていなかった場合) からボタン数-1 まで。 <i>data</i> : 選択されたdataProvider のデータ値。AS3 オブジェクトタイプ。

以下の例は NumericStepper 値の変更をリッスンする方法を示しています:

```
myNS.addEventListener(IndexEvent.INDEX_CHANGE, onValueChange);
function onValueChange(e:IndexEvent) {
    trace("myNS.value: " + e.target.value);
    // Do something
}
```

2.4.2 OptionStepper



図 36: スキニング前の OptionStepper

NumericStepper と同様に、OptionStepper (scaleform.clik.controls.OptionStepper) は 1 つの数値を表示しますが、数字以外も表示することができます。このコンポーネントはdataProvider インスタンスを使って現在の値を照会するので、さまざまなタイプの任意数のエレメントをサポートすることができます。表示される数値は、OptionStepper の selectedIndex プロパティを使ったコードで設定します。このプロパティは、付属しているdataProviderへの 0-ベースのインデックスです。以下の例のようにdataProvider はコードを使って割り当てられます:

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

2.4.2.1 ユーザーの操作

NumericStepper のように、OptionStepper には 2 つの矢印ボタンが含まれており、これらのボタンをマウスや同等のコントローラでクリックして、その現在の値を変更することができます。フォーカスが適用されると、現在の値は左右の矢印キーや同等のコントロールを使って、キーボードで変更することもできます。これらのキーは現在の値を前後の値に変更します。Home キーや End キー、または同等のコントロールを押すと、現在の値がdataProvider の最初と最後のエレメントに変更されます。

2.4.2.2 コンポーネントのセットアップ

NumericStepper クラスを使用する MovieClip には、以下の指定されたサブエレメントが必須です。オプションのエレメントであれば、その旨表記されています：

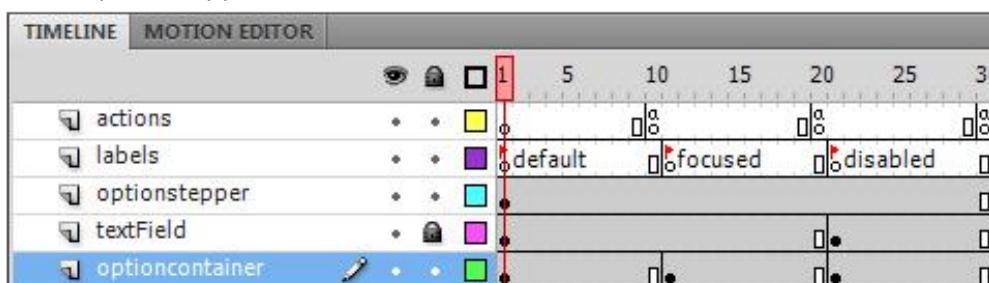
- *textField* : TextField タイプ。現在の値の表示に使用されます。
- *nextBtn* : CLIK Button のタイプです。現在の値をdataProvider の次のエレメントに変更します。
- *prevBtn* : CLIK Button のタイプです。現在の値をdataProvider の前のエレメントに変更します。

2.4.2.3 ステート

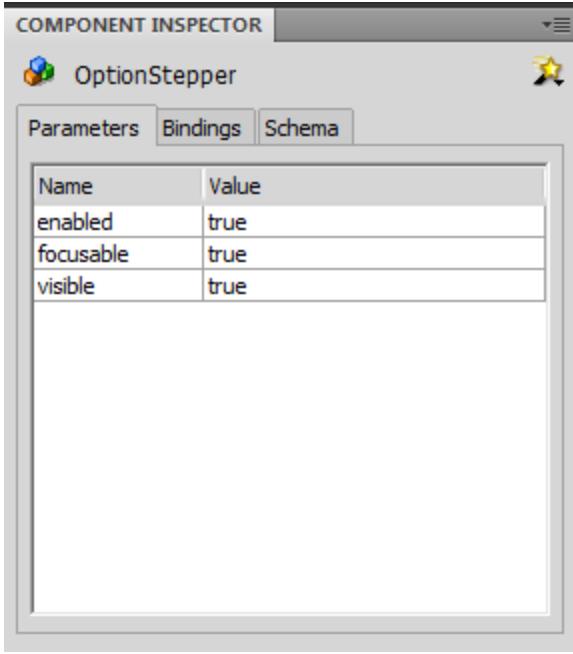
OptionStepper コンポーネントは、その *focused* と *disabled* プロパティに基づいて、3 つのステートをサポートします：

- *default* または有効のステート
- *focused* ステート、*textField* 領域を強調表示します。
- *disabled* ステート

図 37: OptionStepper のタイムライン



2.4.2.4 検証可能なプロパティ



OptionStepper コンポーネントから生じる MovieClip は、以下の検証可能なプロパティを備えています：

enabled	false に設定しておくとボタンを無効にします。無効にされたコンポーネントはユーザーの入力を受け取りません。
focusable	コンポーネントのフォーカス管理を有効化/無効化します。フォーカス可能なプロパティーを false に設定すると Tab キー、方向キー、マウスボタンに基づいたフォーカス変更へのサポートを削除します。
visible	false に設定しておくとコンポーネントを非表示にします。

2.4.2.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベント サブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE)。
- **bubbles**: イベントがバーリングイベントかどうかを示す。
- **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

OptionStepper コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW

可視プロパティーがランタイムで true に設定された。

ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。
ComponentEvent.STATE_CHANGE	コンポーネントのステートが変更された。
FocusHandlerEvent.FOCUS_IN	コンポーネントがフォーカスを受け取った。
FocusHandlerEvent.FOCUS_OUT	コンポーネントがフォーカスを失った。
IndexEvent.INDEX_CHANGE	OptionStepper の変化した値。 <i>index</i> : NumericStepper. int タイプの選択されたインデックス。Values -1 (現在アイテムが選択されていない場合) からボタン数-1 まで。 <i>lastIndex</i> : NumericStepper. int タイプの以前に選択されたインデックス。Values -1 (以前アイテムが選択されていなかった場合) からボタン数-1 まで。 <i>data</i> : 選択されたdataProvider のデータ値。AS3 オブジェクトタイプ。

以下の例は OptionStepper 値の変更をリッスンする方法を示しています:

```
myOS.addEventListener(IndexEvent.INDEX_CHANGE, onChange);
function onChange(e:IndexEvent) {
    trace("myOS.selectedItem: " + e.target.selectedItem);
    // Do something
}
```

2.4.3 ListItemRenderer



図 38: スキンning前の ListItemRenderer

ListItemRenderer (scaleform.clik.controls.ListItemRenderer) は CLIK Button クラスから派生したものであり、このクラスを拡張して、そのコンテナ コンポーネントに便利なリスト関連プロパティを含めます。しかし、これはスタンドアローン コンポーネントとして設計されたものではなく、ScrollingList、TileList、DropdownMenu コンポーネントと組み合わせて使用されるだけです。

2.4.3.1 ユーザーの操作

ListItemRenderer は Button コンポーネントから派生したものなので、マウスを使って押すなど Button と似たユーザーの操作を備えています。マウス カーソルを ListItemRenderer 上に置いたり、そこから離すときもコンポーネントに影響し、さらにマウス カーソルをその上にドラッグイン／ドラッグアウトするときも同様です。キーボードや同等のコントローラ操作は、この ListItemRenderer のコンテナ コンポーネントで定義されます。

2.4.3.2 コンポーネントのセットアップ

CLIK ListItemRenderer クラスを使用する MovieClip には、以下の指定されたサブエレメントが必須です。オプションのエレメントである場合はその旨表記されています:

- *textField* : (オプション) TextField タイプ。リスト アイテムのラベルです。
- *focusIndicator* : (オプション) MovieClip タイプ。フォーカスが適用されたステートの表示に使用される別の MovieClip です。存在する場合、この MovieClip は "show" と "hide" の 2 つの指定されたフレームを備えている必要があります。

2.4.3.3 ステート

ListItemRenderer はコンテナ コンポーネント内で選択できるので、選択状態を表すためにキーフレームの *selected* セットが必要となります。このコンポーネントのステートは以下のとおりです:

- *up* またはデフォルトのステート
- マウス カーソルがコンポーネント上にあるとき、またはフォーカスが適用されているときの *over* ステート
- ボタンが押下されたときの *down* ステート
- *disabled* ステート
- *selected_up* またはデフォルトのステート
- マウス カーソルがコンポーネント上にあるとき、またはフォーカスが適用されているときの *selected_over* ステート
- ボタンが押下されたときの *selected_down* ステート
- *selected_disabled* ステート

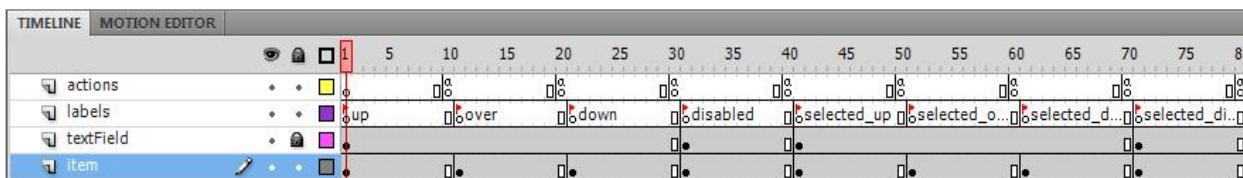
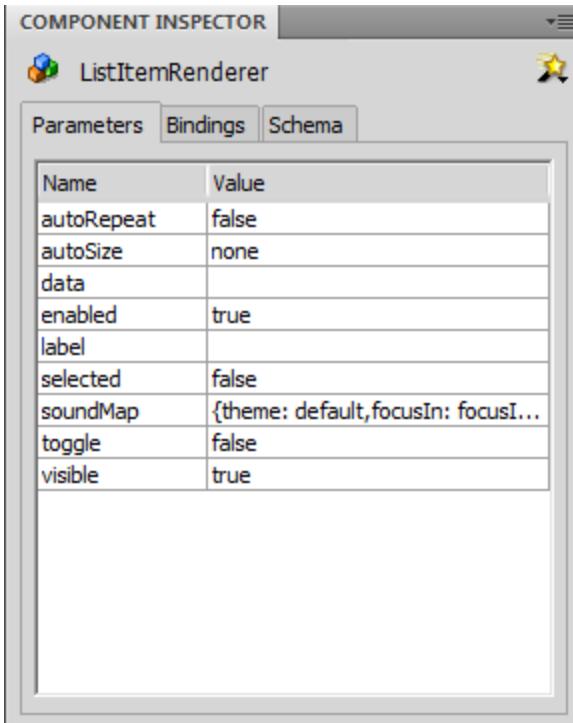


図 39: ListItemRenderer のタイムライン

これは ListItemRenderer に必要な最小限のキーフレーム セットです。Button コンポーネントがサポートするステートとキーフレームの拡張セットと、それに伴う ListItemRenderer コンポーネントについては、「[Getting Started with CLIK Buttons](#)」を参照してください。

2.4.3.4 検証可能なプロパティ



ListItemRenderer はコンテナ コンポーネントが制御し、ユーザーが手動で構成することはないので、少數の Button の検証可能プロパティのサブセットしか含まれていません。

autoRepeat	ListItemRender が押し続けられたときに「クリック」イベントを出すかどうかを決定します。
autoSize	ListItemRender が、その中のテキストサイズに合うように拡大縮小して、サイズの変わったボタンがどの方向に整列されるかを決定します。autoSize プロパティーを autoSize=TextFieldAutoSize.NONE に設定すると、その現在のサイズが変わりません。
Data	ListItemRender 関連のデータ。このプロパティーは、特に List component (ScrollingList / TileList) で ListItemRender を使用する場合に役に立ちます。
enabled	false に設定しておくとボタンを無効にします。無効にされたコンポーネントはユーザーの入力を受け取りません。
Label	ListItemRender のラベルを設定します。
selected	ListItemRender の選択されたステートを設定します。ListItemRender は選択された、非選択の 2 つのマウスステートを持ち得ます。ListItemRender の toggle プロパティーが true の場合、選択されたステートはそのボタンがクリックされたときに変更されますが、toggle プロパティーが false でも ActionScript を用いて選択されたステートを設定可能です。
Toggle	ListItemRender の選択されたトグルプロパティーを設定します。これが true に設定されている場合、ListItemRender はトグルボタンとして動作します。
Visible	false に設定しておくとコンポーネントを非表示にします。

2.4.3.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベント サブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE)。
- **bubbles**: イベントがバーブリングイベントかどうかを示す。
- **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

ListItemRenderer コンポーネントの生成するイベントを以下に掲げます。イベントの隣に挙げられているプロパティーは共通のプロパティー以外のものです。

一般的に、ListItemRenderer の発するイベントをリスンするよりも、List そのものにイベントリスナーを追加するようにしてください。List はその子供の ListItemRenderer との相互作用が起こった場合に、ITEM_PRESS と ITEM_CLICK を含む ListEvents を発します。このためユーザーはマウスクリックイベント (ListEvent.ITEM_CLICK) に対して、List 内の各 ListItemRenderer に 1 つずつ EventListener を加えるのではなく、EventListener をこの List に加えることができます。

ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。
ComponentEvent.STATE_CHANGE	コンポーネントのステートが変更された。
FocusHandlerEvent.FOCUS_IN	コンポーネントがフォーカスを受け取った。
FocusHandlerEvent.FOCUS_OUT	コンポーネントがフォーカスを失った。
Event.SELECT	選択されたプロパティーが変更された。
MouseEvent.ROLL_OVER	マウス カーソルが、ボタン上に移動しました。 <i>mouseIdx</i> : イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。 <i>buttonIdx</i> : イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。
MouseEvent.ROLL_OUT	マウス カーソルが、ボタンから離れました。 <i>mouseIdx</i> : イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。 <i>buttonIdx</i> : イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleform の

	み、イベントを MouseEventEx にキャストすることが必要。
ButtonEvent.PRESS	<p>ボタンが、押下されました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p> <p><i>isKeyboard</i>: イベントがキーボード/ゲームパッドで生成されたときは true、イベントがマウスで生成されたときは false。</p> <p><i>isRepeat</i>: イベントが autoRepeating ボタンを押し続けることで生成されたときは true、ボタンが現在反復していないときは false。</p>
MouseEvent.DOUBLE_CLICK	<p>ボタンが、ダブルクリックされました。</p> <p><i>doubleClickEnabled</i> プロパティが true の場合に限って発生します。</p> <p><i>mouseIdx</i>: イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p> <p><i>buttonIdx</i>: イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。</p>
ButtonEvent.CLICK	<p>ボタンが、クリックされました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p> <p><i>isKeyboard</i>: イベントがキーボード/ゲームパッドで生成されたときは true、イベントがマウスで生成されたときは false。</p> <p><i>isRepeat</i>: イベントが autoRepeating ボタンを押し続けることで生成されたときは true、ボタンが現在反復していないときは false。</p>
ButtonEvent.DRAG_OVER	<p>マウス カーソルが（マウスの左ボタンを押したままの状態で）、ボタン上にドラッグされました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p>
ButtonEvent.DRAG_OUT	<p>マウス カーソルが（マウスの左ボタンを押したままの状態で）、ボタンから離れました。</p> <p><i>controllerIdx</i>: イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。</p>
ButtonEvent.RELEASE_OUTSIDE	マウス カーソルが、ボタンから離れ、マウスの左ボタ

ンが解放されました。
controllerIdx:イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。

2.4.4 ScrollingList

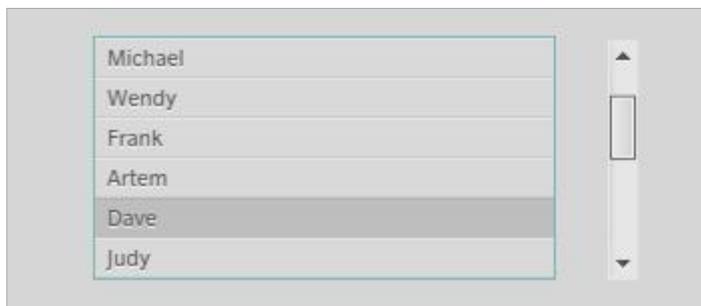


図 40: スキニング前の ScrollingList

ScrollingList (scaleform.clik.controls.ScrollingList) はそのエレメントをスクロールできるリストコンポーネントです。このコンポーネントは自分でリスト項目をインスタンス化したり、ステージ上の既存のリスト項目を使用することができます。また、ScrollIndicator や ScrollBar コンポーネントをこのリスト コンポーネントに追加して、スクロールのフィードバックや制御を行います。このコンポーネントはdataProvider で設定されます。以下の例のようにdataProvider はコードを使って割り当てられます：

```
scrollingList.dataProvider = new DataProvider(["item1", "item2", "item3",  
"item4"]);
```

デフォルトでは、ScrollingList はそのコンテンツのために ListItemRenderer コンポーネントを使用します。したがって、itemRenderer の検証可能プロパティが別のコンポーネントに変更されていない限り、ListItemRenderer は動作するために FLA ファイルの [ライブラリ] にも存在しなければなりません。詳細は「検証可能プロパティ」の章を参照してください。

2.4.4.1 ユーザーの操作

リスト項目、または添付されている ScrollBar インスタンスをクリックすると、フォーカスが ScrollingList コンポーネントに移動します。フォーカスが適用されている間、キーボードの上下の矢印や同等のコントロールを押すと、1 要素ごとにリストの選択範囲をスクロールします。何の要素も選択されていない場合は、最上部の要素が自動的にこの動作に選択されます。カーソルが ScrollingList の境界の先頭にある場合は、マウス ホイールはリストをスクロールします。

リストの境界でスクロールするという動作は、ScrollingList の *wrapping* プロパティで判断され、検証可能ではありません。*wrapping* が "normal" に設定されている場合、選択範囲がリストの先頭、または最後に達すると、フォーカスはそのコンポーネントを離れます。*wrapping* が "wrap" に設定されている場合、選択範囲は先頭か最後にまとめられます。*wrapping* が "stick" に設定されている場合、

データの最後に到達すると選択はそこで止まり、フォーカスは隣接するコンポーネントに移動しません。

キーボードの Page Up や Page Down キー、または同等のコントロールを押すと、ページ単位で、つまりリストに表示されている項目の数で選択範囲をスクロールします。Home キーや End キー、または同等のコントロールを押すと、リストをそれぞれ最初と最後のエレメントにスクロールします。添付されている ScrollBar コンポーネントを操作すると、予想通り ScrollingList に影響します。ScrollBar 独自のユーザーの操作については、ScrollBar の章を参照してください。

開発者は簡単にゲームパッドのコントロールを、キーボードやマウスのコントロールにマップすることができます。たとえば、キーボードの矢印キーは通常、コンソール コントローラの D パッドにマップされます。このマッピングによって CLIK で作成した UI は、各種のプラットフォームで動作することができます。

2.4.4.2 コンポーネントのセットアップ

ScrollingList には指定されたサブエレメントはいっさい必要ありません。ただし、ステージで ScrollingList コンポーネントのインスタンスの配置やサイズ変更を行うときに、目に見える背景は便利です。

2.4.4.3 ステート

ScrollingList コンポーネントは、その focused と disabled プロパティに基づいて、3 つのステートをサポートします：

- *default* または有効のステート
- *focused* ステート、通常コンポーネントの境界領域を強調表示します。
- *disabled* ステート

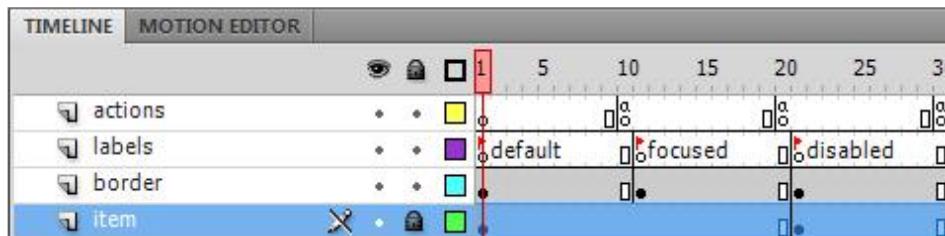
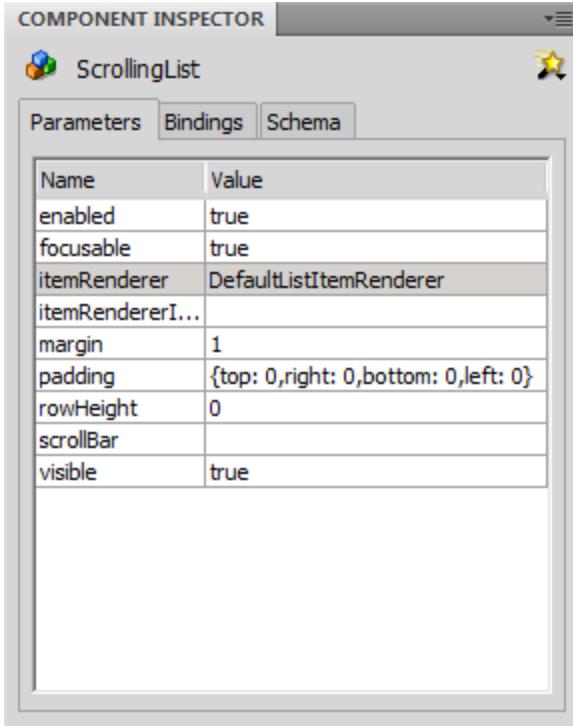


図 41: ScrollingList のタイムライン

2.4.4.4 検証可能なプロパティ



ScrollingList コンポーネントから生じる MovieClip は、以下の検証可能なプロパティを備えています：

enabled	false に設定しておくとボタンを無効にします。無効にされたコンポーネントはユーザーの入力を受け取りません。
focusable	コンポーネントのフォーカス管理を有効化/無効化します。フォーカス可能なプロパティーを false に設定すると Tab キー、方向キー、マウスボタンに基づいたフォーカス変更へのサポートを削除します。
itemRenderer	ListItemRenderer のシンボル名。内部でのリストアイテム インスタンスの生成に使用します。itemRendererInstanceName プロパティーがセットされている場合（つまり外部の ListItemRenderers を使用している場合）、なにも影響を与えません。
itemRendererInstanceName	この ScrollingList コンポーネントと併用するための、外部のリスト項目レンダラのプレフィックスです。ステージ上のリスト項目インスタンスは、このプロパティ値をプレフィックスとして付ける必要があります。このプロパティが 'r' という値に設定されていれば、このコンポーネントで使用されるすべてのリスト項目インスタンスは、'r1', 'r2', 'r3'… という値にする必要があります。最初の項目は 1 になります。
margin	リスト コンポーネントの境界と内部で作成されたリスト項目の間のマージンです。itemRendererInstanceName プロパティーがセットされている場合（つまり外部の ListItemRenderers を使用している場合）、なにも影響を与えません。
padding	リストアイテムに対するエキストラのパディング。

	itemRendererInstanceName プロパティーがセットされている場合（つまり外部の ListItemRenderers を使用している場合）、この値はなにも影響を与えません。パディングは自動的に生成された ScrollBar にはなにも影響を与えません。
rowHeight	内部で生成されたリストアイテム インスタンスの高さ。 itemRendererInstanceName プロパティーがセットされている場合（つまり外部の ListItemRenderers を使用している場合）、この値はなにも影響を与えません。
Visible	false に設定しておくとコンポーネントを非表示にします。これは添付された ScrollBar またはどのような外部の ListItemRenderers を非表示にしません。

2.4.4.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベント サブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- **type:** イベントのタイプ。
- **target:** イベントのターゲット。
- **currentTarget:** イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase:** イベントフローでの現在のフェーズ。（EventPhase.CAPTURING_PHASE、EventPhase.AT_TARGET、EventPhase.BUBBLING_PHASE）。
- **bubbles:** イベントがバーブリングイベントかどうかを示す。
- **cancelable:** イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

ScrollingList コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。
ComponentEvent.STATE_CHANGE	コンポーネントのステートが変更された。
FocusHandlerEvent.FOCUS_IN	コンポーネントがフォーカスを受け取った。
FocusHandlerEvent.FOCUS_OUT	コンポーネントがフォーカスを失った。
ListEvent.INDEX_CHANGE	選択されたインデックスが変更された。 <ul style="list-style-type: none"> • <i>itemRenderer:</i> ダブルクリックされたリストアイテム。 IListItemRenderer タイプ。 • <i>itemData:</i> リストアイテムに関連するデータ。この値はリストのdataProvider から取得されます。 ActionScript Object タイプ。 • <i>index:</i> list. int タイプの新たに選択されたインデックス。 Values -1 (selectedIndex はセットされていない) からリストアイテム数-1 まで。 • <i>controllerIdx:</i> イベントの生成に使用されたコントローラー/マウスのインデックス（複数のマウスカーソル環境下でのみ該当）。

ListEvent.ITEM_PRESS	リストアイテムが押された。 <ul style="list-style-type: none"> • <i>itemRenderer</i>:ダブルクリックされたリストアイテム。 IListItemRenderer タイプ。 • <i>itemData</i>:リストアイテムに関連するデータ。この値はリストのdataProvider から取得されます。ActionScript Object タイプ。 • <i>index</i>:list. int タイプの新たに選択されたインデックス。Values -1 (selectedIndex はセットされていない) からリストアイテム数-1 まで。 • <i>controllerIdx</i>:イベントの生成に使用されたコントローラー/マウスのインデックス (複数のマウスカーソル環境下でのみ該当)。
ListEvent.ITEM_CLICK	リストアイテムがクリックされた。 <ul style="list-style-type: none"> • <i>itemRenderer</i>:ダブルクリックされたリストアイテム。 IListItemRenderer タイプ。 • <i>itemData</i>:リストアイテムに関連するデータ。この値はリストのdataProvider から取得されます。ActionScript Object タイプ。 • <i>index</i>:list. int タイプの新たに選択されたインデックス。Values -1 (selectedIndex はセットされていない) からリストアイテム数-1 まで。 • <i>controllerIdx</i>:イベントの生成に使用されたコントローラー/マウスのインデックス (複数のマウスカーソル環境下でのみ該当)。
ListEvent.ITEM_DOUBLE_CLICK	リストアイテムがダブルクリックされた。 <ul style="list-style-type: none"> • <i>itemRenderer</i>:ダブルクリックされたリストアイテム。 IListItemRenderer タイプ。 • <i>itemData</i>:リストアイテムに関連するデータ。この値はリストのdataProvider から取得されます。ActionScript Object タイプ。 • <i>index</i>:list. int タイプの新たに選択されたインデックス。Values -1 (selectedIndex はセットされていない) からリストアイテム数-1 まで。 • <i>controllerIdx</i>:イベントの生成に使用されたコントローラー/マウスのインデックス (複数のマウスカーソル環境下でのみ該当)。
ListEvent.ITEM_ROLL_OVER	マウスのカーソルがリストアイテムの上に来た。 <ul style="list-style-type: none"> • <i>itemRenderer</i>:ダブルクリックされたリストアイテム。 IListItemRenderer タイプ。 • <i>itemData</i>:リストアイテムに関連するデータ。この値はリストのdataProvider から取得されます。ActionScript Object タイプ。 • <i>index</i>:list. int タイプの新たに選択されたインデックス。Values -1 (selectedIndex はセットされていない) からリストアイテム数-1 まで。 • <i>controllerIdx</i>:イベントの生成に使用されたコン

ListEvent.ITEM_ROLL_OUT	<p>トローラー/マウスのインデックス（複数のマウスカーソル環境下でのみ該当）。</p> <p>マウスのカーソルがリストアイテムの上から離れた。</p> <ul style="list-style-type: none"> • <i>itemRenderer</i>: ダブルクリックされたリストアイテム。 <code>IListItemRenderer</code> タイプ。 • <i>itemData</i>: リストアイテムに関連するデータ。この値はリストの <code>dataProvider</code> から取得されます。 <code>ActionScript Object</code> タイプ。 • <i>index:list. int</i> タイプの新たに選択されたインデックス。 <code>Values -1</code> (<code>selectedIndex</code> はセットされていない) からリストアイテム数-1 まで。 • <i>controllerIdx</i>: イベントの生成に使用されたコントローラー/マウスのインデックス（複数のマウスカーソル環境下でのみ該当）。
--------------------------------	---

以下の例は、クリックされているリスト項目をリッスンする方法を示しています:

```
myList.addEventListener(ListEvent.ITEM_CLICK, onItemClick);
function onItemClick(e:ListEvent) {
    trace("ListItemRenderer Clicked: " + e.itemRenderer);
    // Do something
}
```

2.4.5 TileList



図 42: スキニング前の TileList

`TileList` (`scaleform.clik.controls.TileList`) は `ScrollingList` と同様に、そのエレメントをスクロールできるリスト コンポーネントです。このコンポーネントは自分でリスト項目をインスタンス化したり、ステージ上の既存のリスト項目を使用することができます。また、 `ScrollIndicator` や `ScrollBar` コンポーネントをこのリスト コンポーネントに追加して、スクロールのフィードバックや制御を行います。`TileList` と `ScrollingList` の違いは、`TileList` が同時に複数の行と列をサポートできることです。リスト項目の選択範囲は 4 つの基本方位すべてに移動できます。このコンポーネントは `dataProvider` で設定されます。以下の例のように `dataProvider` はコードを使って割り当てられます:

```
tileList.dataProvider = new DataProvider(["item1", "item2", "item3", "item4",
"item5"]);
```

デフォルトでは `TileList` はそのコンテンツのために `ListItemRenderer` コンポーネントを使用します。したがって、`itemRenderer` の検証可能プロパティが別のコンポーネントに変更されていない限り、

`ListItemRenderer` は動作するために FLA ファイルの [ライブラリ] にも存在しなければなりません。詳細は「検証可能プロパティ」の章を参照してください。

2.4.5.1 ユーザーの操作

リスト項目、または添付されている ScrollBar インスタンスをクリックすると、フォーカスが `TileList` コンポーネントに移動します。フォーカスが適用されている間、キーボードの上下の矢印や同等のコントロールを押すと、複数の行が含まれている場合は 1 要素ごとにリストの選択範囲を縦にスクロールします。左右の矢印、または同等のコントロールは、複数の列が含まれている場合は横にリストの選択範囲をスクロールします。`TileList` に複数の行と列が含まれている場合、4 つすべての矢印キー、または同等のコントロールを使って、リストの選択範囲を移動することができます。何の要素も選択されていない場合は、最上部の要素が自動的にこの動作に選択されます。カーソルが `TileList` の境界の先頭にある場合は、マウスホイールはリストをスクロールします。

キーボードの `Page Up` や `Page Down` キー、または同等のコントロールを押すと、ページ単位で、つまりリストに表示されている行数で選択範囲をスクロールします。`Home` キーや `End` キー、または同等のコントロールを押すと、リストをそれぞれ最初と最後のエレメントにスクロールします。添付されている ScrollBar コンポーネントを操作すると、予想通り `ScrollingList` に影響します。ScrollBar 独自のユーザーの操作については、ScrollBar の章を参照してください。

2.4.5.2 コンポーネントのセットアップ

`TileList` には指定されたサブエレメントはいっさい必要ありません。ただし、ステージで `TileList` コンポーネントのインスタンスの配置やサイズ変更を行うときに、目に見える背景は便利です。

2.4.5.3 ステート

`TileList` コンポーネントは、その `focused` と `disabled` プロパティに基づいて、3 つのステートをサポートします：

- `default` または有効のステート
- `focused` ステート、通常コンポーネントの境界領域を強調表示します。
- `disabled` ステート

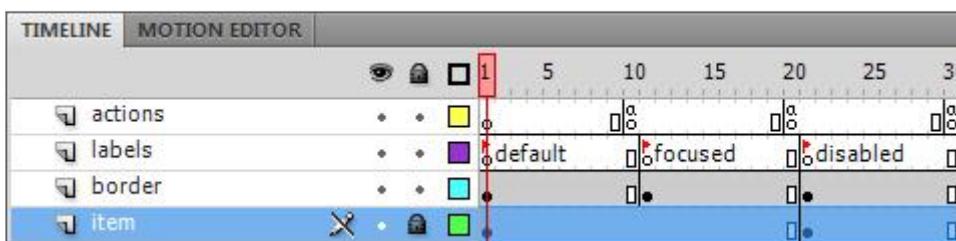
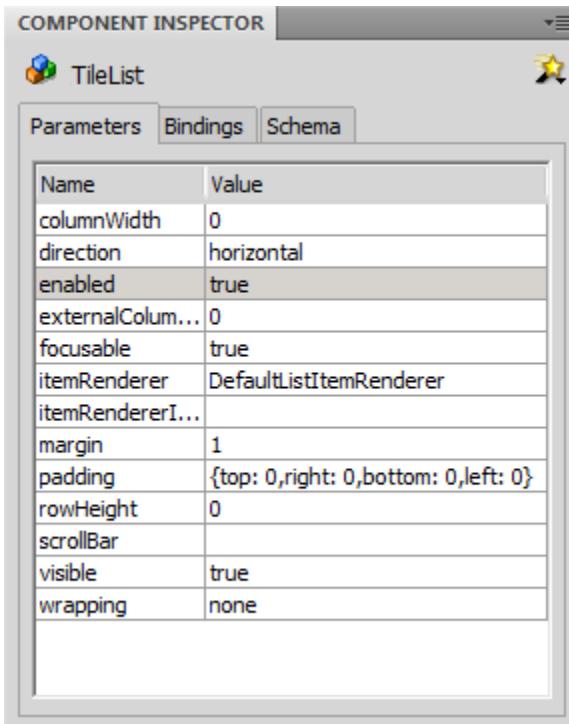


図 43: `TileList` のタイムライン

2.4.5.4 検証可能なプロパティ



TileList コンポーネントから生じる MovieClip は、以下の検証可能なプロパティを備えています：

columnWidth	内部で作成されたリスト項目インスタンスの幅です。 <i>rendererInstanceName</i> プロパティが設定されている場合、この値に効果はありません。
direction	スクロールの方向。行と列の意味はこの値に応じては変化しません。
enabled	false に設定しておくとコンポーネントを無効にします。無効にされたコンポーネントはユーザーの入力を受け取りません。
externalColumnCount	<i>rendererInstanceName</i> プロパティが設定されると、この値を使って外部のレンダラが使用する列の数を TileList に通知します。
focusable	コンポーネントのフォーカス管理を有効化/無効化します。フォーカス可能なプロパティーを false に設定すると Tab キー、方向キー、マウスボタンに基づいたフォーカス変更へのサポートを削除します。
itemRenderer	<i>ListItemRenderer</i> のシンボル名。内部でのリストアイテムインスタンスの生成に使用します。 <i>itemRendererInstanceName</i> プロパティがセットされている場合（つまり外部の <i>ListItemRenderers</i> を使用している場合）、なにも影響を与えません。
itemRendererInstanceName	この ScrollingList コンポーネントと共に使用する外部リストアイテム レンダラーのプリフィックス。Stage のリストアイテム

	インスタンスはこのプロパティー値でプリフィックスが付けられています。このプロパティーが「r」の値にセットされている場合、このコンポーネントと共に使用されるすべてのリストアイテム インスタンスは次の値である必要があります。「r1」、「r2」、「r3」、… 最初のアイテムは数字 1 を持つ必要があります。
margin	リストコンポーネントの境界と内部で生成されたリストアイテムの間のマージン。 <i>itemRendererInstanceName</i> プロパティーがセットされている場合（つまり外部の <i>ListItemRenderers</i> を使用している場合）、この値はなにも影響を与えません。
padding	リストアイテムに対するエキストラのパディング。 <i>itemRendererInstanceName</i> プロパティーがセットされている場合（つまり外部の <i>ListItemRenderers</i> を使用している場合）、この値はなにも影響を与えません。パディングは自動的に生成された <i>ScrollBar</i> にはなにも影響を与えません。
rowHeight	内部で生成されたリストアイテム インスタンスの高さ。 <i>itemRendererInstanceName</i> プロパティーがセットされている場合（つまり外部の <i>ListItemRenderers</i> を使用している場合）、この値はなにも影響を与えません。
visible	false に設定しておくとコンポーネントを非表示にします。これは添付された <i>ScrollBar</i> またはどのような外部の <i>ListItemRenderers</i> を非表示にしません。

2.4.5.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベントサブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase**: イベントフローでの現在のフェーズ。（EventPhase.CAPTURING_PHASE、EventPhase.AT_TARGET、EventPhase.BUBBLING_PHASE）。
- **bubbles**: イベントがバブルリングイベントかどうかを示す。
- **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

TileList コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティーは、共通プロパティーに追加して提供されています。

ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。
ComponentEvent.STATE_CHANGE	コンポーネントのステートが変更された。
FocusHandlerEvent.FOCUS_IN	コンポーネントがフォーカスを受け取った。
FocusHandlerEvent.FOCUS_OUT	コンポーネントがフォーカスを失った。

ListEvent.INDEX_CHANGE	選択されたインデックスが変更された。
	<ul style="list-style-type: none"> • <i>itemRenderer</i>:ダブルクリックされたリストアイテム。 IListItemRenderer タイプ。 • <i>itemData</i>:リストアイテムに関連するデータ。この値はリストのdataProvider から取得されます。ActionScript Object タイプ。 • <i>index</i>:list. int タイプの新たに選択されたインデックス。Values -1 (selectedIndex はセットされていない) からリストアイテム数-1 まで。 • <i>controllerIdx</i>:イベントの生成に使用されたコントローラー/マウスのインデックス (複数のマウスカーソル環境下でのみ該当)。
ListEvent.ITEM_PRESS	リストアイテムが押された。
	<ul style="list-style-type: none"> • <i>itemRenderer</i>:ダブルクリックされたリストアイテム。 IListItemRenderer タイプ。 • <i>itemData</i>:リストアイテムに関連するデータ。この値はリストのdataProvider から取得されます。ActionScript Object タイプ。 • <i>index</i>:list. int タイプの新たに選択されたインデックス。Values -1 (selectedIndex はセットされていない) からリストアイテム数-1 まで。 • <i>controllerIdx</i>:イベントの生成に使用されたコントローラー/マウスのインデックス (複数のマウスカーソル環境下でのみ該当)。
ListEvent.ITEM_CLICK	リストアイテムがクリックされた。
	<ul style="list-style-type: none"> • <i>itemRenderer</i>:ダブルクリックされたリストアイテム。 IListItemRenderer タイプ。 • <i>itemData</i>:リストアイテムに関連するデータ。この値はリストのdataProvider から取得されます。ActionScript Object タイプ。 • <i>index</i>:list. int タイプの新たに選択されたインデックス。Values -1 (selectedIndex はセットされていない) からリストアイテム数-1 まで。 • <i>controllerIdx</i>:イベントの生成に使用されたコントローラー/マウスのインデックス (複数のマウスカーソル環境下でのみ該当)。
ListEvent.ITEM_DOUBLE_CLICK	リストアイテムがダブルクリックされた。
	<ul style="list-style-type: none"> • <i>itemRenderer</i>:ダブルクリックされたリストアイテム。 IListItemRenderer タイプ。 • <i>itemData</i>:リストアイテムに関連するデータ。この値はリストのdataProvider から取得されます。ActionScript Object タイプ。 • <i>index</i>:list. int タイプの新たに選択されたインデックス。Values -1 (selectedIndex はセットされていない) からリストアイテム数-1 まで。 • <i>controllerIdx</i>:イベントの生成に使用されたコン

	トローラー/マウスのインデックス（複数のマウスカーソル環境下でのみ該当）。
ListEvent.ITEM_ROLL_OVER	<p>マウスのカーソルがリストアイテムの上に来た。</p> <ul style="list-style-type: none"> • <i>itemRenderer</i>:ダブルクリックされたリストアイテム。 IListItemRenderer タイプ。 • <i>itemData</i>:リストアイテムに関連するデータ。この値はリストのdataProvider から取得されます。ActionScript Object タイプ。 • <i>index:list. int</i> タイプの新たに選択されたインデックス。Values -1 (selectedIndex はセットされていない) からリストアイテム数-1 まで。 • <i>controllerIdx</i>:イベントの生成に使用されたコントローラー/マウスのインデックス（複数のマウスカーソル環境下でのみ該当）。
ListEvent.ITEM_ROLL_OUT	<p>マウスのカーソルがリストアイテムの上から離れた。</p> <ul style="list-style-type: none"> • <i>itemRenderer</i>:ダブルクリックされたリストアイテム。 IListItemRenderer タイプ。 • <i>itemData</i>:リストアイテムに関連するデータ。この値はリストのdataProvider から取得されます。ActionScript Object タイプ。 • <i>index:list. int</i> タイプの新たに選択されたインデックス。Values -1 (selectedIndex はセットされていない) からリストアイテム数-1 まで。 • <i>controllerIdx</i>:イベントの生成に使用されたコントローラー/マウスのインデックス（複数のマウスカーソル環境下でのみ該当）。

以下の例は TileList がいつフォーカスを受け取ったかを判断する方法を示しています:

```
myList.addEventListener(FocusHandlerEvent.FOCUS_IN, onListFocused);
function onListFocused(e:FocusHandlerEvent) {
    trace("myList is now focused!");
    // Do something
}
```

2.4.6 DropdownMenu

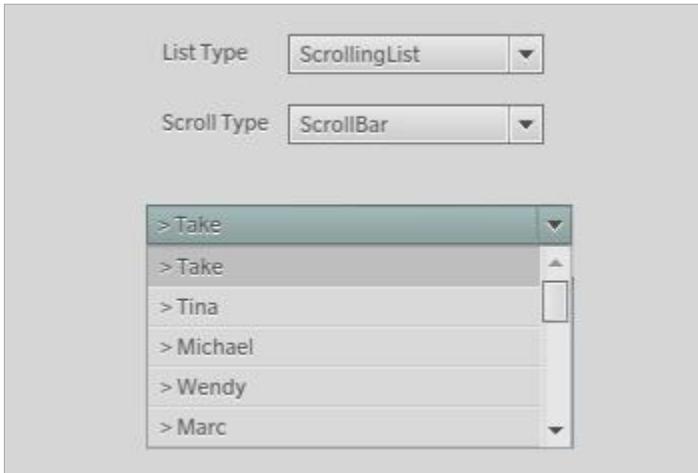


図 44: スкиニング前の DropdownMenu

DropdownMenu (scaleform.clik.controls.DropdownMenu) はボタンとリストの動作をラップします。このコンポーネントをクリックすると、選択するエレメントが含まれたリストが表示されます。DropdownMenu はアイドル状態の選択されたエレメントのみを表示します。このコンポーネントは、ScrollBar、またはScrollIndicator のいずれかを伴う ScrollingList か TileList の、どちらかを使用するよう構成できます。このリストはインストールされたdataProvider で設定されます。DropdownMenu のリスト エレメントはdataProvider で設定されます。以下の例のようにdataProvider はコードを使って割り当てられます:

```
dropdownMenu.dataProvider = new DataProvider(["item1", "item2", "item3",  
"item4"]);
```

デフォルトでは DropdownMenu はそのコンテンツのために ScrollingList コンポーネントを使用します。したがって、dropdown 検証可能プロパティが別のコンポーネントに変更されていない限り、ScrollingList と ListItemRenderer は動作するために FLA ファイルの [ライブラリ] にも存在しなければなりません。詳細は「検証可能プロパティ」の章を参照してください。

また、デフォルトでは DropdownMenu はスクロールバーをそのリスト エレメントには添付しないので注意してください。ScrollBar、またはScrollIndicator は、コードを使って DropdownMenu のリスト エレメントに添付する必要があります。詳細は「ヒントとコツ」の章を参照してください。

2.4.6.1 ユーザーの操作

DropdownMenu インスタンスをクリックする、または Enter キー、あるいは同等のコントロールを押すと、選択可能なエレメントのリストが表示されます。リストが開くと、フォーカスもそのリストに移動されます。ユーザーは ScrollingList、TileList、ScrollBar の「ユーザーの操作」の章で説明したように、リストを操作することができます。リスト項目をクリックすると、DropdownMenu コンポーネントでその項目を選択する、リストを閉じる、そして選択した項目を表示するようになります。リストの境界外でクリックすると自動的にリストが閉じられ、フォーカスは DropdownMenu コンポーネントに戻ります。

2.4.6.2 コンポーネントのセットアップ

DropdownMenu のそのほとんどの機能は Button コンポーネントから派生しています。その結果、DropdownMenu クラスを使用する MovieClip は以下の指定されたサブエレメントが必須です。このリストとスクロールバーはダイナミックに作成されます。オプションのエレメントである場合には、その旨表記されています：

- *textField* : (オプション) TextField タイプ。ボタンのラベルです。
- *focusIndicator* : (オプション) MovieClip タイプ。フォーカスが適用されたステートの表示に使用される別の MovieClip です。存在する場合、この MovieClip は "show" と "hide" の 2 つの指定されたフレームを備えている必要があります。

2.4.6.3 ステート

DropdownMenu は開いているときに切り替えられるので、ToggleButton や CheckBox と同じ選択状態を表すステートが必要となります。このようなステートには以下が含まれます：

- *up* またはデフォルトのステート
- マウス カーソルがコンポーネント上にあるとき、またはフォーカスが適用されているときの *over* ステート
- ボタンが押下されたときの *down* ステート
- *disabled* ステート
- *selected_up* またはデフォルトのステート
- マウス カーソルがコンポーネント上にあるとき、またはフォーカスが適用されているときの *selected_over* ステート
- ボタンが押下されたときの *selected_down* ステート
- *selected_disabled* ステート

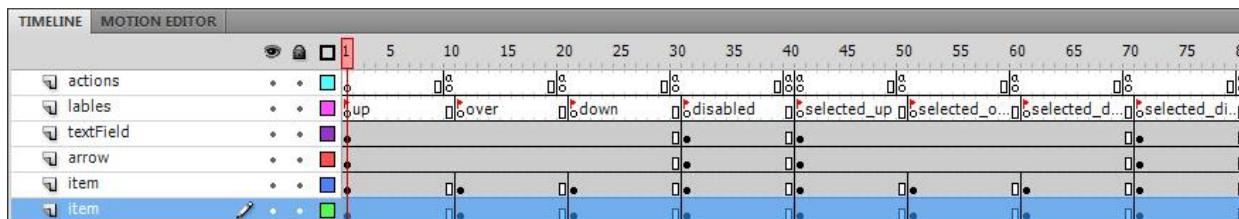
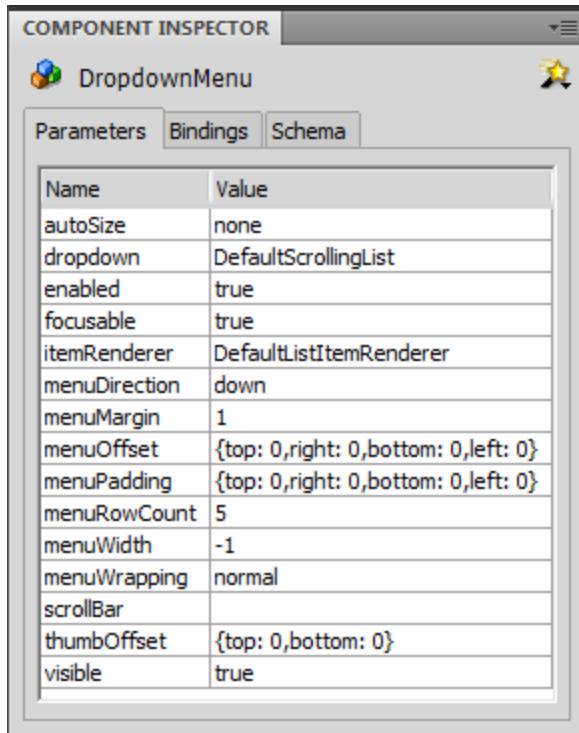


図 45: DropdownMenu のタイムライン

これは DropdownMenu に必要な最小限のキーフレーム セットです。Button コンポーネントがサポートするステートとキーフレームの拡張セットと、それに伴う DropdownMenu コンポーネントについては、「[Getting Started with CLIK Buttons](#)」を参照してください。

2.4.6.4 検証可能なプロパティ



DropdownMenu コンポーネントの検証可能なプロパティは以下のとおりです:

autoSize	閉じられた DropdownMenu が、その中のテキストサイズに合うように拡大縮小して、サイズの変わったボタンがどの方向に整列されるかを決定します。autoSize プロパティーを autoSize="none"に設定すると、その現在のサイズが変わりません。
Dropdown	DropdownMenu コンポーネントと併用するリスト コンポーネント (ScrollingList または TileList) のシンボル名です。
Enabled	Disables the component if set to false.
Focusable	デフォルトでは、コンポーネントはユーザーの対話的操でフォーカスを受け取ることができます。このプロパティーを false に設定するとフォーカス取得は無効になります。
menuDirection	ドロップダウンリストの開く方向。有効な値は"up"と"down"です。
menuMargin	リスト コンポーネントの境界と内部で作成されたリスト項目の間のマージンです。このマージンは、自動的に生成されたスクロールバーにも影響します。
menuOffset	ドロップダウンリストの、ドロップダウン最下部からの水平方向と垂直方向のオフセット量。水平方向の正の値はリストをドロップダウンボタンの水平位置の右に動かします。垂直方向の正の値はリストをボタンから遠ざけるように動かします。
menuPadding	リストアイテムの上、下、左、右のエキストラのパディング。自動的に生成されたスクロールバーにはなにも影響を与えません。

menuRowCount	リストの表示すべき行の数。
menuWidth	If set, this number will be enforced as the width of the drop down's list.
thumbOffset	スクロールバーのサムトップとボトムのオフセット。リストが自動的にスクロールバーインスタンスを生成しない場合は、このプロパティーはなにも影響を与えません。
scrollbar	ドロップダウンリストのスクロールバーのシンボル名。ドロップダウンリスト インスタンスが生成します。値が空の場合、ドロップダウンリストにはスクロールバーがありません。
Visible	Hides the component if set to false.

2.4.6.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベント サブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE、EventPhase.AT_TARGET、EventPhase.BUBBLING_PHASE) 。
- **bubbles**: イベントがバーブリングイベントかどうかを示す。
- **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

DropdownMenu コンポーネントが作成するイベントは以下の表のとおりです。change イベント以外、Button コンポーネントと同じです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。
ComponentEvent.STATE_CHANGE	コンポーネントのステートが変更された。
FocusHandlerEvent.FOCUS_IN	コンポーネントがフォーカスを受け取った。
FocusHandlerEvent.FOCUS_OUT	コンポーネントがフォーカスを失った。
ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ListEvent.INDEX_CHANGE	選択されたインデックスが変更された。 <ul style="list-style-type: none"> • itemRenderer: ダブルクリックされたリストアイテム。 IListItemRenderer タイプ。 • itemData: リストアイテムに関連するデータ。この値はリストのdataProvider から取得されます。ActionScript Object タイプ。 • index: list. int タイプの新たに選択されたインデックス。Values -1 (selectedIndex はセットされていない) からリストアイテム数-1 まで。 • controllerIdx: イベントの生成に使用されたコントローラー/マウスのインデックス (複数のマウスカーソル環境下でのみ該当) 。

MouseEvent.ROLL_OVER	マウス カーソルが、ボタン上に移動しました。 <i>mouseIdx</i> : イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。 <i>buttonIdx</i> : イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。
MouseEvent.ROLL_OUT	マウス カーソルが、ボタンから離れました。 <i>mouseIdx</i> : イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。 <i>buttonIdx</i> : イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。
ButtonEvent.PRESS	ボタンが、押下されました。 <i>controllerIdx</i> : イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。 <i>isKeyboard</i> : イベントがキーボード/ゲームパッドで生成されたときは true、イベントがマウスで生成されたときは false。 <i>isRepeat</i> : イベントが autoRepeating ボタンを押し続けることで生成されたときは true、ボタンが現在反復していないときは false。
MouseEvent.DOUBLE_CLICK	ボタンが、ダブルクリックされました。 <i>doubleClickEnabled</i> プロパティが true の場合に限って発生します。 <i>mouseIdx</i> : イベントの生成に使用されたマウスのカーソルのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。 <i>buttonIdx</i> : イベントがどのボタンに対して生成されたかを示します（ゼロ基準のインデックス）。Scaleform のみ、イベントを MouseEventEx にキャストすることが必要。
ButtonEvent.CLICK	ボタンが、クリックされました。 <i>controllerIdx</i> : イベントの生成に使用されたコントローラーのインデックス（複数のマウスカーソル環境下でのみ該当）。ユニットタイプ。 <i>isKeyboard</i> : イベントがキーボード/ゲームパッドで生成されたときは true、イベントがマウスで生成されたときは false。

	<i>isRepeat</i> : イベントが autoRepeating ボタンを押し続けることで生成されたときは true、ボタンが現在反復していないときは false。
ButtonEvent.DRAG_OVER	マウス カーソルが (マウスの左ボタンを押したままの状態で)、ボタン上にドラッグされました。 <i>controllerIdx</i> : イベントの生成に使用されたコントローラーのインデックス (複数のマウスカーソル環境下でのみ該当)。ユニットタイプ。
ButtonEvent.DRAG_OUT	マウス カーソルが (マウスの左ボタンを押したままの状態で)、ボタンから離れました。 <i>controllerIdx</i> : イベントの生成に使用されたコントローラーのインデックス (複数のマウスカーソル環境下でのみ該当)。ユニットタイプ。
ButtonEvent.RELEASE_OUTSIDE	マウス カーソルが、ボタンから離れ、マウスの左ボタンが解放されました。 <i>controllerIdx</i> : イベントの生成に使用されたコントローラーのインデックス (複数のマウスカーソル環境下でのみ該当)。ユニットタイプ。

2.5 プログレス タイプ[°]

プログレス タイプは、イベントや動作のステータスや進捗状況 (プログレス) の表示に使用されます。Scaleform CLIK フレームワークには、このカテゴリーに属する StatusIndicator と ProgressBar という 2 つのコンポーネントが含まれています。StatusIndicator コンポーネントは、イベントや動作のステータスの表示に使用されます。ProgressBar コンポーネントは StatusIndicator と同じセマンティクスを備えていますが、プログレス イベントを作成する他のコンポーネントや動作をリッスンする追加機能を含んでいます。



図 46: *Mercenaries 2 - FACTIONS* (勢力) ステータス インジケータの例

2.5.1 StatusIndicator

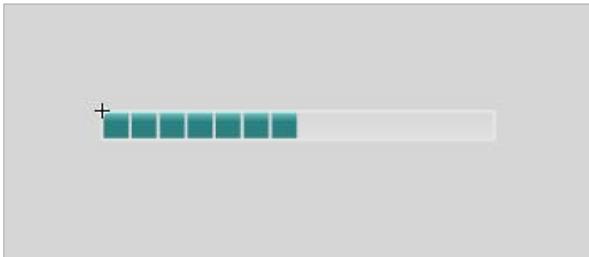


図 47: スキニング前の StatusIndicator

StatusIndicator コンポーネント (`scaleform.clik.controls.StatusIndicator`) は、視覚的なインジケータとしてタイムラインを使用するイベントや動作のステータスを表示します。StatusIndicator の値は、コンポーネントのタイムラインで再生されるフレーム番号を生成するための、最小値と最大値で補間されます。コンポーネントのタイムラインはステータスの表示に使用されるので、画期的な視覚インジケータを完全に自由に作成することができます。

2.5.1.1 ユーザーの操作

StatusIndicator にはユーザーの操作はありません

2.5.1.2 コンポーネントのセットアップ

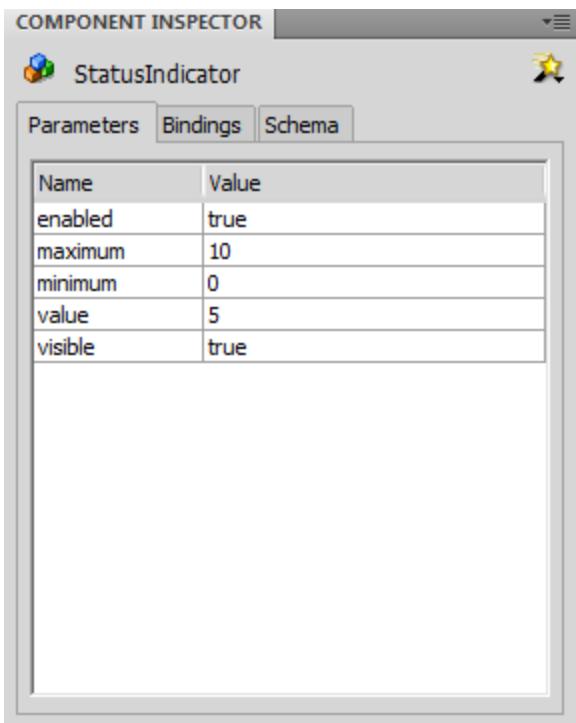
CLIK StatusIndicator クラスを使用する MovieClip は、指定されたサブエレメントをいっさい必要としません。ただし、StatusIndicator は、正確な操作に少なくとも 2 つのフレームを必要とします。最初のフレームに `stop()` コマンドを必ず挿入して、フレームを再生しないようにします。

StatusIndicator コンポーネントは、その `value` プロパティで作成される関連フレームで `gotoAndStop()` となります。

2.5.1.3 ステート

StatusIndicator コンポーネントにはステートはありません。コンポーネントのフレームは、イベントや動作のステータスの表示に使用されます。

2.5.1.4 検証可能なプロパティ



StatusIndicator コンポーネントから生じる MovieClip は、以下の検証可能なプロパティを備えています：

visible	false に設定した場合、このコンポーネントを非表示にします。
enabled	true に設定した場合、コンポーネントを無効にします。
value	イベント、または動作のステータス値です。再生されるフレーム番号を生成するため、最小値と最大値で補間されます。
minimum	ターゲット フレームの補間に使用される最小値です。
maximum	ターゲット フレームの補間に使用される最大値です。

2.5.1.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベントサブクラス パラメーターを受け取ります。次のプロパティはすべてのイベントに対して共通です。

- **type**: イベントのタイプ。
- **target**: イベントのターゲット。
- **currentTarget**: イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase**: イベントフローでの現在のフェーズ。 (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE) 。
- **bubbles**: イベントがバーピングイベントかどうかを示す。
- **cancelable**: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。

StatusIndicator は特別なイベントをいっさい作成しません。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。

2.6 その他のタイプ

また、Scaleform CLIK フレームワークは簡単に分類できない複数のコンポーネントでも構成されていますが、それらも UI 開発者に貴重な機能を提供しています。ウィンドウのコンポーネントはモーダルと、コンテンツまたはダイアログとして使用できるモーダルの無いウィンドウを表示できます。

DragSlot は CLIK コンポーネントのベース実装で、ドラッグ・ドロップ機能を要する UI での使用のために設計されています。ドラッグ・ドロップは CLIK 内でカスタムフレームワークとして実装され、CLIK DragManager のインストールを要します (scaleform.clik.managers.DragManager.as 参照)。

CLIK ウィンドウコンポーネント (scaleform.clik.controls.Window) は Alert ダイアログのようなウィンドウやダイアログを表示します。

PopUpManager はウィンドウをダイアログとして開く機能を与え (PopUpManager.showModal() 参照)、複数のダイアログを同時に管理します。

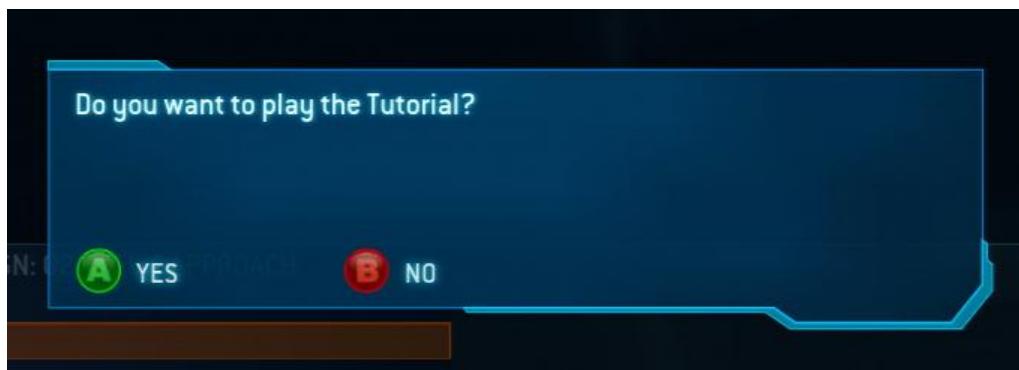


Figure 48: Window example from *Halo Wars*.

2.6.1 Window

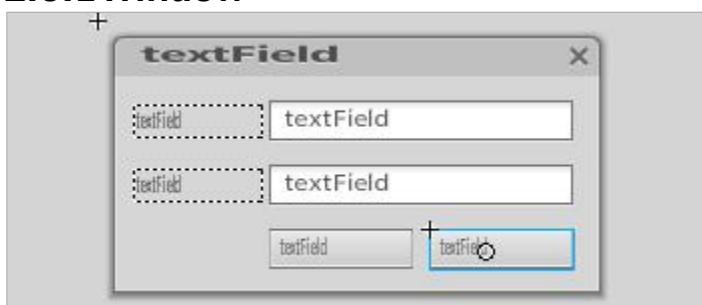


Figure 49: Sample unskinned Window.

CLIK ウィンドウコンポーネント (scaleform.clik.controls.Window) は Alert ダイアログのようなウィンドウやダイアログを表示します。

このコンポーネントは、ダイアログとして任意の MovieClip のインスタンス化、表示、非表示を行うスタティックインターフェイスや、実際のダイアログ MovieClip として使用（拡張）できるベースクラスを提供します。一度に 1 つのダイアログだけを確実に開くため、新規の Dialog.show() 呼び出しは、現在開いているダイアログを閉じます。

PopUpManager はウィンドウをダイアログとして開く機能を与え（PopUpManager.showModal() 参照）、複数のダイアログを同時に管理します。

2.6.1.1 ユーザーの操作

Dialog のユーザー操作は、このコンポーネントが作成するダイアログ内で定義されます。

2.6.1.2 コンポーネントのセットアップ

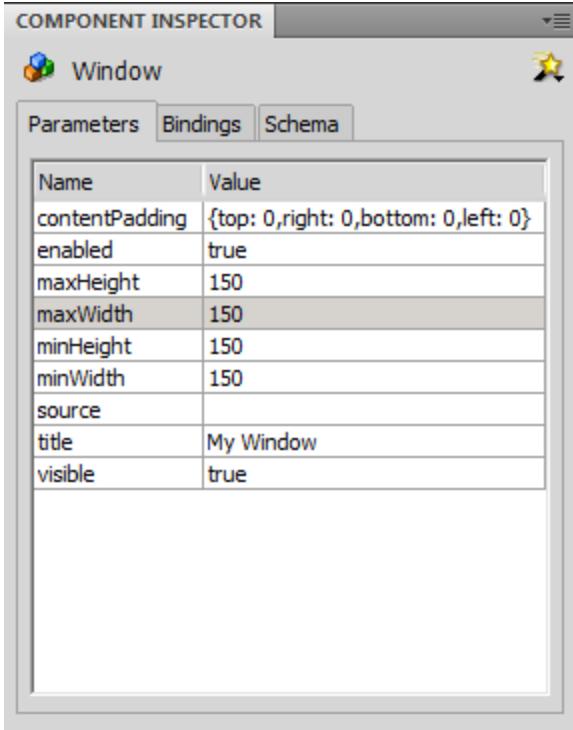
CLIK Dialog クラスを使用する MovieClip には、以下の指定されたサブエレメントが必須です。オプションのエレメントである場合には、その旨表記しています：

- **closeBtn:** (オプション) CLIK Button タイプ。ウィンドウを閉じる Button。
- **titleBtn:** (オプション) CLIK Button タイプ。ダイアログに対するドラッグできるタイトルバー。
- **okBtn:** (オプション) CLIK Button タイプ。クリックされるとウィンドウを閉じる「Accept」または「OK」ボタン。
- **resizeBtn:** (オプション) CLIK Button タイプ。押して、ドラッグするとユーザーがウィンドウをサイズ変更できる Button。
- **background** (オプション) MovieClip タイプ。ウィンドウのバックグラウンド。ウィンドウがサイズ変更された場合に拡大縮小、サイズ調節されます。
- **hit:** (オプション) MovieClip タイプ。ウィンドウの hitArea。

2.6.1.3 ステート

Dialog コンポーネントにはステートはありません。ダイアログとして表示される MovieClip は、独自のステートを持つ場合も、持たない場合もあります。

2.6.1.4 検証可能なプロパティ



Dialog コンポーネントから生じる MovieClip は、以下の検証可能なプロパティを備えています：

contentPadding	Window のコンテンツローダーに適用されるべき上下左右のパディング。
enabled	Disables the component if set to false.
maxHeight	Window がサイズ変更ボタンでサイズ変更されたときの最大高。
maxWidth	Window がサイズ変更ボタンでサイズ変更されたときの最大幅。
minHeight	Window がサイズ変更ボタンでサイズ変更されたときの最小高。
minWidth	Window がサイズ変更ボタンでサイズ変更されたときの最小幅。
source	Window にロードされるべきシンボルのエクスポート名。
title	Window に titleBtn 部分要素がある場合、この String はそのラベルとして使用されます。
visible	false に設定しておくとコンポーネントを非表示にします。

2.6.1.5 イベント

イベントに対するすべてのコールバックは、イベントに関する情報を含む 1 つのイベントまたはイベントサブクラス パラメーターを受け取ります。次のプロパティーはすべてのイベントに対して共通です。

- **type:** イベントのタイプ。
- **target:** イベントのターゲット。
- **currentTarget:** イベントリスナー付きのイベントオブジェクトをアクティブに処理するオブジェクト。
- **eventPhase:** イベントフローでの現在のフェーズ。（EventPhase.CAPTURING_PHASE、EventPhase.AT_TARGET、EventPhase.BUBBLING_PHASE）。
- **bubbles:** イベントがバブルリングイベントかどうかを示す。

cancelable: イベントに関連づけられているビヘイビアを回避できるかどうかを示す。Dialog コンポーネントが作成するイベントは以下の表のとおりです。イベントの隣に表示されているプロパティは、共通プロパティに追加して提供されています。

ComponentEvent.SHOW	可視プロパティーがランタイムで true に設定された。
ComponentEvent.HIDE	可視プロパティーがランタイムで false に設定された。

以下の例はダイアログ送信イベントを処理する方法を示しています:

```
myWindow.addEventListener(ComponentEvent.HIDE, onWindowClosed);
function onWindowClosed(e:ComponentEvent) {
    // Do something.
}
```

3 アートの詳細

この章は Scaleform CLIK コンポーネントのスキンの開発でデザイナーを支援し、簡単なコンポーネント サンプルのスキニングの詳細、さらにベスト プラクティス、アニメーション、フォントの埋め込みについて説明しています。

3.1 ベスト プラクティス

この章には、Scaleform CLIK コンポーネントのスキンを作成するときの、ベスト プラクティスが含まれています。

3.1.1 ピクセル パーフェクトなイメージ

CLIK コンポーネントのベクター ベースのスキンを開発している場合、すべてのアセットをピクセル パーフェクトにすることをお勧めします。ピクセル パーフェクトなアセットとは、Flash グリッド上に完璧に並んでいるアセットです。

グリッドを有効にして変更する：

1. Flash メニューの [表示] を選択します。
2. [グリッド] を選択して [グリッドを表示] をオンにします。
3. Flash メニューの [表示] を再度選択します。
4. [グリッド] を選択し [グリッドの編集] を選択します。
5. 縦横のサイズを 1 px と入力します。
6. [OK] をクリックします。

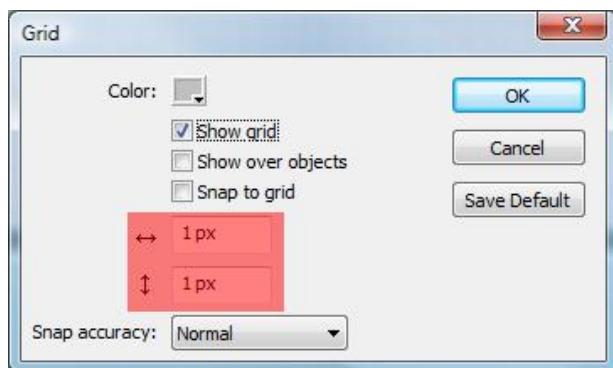


図 50: [グリッドの編集] のウィンドウ

ステージにグリッド オーバーレイが表示されているはずです。各グリッドは正確に 1 ピクセルを表しています。アート アセットを作成するときに、それらのアセットがこのグリッドに吸着るようにしてください。これにより、最終的な SWF は確実にぼやけないようになります。注意：すべてのイメージサイズを 2 のべき乗に維持してください。そうしないと、設定してもぼやけてしまう場合があります。以下の 2 のべき乗の小見出しを参照してください。

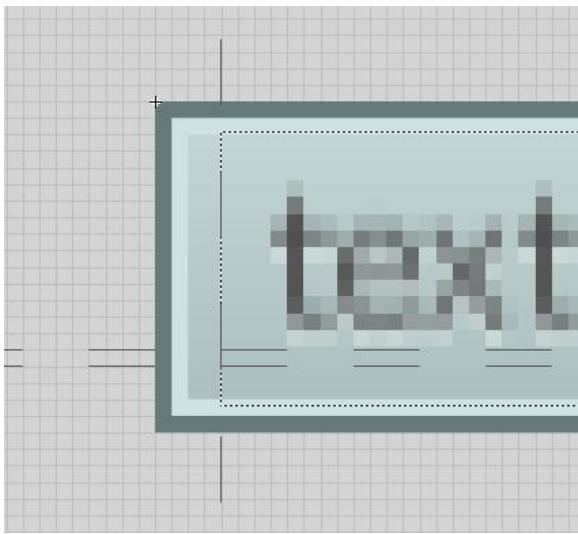


図 51: ピクセル パーフェクトなベクター グラフィック スキン

3.1.2マスク

Flash のマスクを使うと、デザイナーは実行時にグラフィックを部分的に非表示にすることができます。マスクはアニメーション効果によく使用されますが、実行時に非常に負荷が高くなる場合があります。Scaleform は可能な限りマスクの使用を避けることをお勧めします。マスクを使用しなければならない場合、その数を一桁台に保ち、マスクありとマスクなしでムービーのパフォーマンスをテストしてください。

Flash でマスクの代わりに使用できるものとして、マスク アウトが必要な領域に、アルファ ブレンド付きの PNG を Photoshop® で作成することができます。ただし、これは透明な領域のアニメーションがないイメージのみに有効です。

3.1.3アニメーション

四角形を円に変形するなど、1 つのベクター シエイプを別のシェイプに変形するアニメーションは、避けておくのが最善の策です。そのシェイプをすべてのフレームで再評価しなければならないので、この種のアニメーションは非常に負荷が高くなります。

可能な場合は、アニメーションをベクター グラフィックのためにスケーリングすることを避けてください。追加のテッセレーション (ベクター シエイプを三角形に変換する処理) のため、メモリとパフォーマンスに影響を及ぼす可能性があるからです。最も負荷の低いアニメーションは変形と回転です。追加のテッセレーションが必要ないからです。

プログラム的なトゥイーンを避け、タイムライン ベースのトゥイーンを選択します。タイムライン トゥイーンのほうがパフォーマンスの点でははるかに負荷が低いからです。タイムライン トゥイーンの場合、キーフレーム数を希望するフレームレートで滑らかなアニメーションの実現に必要な最小限の数に維持してください。

3.1.4 レイヤーと描画プリミティブ

Flash でスキンを作成するときは、可能な限り使用するレイヤーの数を少なくします。使用されるレイヤーごとに最低でも 1 つの描画プリミティブが追加されます。使用される描画プリミティブの数が増えると、必要なメモリ サイズも増加し、パフォーマンスも急激に低下します。

3.1.5 複雑なスキン

複雑なスキンにはビットマップを使用するのが最善の方法ですが、簡単なスキンにはベクター グラフィックのほうがメモリの節約になり、画質の劣化 (ぼやけ) を起こさずに、どのような解像度にもスケーリングすることができます。

3.1.6 2 のべき乗

すべてのビットマップのサイズは 2 のべき乗を保つ必要があります。2 のべき乗のビットマップ サイズの例:

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

3.2 既知の問題と推奨ワークフロー

この章では、アートに関する既知の Flash の問題と、Scaleform CLIK で作業する場合の推奨ワークフローを紹介しています。

3.2.1 コンポーネントの複製

Flash にはコンポーネントを複製するときの問題があり、このためリンクエージ情報やコンポーネントの定義情報が、コピー元から新規のコンポーネントにコピーされなくなります。そのような場合、このリンクエージ情報のないコンポーネントは機能しません。この章では、この問題を回避する 2 つの方法を紹介しています。

3.2.1.1 コンポーネントの複製 (方法 1)

Button などの未変更 (スキニング前の) CLIK コンポーネントを、外部の FLA ファイルからコピー先の FLA ファイルに複製する最も早い方法は以下のとおりです:

1. *CLIK_Components.fla* ファイルを開きます。
2. そのファイルの [ライブラリ] パネルからコンポーネント (Button など) を、そのコンポーネント上で右クリックして [コピー] を選択し、次にコピー先の FLA の [ライブラリ] パネルで右クリックして [ペースト] を選択してペーストします。

3. コピー先の FLA の [ライブラリ] パネルでそのコンポーネントを右クリックして [名前を変更] を選択し、'Button' 以外の名前に変更します。
4. 再度、コピー先の FLA の [ライブラリ] パネルでそのコンポーネントを右クリックして、[プロパティ] を選択します。
5. ステップ 3 でコンポーネントに選択した新規の名前に合致するように、[識別子] フィールドを変更します。
6. [ライブラリ] パネルの空白の領域で右クリックして、[ペースト] を選択します。元のコンポーネントの新規のコピーが、リンクエージ情報はすべてそのままでペーストされます。

3.2.1.2 コンポーネントの複製 (方法 2)

同じライブラリにシンボル (コンポーネント) を複製するとき、リンクエージ情報は新規の複製したシンボルにはコピーされません。そのコンポーネントを機能させるために、この情報を入力する必要があります。以下の手順に従います:

1. [ライブラリ] パネルで複製するコンポーネントを右クリックして、[プロパティ] を選択します。
2. [シンボルプロパティ] ウィンドウで、[クラス] テキストフィールドのテキスト (scaleform.clik.controlsButton など) をダブルクリックして強調表示し、キーボードで Ctrl+C キーをクリックしてコピーします。
3. [キャンセル] をクリックします。
4. 再度、複製するコンポーネントを右クリックして、[複製] を選択します。
5. [シンボルの複製] ウィンドウで [ActionScript に書き出し] チェックボックスをクリックしてオンにします。
6. [クラス] フィールドをクリックし、Ctrl+V キーを押してリンクエージ情報をこのテキストフィールドにペーストします。
7. [OK] をクリックします。
8. [ライブラリ] パネルでこの新規にコピーしたコンポーネントを右クリックして、[コンポーネント定義] を選択します。
9. 空白の [クラス] フィールドをクリックし、Ctrl+V キーを押してリンクエージ情報をこのテキストフィールドにペーストします。
10. [OK] をクリックします。

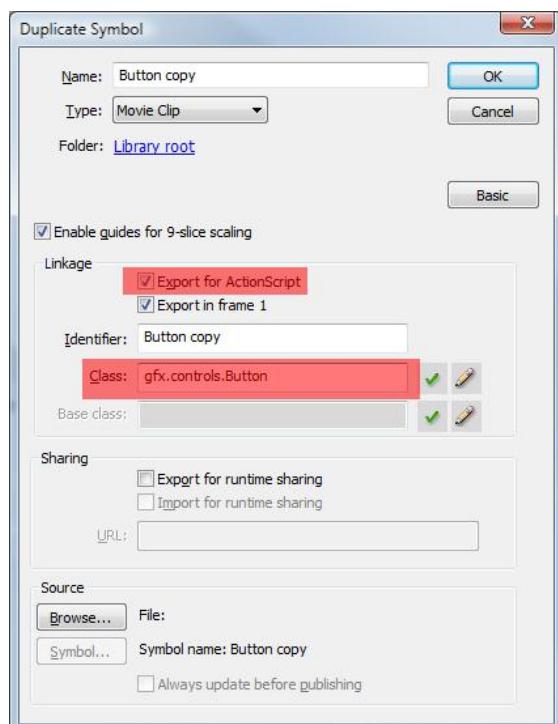


図 52: シンボルの複製ダイアログのリンクエージ (赤で強調表示されている領域を設定する)

図 53: コンポーネント定義ダイアログ ([クラス] に入力する)

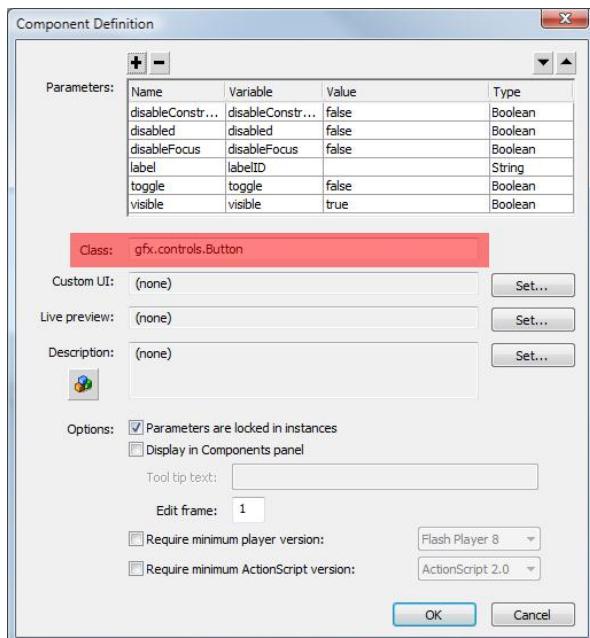
3.3 スкиニングの例

Scaleform CLIK の大きな利点は、表示レイヤーと機能レイヤーを分けています。この分割により大部分は、プログラマとデザイナーがお互いに独立して作業できます。楽に外観と操作感をカスタマイズできることは、この分割の主要な利点の 1 つです。

プリビルド CLIK コンポーネントは標準の外観と操作感を備えていますが、ユーザーが自分のニーズに合わせてカスタマイズするように作られています。これ以降の章で、このようなプリビルド コンポーネントをカスタマイズする方法について説明します。

一連の CLIK コンポーネントは、標準の Flash シンボルと同じように簡単にスキンングされます。FLA の [ライブラリ] パネルでコンポーネントをダブルクリックし、そのコンポーネントのタイムラインにアクセスして以下のいずれかを行います：

- Flash でデフォルトのスキンを変更する。
- Flash でカスタムのスキンを始めから作成する。
- Photoshop や Illustrator® で作成したアート アセットを Flash にインポートする。



詳細なスキンングのチュートリアルについては、「[Getting Started with CLIK](#)」を参照してください。

3.3.1 StatusIndicator のスキンング

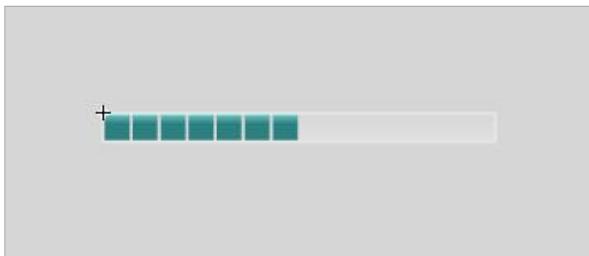


図 54: スкиニング前の StatusIndicator

StatusIndicator は独自なコンポーネントで、Button をベースにしている他のほとんどのコンポーネントとは少し異なる方法が必要となります。StatusIndicator コンポーネントを開き、タイムラインを確認します。*indicator* と *track* という 2 つのレイヤーがあります。*track* は背景のグラフィックの視覚的な表示に使用されます。それ以外の目的には機能しません。*indicator* レイヤーは複数のキーフレームと、ビットマップか、ベクター グラフィックのいずれかを使用して、昇順で段階的にステータスを表現します。

このチュートリアルでは、1 つの Photoshop ファイルで作成した複数のビットマップを複数のレイヤーで使用して、ステータス インジケーターを表現します。ここでは StatusIndicator をスキニングする 1 つの方法を説明していますが、ステージでグラフィックを作成し組み立てる方法は他にもあります。StatusIndicator が適切に動作するためには、最終的な結果は同じになる必要があります。

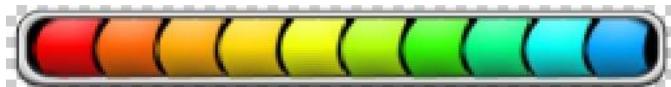


図 55: StatusIndicator の PSD ファイル

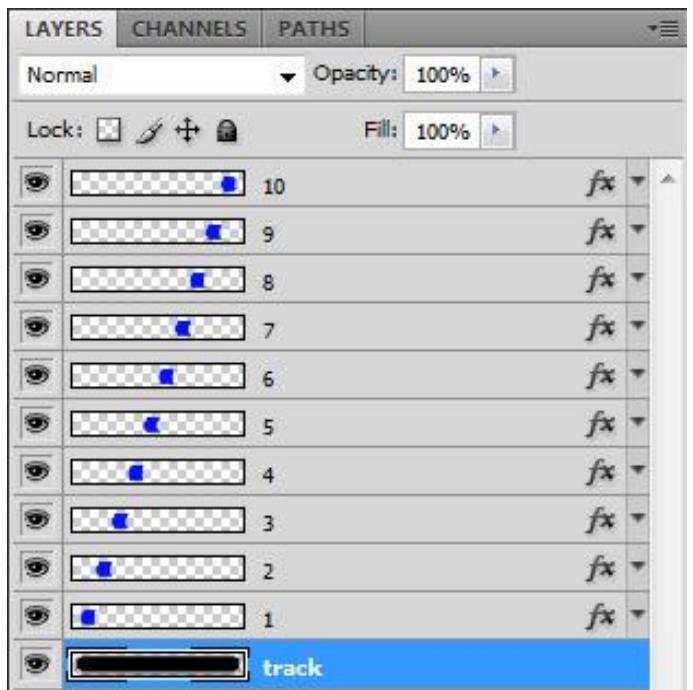


図 56: StatusIndicator PSD ファイルの Photoshop におけるレイヤー セットアップ

1. 上図のような StatusIndicator ビットマップを Photoshop で作成します。レイヤーの順序と番号の付け方、さらにレイヤー上の各イメージの位置に注意します。背景が透明であることを確認します。このチュートリアル用に同様のファイルを作成してください。
2. PSD として作ったファイルを保存します。
3. Flash でメニューの [ファイル] を選択し、[読み込み]->[ステージに読み込み] を選択します。
4. StatusIndicator のスキン PSD ファイル（先ほど保存しておいたもの）を参照して選択します。
5. […に読み込む] ウィンドウで [レイヤーを次に変換] が [Flash レイヤー] に設定されていることを確認します。
6. [OK] をクリックします。
7. PSD ファイルの各レイヤーに対して、新規の Flash タイムライン レイヤーが作成されたはずです。上記のイメージの場合、PSD ファイルには 10 個のレイヤーがあるので、10 個のレイヤーが作成されました（フレームごとに 1 つ）。各レイヤーは 1 から 10 までのラベルが付き、1 が最下部にあり 10 が最上部にあります。[レイヤー1] を選択します。
8. ステージですべてのビットマップ イメージの周囲に、選択範囲を描画します。
9. イメージを以前のスキニング前のトラックの上に移動して、必要に応じてサイズ変更します。
- 10.[レイヤー1] の最初のキーフレームを選択して、フレーム 6 にドラッグします。
- 11.[レイヤー1] でフレーム 11 に新規のキーフレームを追加します。フレーム 11 で右クリックして [キーフレームの挿入] を選択します。
- 12.[レイヤー2] のビットマップを選択して、Ctrl+X キーを押してカットします。
- 13.[レイヤー1] のフレーム 11 の新規のキーフレームを選択して、Ctrl+Shift+V キーを押して [レイヤー1] で正確に同じ場所にそのビットマップを配置します。
- 14.10 個のビットマップすべてを同じレイヤー（レイヤー1）に置くまで、正しいキーフレームでこの処理を繰り返します。後続の各ビットマップは、前回のキーフレームから 5 フレーム後のキーフレームにコピーされるはずです。[レイヤー2] のビットマップはキーフレーム 11 に、[レイヤー3] のビットマップはフレーム 16 に、[レイヤー4] のビットマップはフレーム 21 に、という具合にコピーされるはずです。10 個のイメージの PSD を使うと、最後のキーフレームはフレーム 51 になるはずです。
15. 空白のレイヤー（2-10）を削除してクリーンアップします。
16. 最後のビットマップ キーフレームの後にタイムライン上に追加のフレームがある場合、さらに 5 つのキーフレームを数えて、次に残りのフレームを選択し右クリックして [フレームの削除] を選択し削除します。このチュートリアルでは、最後のフレームはフレーム 55 になるはずです。
17. 以前の *indicator* レイヤーを選択して削除します。
18. 以前の *track* レイヤーを選択して削除します。インポートされた新規の *track* レイヤーを削除しないようにしてください。
- 19.[レイヤー1] を選択して、'indicator' という名前に変更します。
20. この *indicator* レイヤーと *track* レイヤーを *actions* レイヤーの下にドラッグします。*track* レイヤーが必ず *indicator* レイヤーの下になるようにします。

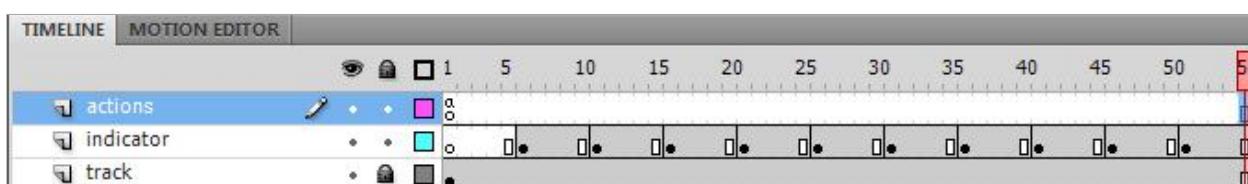


図 57: 最終的なタイムライン (キーフレームの位置と最後のフレームの位置に注意する)

- 21.StatusIndicator のタイムラインを終了します。
- 22.検証可能パラメータの値を 1 から 10 までの任意の数に設定します。
- 23.このファイルを保存します。
- 24.ファイルをパブリッシュして、新規にスキニングされたインジケータを確認します。



図 58: スкиニングされた StatusIndicator

3.4 フォントとローカライズ

この章では、Scaleform CLIK コンポーネントのフォントの使用について説明します。

3.4.1はじめに

フォントを Scaleform で正しく表示するためには、必要なグリフ (フォントの文字) を SWF に埋め込むか、または Scaleform のローカライズ システムを使ってセットアップする必要があります。これ以降の章では、Flash UI でフォントを管理する方法について説明します。

3.4.2 フォントの埋め込み

Flash Player とは異なり、Scaleform でフォントを適切に表示するには、そのフォントを埋め込む必要があります。フォントが埋め込まれていない場合、それらのフォントがシステムに存在するとしても、Scaleform Player は空白の四角形を表示します。この表示の利点は、フォントが Scaleform に正しく埋め込まれていることを簡単に確認できることです。プリビルド コンポーネントセットでは、Slate Mobile が各 textField インスタンスに埋め込まれています。Slate Mobile には中国語、日本語、韓国語 (CJK) の文字、またはグリフは一切含まれていません。プリビルド コンポーネントは ASCII グリフしか埋め込んでいないので注意してください。Slate Mobile を CJK グリフを含むフォントに置き換えて、適切な埋め込みオプションを設定して変更することができます。

フォントを直接 textField に埋め込むことは、Scaleform のローカライズ システム (3.4.4 章を参照してください) とは互換性がないので注意してください。Scaleform は実際には、Scaleform のローカライズ システムを使ってフォントをセットアップすることをお勧めしています。直接埋め込むよりも多くの利点を備えているからです。ただし、ローカライズが必要ない場合など、フォントの埋め込みのほうが優れている場合もあります。UI を管理するのと同様に、フォントの管理もローカライズやメモリの管理など、複数の考慮点があります。

3.4.3 textField にフォントを埋め込む

フォントを埋め込む必要があるのは、ダイナミック、またはインプット textField だけです。スタイルテキストは、コンパイルすると自動的にアウトライン（未加工のベクター シエイプ）に変換されます。フォントをダイナミック textField に埋め込むには、ステージでその textField を選択して、[プロパティ インスペクタ] ([ウィンドウ]->[プロパティ インスペクタ]) の [埋め込み] ボタンをクリックします。埋め込む必要のある文字、または文字セットを選択して、[OK] を選択します。これらの手順を完了すると、そのフォントは FLA で使用できるようになります。同じフォントが同じ FLA の複数の textField で使用される場合は、この方法でそのフォントを埋め込むのは一度だけです。逆に、何回もそのフォントを埋め込んでもファイル サイズやメモリの使用量は増えません。中国語など多くのグリフを含む文字セットは、ロードされたときに大量のメモリを使用する可能性があるので注意してください。

3.4.4 Scaleform のローカライズ システム

Scaleform のローカライズ システムは現在のロケールが変更になるたびに、フォント ライブラリをロード/アンロードするホット スワップ メカニズムを使用しています。このようなフォント ライブラリ自体は、コンテンツ SWF が使用する埋め込まれたフォント グリフを含んだ SWF ファイルです。また、Scaleform はこのフォント ライブラリのグリフを使って、ダイナミックにオンデマンドでフォントを変更する効果的な方法を提供します。

Scaleform のローカライズ システムは、textField が直接グリフを埋め込んでいる場合はフォントを交換できないので、その textField はインポートされたフォント シンボルを使用する必要があります。通常、フォント シンボルは gfonfotlib.fla というファイルに作成され、コンテンツ SWF にインポートされます。このインポートされたフォントは、フォントの交換をサポートするすべての textField 上に設定されます。Scaleform は、このフォント シンボル リンクを乗っ取って、現在のロケールに基づいた別のフォントをロードできるようになっているのです。

フォント ライブラリはフォントをエクスポートする必要はありません。該当するフォントを埋め込む必要があるだけです（フォントを埋め込む方法について、前の章を参照してください）。Scaleform のローカライズ システムは fontconfig.txt ファイルを使って、ロケールごとに使用するフォント ライブラリを定義し、さらに textField が使用するフォント シンボルとフォント ライブラリに埋め込まれている物理フォント間のフォント マッピングを定義します。

また、fontconfig.txt ファイルには変換マップも含まれています。Scaleform のローカライズ システムはオンザフライで、ステージ上のすべてのダイナミック、またはインプット textField に表示されるテキストの変換を実行します。たとえば、textField に '\$TITLE' というテキストが含まれていて、変換マップには \$TITLE=Scaleform などのマッピングがある場合、この textField は自動的に 'Scaleform' と表示します。

この章で提示している情報は、Scaleform フォントとローカライズ システムの大まかな概要を示したもので、Scaleform におけるフォントとローカライズを完全に理解するには、「[Font Overview](#)」を参照してください。

4 プログラミングの詳細

この章では、フレームワークの要点を説明し、各サブシステムに焦点を当てて Scaleform CLIK コンポーネントのアーキテクチャを大まかに理解できるようにします。

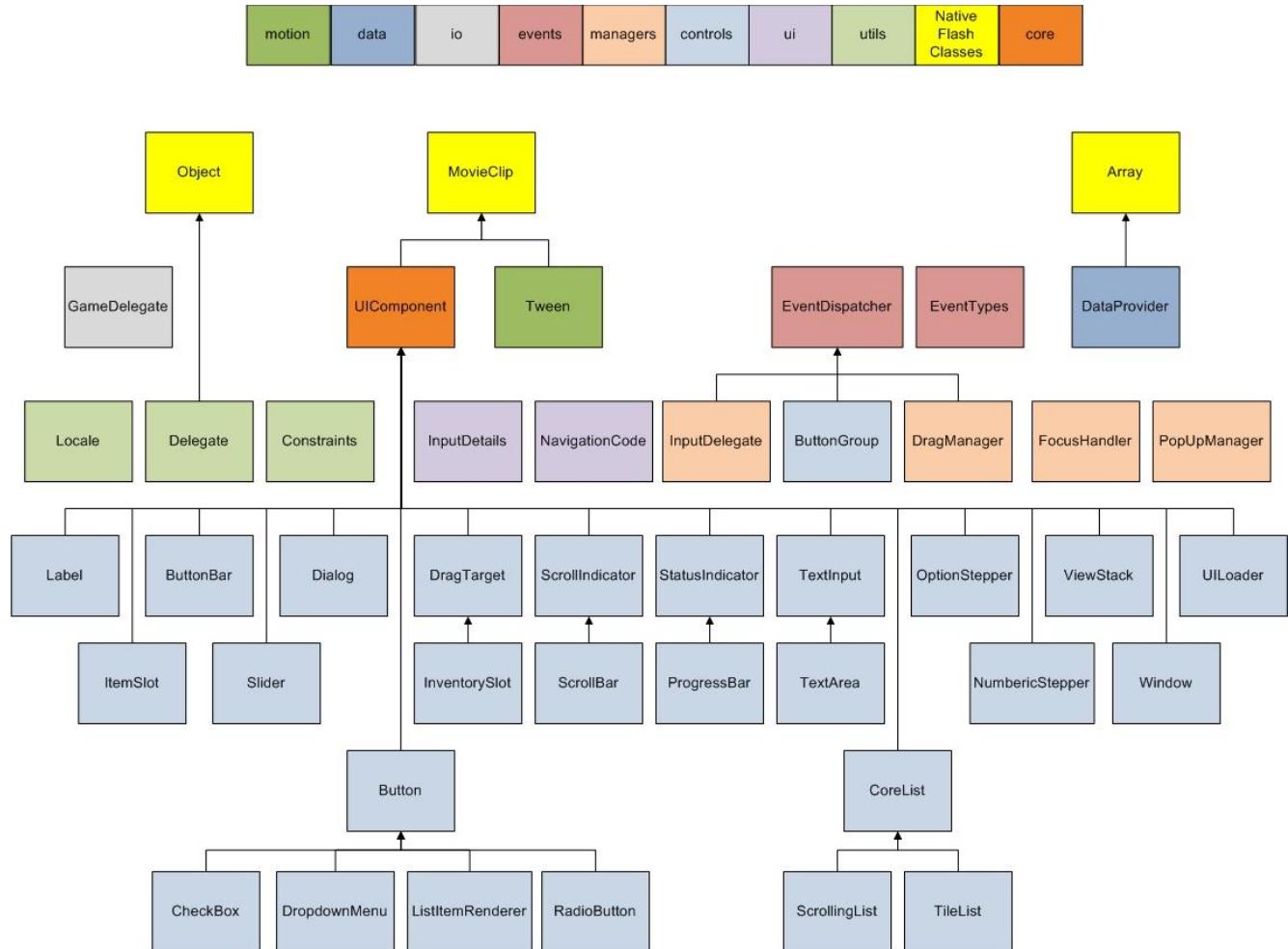


図 59: Scaleform CLIK クラスの階層

4.1 UIComponent

「プリビルド コンポーネント」の章では、Scaleform CLIK にバンドルされた具体的なコンポーネントが使用するクラスについて説明しました。それらのコンポーネントはすべて、**UIComponent** クラス (`scaleform.clik.core.UIComponent`) からコア機能を継承しています。この **UIComponent** クラスはすべての CLIK コンポーネントの基礎となっていますので Scaleform は、カスタムのコンポーネントを作成される場合には、**UIComponent** からサブクラス化することをお勧めします。

UIComponent クラス自体は Flash 8 MovieClip クラスから派生したものなので、標準の Flash MovieClip のプロパティやメソッドをすべて継承しています。**UIComponent** クラスは以下のカスタ

ムの読み取り/書き込みプロパティをサポートします:

- **enabled;**
- **visible;**
- **focused;**
- **focusable;**
- **width;**
- **height;**
- **scaleX;**
- **scaleY;**
- **displayFocus** : コンポーネントが自らの focused ステートを表示する必要がある場合、このプロパティを true に設定します。詳細は「[フォーカスの処理](#)」のセクションを参照してください。
- **focusTarget**: Settings this property to another MovieClip will cause that MovieClip to receive focus rather than this component if focus is ever set to this MovieClip. For more information see the [Focus Handling](#) section.

UIComponent にはサブクラスが実装する予定の以下のような空のメソッドが含まれています:

- **configUI** : コンポーネントの構成を行うために呼び出されます。
- **draw** : コンポーネントが無効になった場合に呼び出されます。詳細は「[無効化](#)」の章を参照してください。
- **changeFocus** : コンポーネントにフォーカスが適用されたとき、またはフォーカスを失ったときに呼び出されます。
- **scrollWheel** : カーソルがコンポーネント上にあるときに、マウス ホイールを使用すると呼び出されます。

UIComponent は Mix-in を使用して EventDispatcher クラスを継承し、イベントのサブスクライブとディスパッチをサポートします。したがって、すべてのサブクラスはイベントのサブスクライブとディスパッチをサポートします。

詳細は「[イベントモデル](#)」の章を参照してください。

4.1.1 初期化

このクラスは以下の初期化の手順を、onLoad() イベント ハンドラの内部で行います:

- MovieClip のサイズにデフォルト サイズを設定する。
- コンポーネントの構成を行う。この手順は configUI() メソッドを呼び出します。
- コンテンツを描画する。この手順は validateNow() メソッドを呼び出します。このメソッドは再描画を直ちに行います。つまり、draw() を呼び出します。

4.2 コンポーネント ステート

ほぼすべての Scaleform CLIK コンポーネントは視覚的なステートをサポートします。ステートは、そのコンポーネントのタイムラインで特定のキーフレームに移動する、またはステート情報をサブメントに渡すことで設定されます。以下のような 3 種類の一般的なステートの設定があります。

4.2.1 ボタン コンポーネント

マウスの操作に反応したり、オプションで選択可能であったりする、ボタンのように動作するコンポーネントはすべてこのカテゴリーに分類されます。このスキーマを使った CLIK コンポーネントの例は、Button とそのバリエーションである ListItemRenderer、RadioButton、CheckBox です。ScrollBar などの複合コンポーネントのサブエレメントもこのボタン コンポーネントです。このカテゴリーに分類される CLIK コンポーネントは、直接 Button クラス (scaleform.clik.controls.Button) を使用するか、または Button クラスから派生したクラスを使用するか、のいずれかであることになります。

ボタン コンポーネントがサポートする基本ステートは以下のとおりです:

- *up* またはデフォルトのステート
- マウス カーソルがコンポーネント上にあるとき、またはフォーカスが適用されているときの *over* ステート
- マウス ボタンがコンポーネント上で押されたときの *down* ステート
- コンポーネントが無効になったときの *disabled* ステート

また、ボタン コンポーネントはプレフィックスのステートもサポートします。これは他のプロパティの値に応じて設定することができます。デフォルトでは、コア CLIK Button コンポーネントは "selected_" というプレフィックスしかサポートしません。このプレフィックスはコンポーネントが、選択された状態のときにフレーム ラベルに付加されます。

ボタン コンポーネントがサポートする基本ステートは、*selected* ステートも含めると、以下のとおりになります:

- *up* またはデフォルトのステート
- マウス カーソルがコンポーネント上にあるとき、またはフォーカスが適用されているときの *over* ステート
- マウス ボタンがコンポーネント上で押されたときの *down* ステート
- コンポーネントが無効になったときの *disabled* ステート
- *selected_up* またはデフォルトのステート
- マウス カーソルがコンポーネント上にあるとき、またはフォーカスが適用されているときの *selected_over* ステート
- ボタンが押下されたときの *selected_down* ステート
- *selected_disabled* ステート

注意: この章で説明しているステートは、CLIK ボタン コンポーネントがサポートする全ステートのはんの一握りです。ステートの全リストについては、「[Getting Started with CLIK Buttons](#)」を参照してください。

CLIK Button クラスは `getStatePrefixes()` メソッドを提供します。このメソッドを使うと、開発者はコンポーネントのプロパティに応じてプレフィックスのリストを変更することができます。このメソッドは以下のように定義されます:

```
protected function getStatePrefixes():Vector.<String> {
    return _selected ? statesSelected : statesDefault;
}
```

前頁に記載しているとおり、デフォルトでは CLIK のボタンは "selected_" というプレフィックスしかサポートしていません。getStatePrefixes() メソッドはその selected プロパティに応じて、別の配列のプレフィックスを返します。このプレフィックスの配列は、適切なステート ラベルと組み合わせて内部で使用され、再生するフレームを決定します。

ステートが内部で設定されると、たとえばマウスのロールオーバーのときは、ルックアップ テーブルが、フレーム ラベルのリストのために照会されます。Button クラスの stateMap プロパティは、フレーム ラベルのマッピングにステートを定義します。以下は CLIK Button クラスで定義されたステート マッピングです：

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

各ステートは複数のターゲット ラベルを備えている場合もあります。このステート マップから返される値を getStatePrefixes() が返すプレフィックスと組み合わせて、再生されるターゲット フレームのリストを作成します。以下の図は、再生する正確なフレームを決定する場合に使用される全プロセスを表したものです：

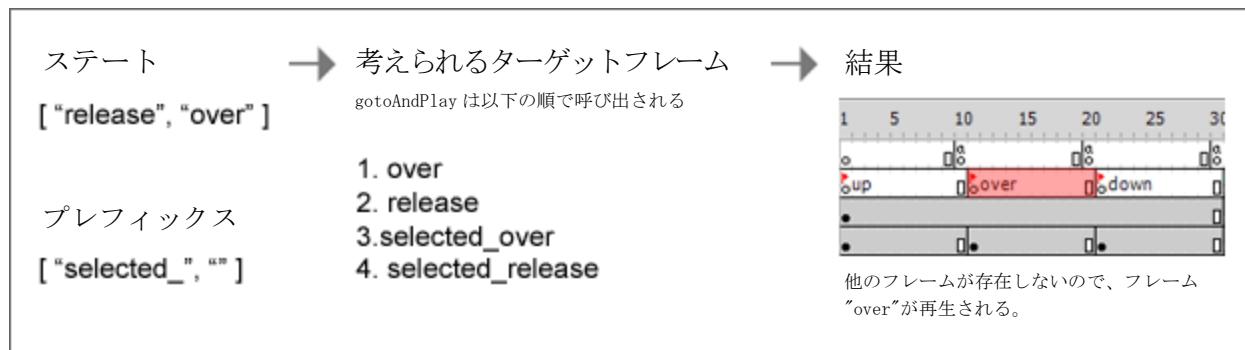


図 60: 再生される正しいキーフレームの決定に使用されるプロセス

再生ヘッドは常に最後の使用可能なフレームにジャンプするので、プレフィックスを持つ特定のフレームが使用できない場合、必然的に、コンポーネントは前回リクエストされたフレームをデフォルトにします。開発者はこのステート マップをオーバーライドして、カスタムの動作を作成することができます。

4.2.2 ボタン以外のインタラクティブ コンポーネント

このカテゴリーはインタラクティブで、フォーカスを適用することはできてもマウス イベントには応答しないコンポーネントをすべて含みます。このスキーマを使用する CLIK コンポーネントの例として、ScrollingList、OptionStepper、Slider、TextArea が挙げられます。このようなコンポーネントには、マウス イベントに応答する子エレメントが含まれている場合があります。ボタン以外のインタラクティブ コンポーネントがサポートするステートは以下のとおりです：

- *default* ステート
- *focused* ステート
- *disabled* ステート

4.2.3 非インタラクティブ コンポーネント

インタラクティブではなくても、無効にすることができるコンポーネントはすべてこのカテゴリーに含まれます。Label コンポーネントは、ステートをサポートするデフォルト コンポーネント セットの唯一の非インタラクティブ コンポーネントです。非インタラクティブ コンポーネントがサポートするステートは以下のとおりです：

- *default* ステート
- *disabled* ステート

4.2.4 特殊なケース

上記のステートルールに従わないコンポーネントもあります。StatusIndicator とそのサブクラスの ProgressBar はタイムラインを使ってコンポーネントの値を表示します。再生ヘッドはコンポーネントのパーセント値を表すフレームに設定されます。たとえば、最小値 0、最大値が 10 で現在の値が 5 (50%) に設定されている StatusIndicator の 50 フレームのタイムラインは、フレーム 25 (50 フレームの 50%) に gotoAndStop() されます。これらのコンポーネントを拡張して、表示をプログラム的に管理するのは非常に簡単です。この動作を変えるように updateValue() メソッドを変更、またはオーバーライドすることができます。

或る場合には、コンポーネントは、デフォルトの動作に加えて、追加のコンポーネント ステートをサポートするような特別なモードを持っていることもあります。TextInput と TextArea コンポーネントでは、**over** と **out** ステートが有効になる場合があります。これは、TextInput または TextArea コンポーネントの actAsButton プロパティが、フォーカスされていない時にマウスカーソルのロールオーバーとロールアウトをサポートするように設定されている場合です。

4.3 イベント モデル

Scaleform CLIK コンポーネント フレームは「イベント モデル」と呼ばれる通信パラダイムを使用します。コンポーネントは変更、または操作されるときに、イベントを「ディスパッチ」して、コンテナ コンポーネントは別のイベントにサブスクライブすることができます。これで複数のオブジェクトに変更を通知できるようになります。ActionScript 3 には実際堅牢なイベントシステムが備わっており、CLIK はこれを利用します。

ActionScript 3 Event システムに関する詳細な情報は次を参照してください。

http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/events/Event.html

4.3.1 使用法とベスト プラクティス

4.3.1.1 イベントのサブスクライブ

イベントをサブスクライブするには、リッスンする操作のタイプを指定するタイプ パラメータと共に addEventListener() メソッドを使用します。逆に、removeEventListener() はイベントからアンサブスクライブします。複数のリスナーが正確に同じパラメータと共に追加される場合、1 つのイベントのみが発生します。また、このようなメソッドはそれぞれ、リッスン オブジェクトであるスコープ パラメータと callBack も必要とします。これらの方法のそれぞれは、以前に定義された関数または Function 変数と関連づけられているタイプ Function のリスナーパラメーターを要します。

CLIK は様々なカスタムイベントを使用し、それぞれには独自のプロパティーがあつて、イベントに関するより詳細な情報を提供します。Event クラスから派生する標準 Clik イベントすべては scaleform.clik.events にあります。

addEventListener()もまたさらに次の 3 つのパラメーターをアクセプトし、これらにはデフォルト値がありますので注意してください。**useCapture**、**priority**、**useWeakReference**。一般的に、ユーザーはそのすべてのイベントリスナーが弱い参照を使う方が良く、これでオブジェクトへの強い参照が無ければガーベージ コレクトされてしまうオブジェクトの維持を回避します。リスナーが弱い参照を使用するようにするには、addEventListener()のシンタックスを次のようにしてください。

```
buttonInstance.addEventListener(ButtonEvent.CLICK, onButtonClick, false, 0, true);
```

addEventListener()パラメーターに関する詳細な情報は次を参照してください。

[http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/events/EventDispatcher.html#addEventListener\(\)](http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/events/EventDispatcher.html#addEventListener())

```
buttonInstance.addEventListener(ButtonEvent.CLICK, onButtonClick, false, 0, true);
function onButtonClick (e:Event):void {
    buttonInstance.removeEventListener(ButtonEvent.CLICK, onButtonClick, false);
}
```

この例では、このクラスは buttonInstance から発せられた ButtonEvent.CLICK イベントをリスンするリスナーを生成しています。このイベントが受け取られるといつでも onButtonClick 関数が実行されます。リスナー関数はその後イベントリスナーを削除します。

リスナーの Event パラメーターのプロパティーは Event のオリジンとタイプによって異なります。 Clik コンポーネントが作成するイベント オブジェクト (およびそのプロパティ) の全リストについては、「[プリビルド コンポーネント](#)」を参照してください。

4.3.1.2 イベントのディスパッチ

サブスクライブされたリスナーに変更や操作を通知したい CLIK コンポーネントは、`dispatchEvent()` メソッドを使用します。このメソッドは 1 つの引数を必要とします。それは、ディスパッチされるイベントのタイプを指定する必須タイプ プロパティを含めて、関連データが含まれたオブジェクトです。コンポーネント フレームワークは自動的に `target` プロパティを追加します。これはそのイベントをディスパッチするオブジェクトへのリファレンスですが、手動でカスタム ターゲットでオーバーライドするように設定することもできます。

```
dispatchEvent(new Event(Event.CHANGE));
```

4.4 ランタイムでのコンポーネントの生成

ActionScript 3 でランタイムで動的にコンポーネントを生成するには、次のシンタックスにしたがってください。

```
import flash.utils.getDefinitionByName;
import scaleform.clik.controls.Button;

public var myButton:Button;

// Class の定義への参照を取得する。
var classRef:Class = getDefinitionByName("ButtonLinkageID") as Class;
// クラスの新規インスタンスをインスタンシエートし、これを保存する。
if (classRef != null) { myButton = new classRef () as Button; }
myButton.addEventListener(ButtonEvent.CLICK, onButtonClick, false, 0, true);
```

このサンプルでは、FLA のライブラリにその Class を `ButtonLinkageID` に、Base Class を `scaleform.clik.controls.Button` に設定された Symbol があります。

FLA のライブラリに Class があって、Base Class のない Symbol がある場合、その Symbol のインスタンスは Symbol とクラスが結びつけられる時に単に Class のコンストラクターを使用して生成できます。

4.5 フォーカスの処理

Scaleform CLIK コンポーネントはカスタムのフォーカス処理フレームワークを使用します。これはほとんどのコンポーネントに実装され、非フレームワーク コンポーネントやシンボルとうまく動作するはずです。

CLIK FocusHandler マネージャー (`scaleform.clik.managers.FocusHandler`) はコンポーネント 1 つのクラスが生成されるとすぐにインスタンシエートされます。FocusHandler を直接インスタンシエートする必要はありません。

フォーカスの変更はすべて Scaleform プレーヤーレベルで起き、これはマウスまたはキーパッド（ゲームパッド）のフォーカス変更または ActionScript 経由で行われます。ActionScript でフォーカスを設定する方法には次のものがあります。

- myUIComponent.focused = 1; // Where 1 is a bit for the controller at index 0.
- stage.focus = myMovieClip; // As long as the MovieClip with this logic is on the Stage.
- FocusHandler.getInstance().setFocus(myMovieClip); // FocusHandler is a static class.

4.5.1 使用法とベスト プラクティス

フォーカスは `InteractiveDisplayObject` に設定しておく必要があります、そうしない場合にはフォーカスはプレーヤーとなります。設定されていない場合、フォーカス管理システムは正しく動作しません（たとえば、Tab キーはフォーカスを切り替えなくなります）。フォーカスは、コンポーネント上の `focused` プロパティを `true` に設定することで適用することができます。非コンポーネント エレメントでも動作するフォーカスの適用を行う他の方法は以下のとおりです：

`tabEnabled` と **`mouseEnabled`** のある MovieClips で `false` に設定されたものは Scaleform または Flash プレーヤーでフォーカスできず、フォーカス変更イベントを生成しません。ほとんどの CLIK コンポーネントは、デフォルトでそれ自身とその子供に対して `tabEnabled` と `mouseEnabled` プロパティーを設定します。

場合によっては、`tabEnabled` と `mouseEnabled` にしておきたいがフォーカスを受け取れないようにしておきたいコンポーネントのあることがあります。`UIComponent.focusable` プロパティーはこの問題に対処するもので、ここには AS3 に `MovieClip.focusEnabled` はありません。このプロパティーが `false` に設定されている場合、`UIComponent` は CLIK フォーカスフレームワークでフォーカスを受け取れません。**`focusable`** プロパティーの変更はコンポーネントの `tabEnabled` と `mouseEnabled` プロパティーに内部で影響を与え、これでそのコンポーネントに以前適用された `tabEnabled` と `mouseEnabled` 設定を維持しようとします。

`ScrollBar` などのフォーカス可能なサブエレメントを持つコンポーネントは、自らの `focusTarget` プロパティーを使って、自身のオーナー コンポーネントにフォーカスを渡します。フォーカスがコンポーネントに変更されると、`FocusHandler` は再帰的に `focusTarget` チェーンを検索して、`focusTarget` を返さない最後のコンポーネントにフォーカスを適用します。

場合によっては、コンポーネントが実際には Player やアプリケーションのフォーカスではないときに、そのコンポーネントをフォーカスが適用されているように表示する必要があります。`displayFocus` プロパティーを `true` に設定して、フォーカスが適用されているかのように動作することを、コンポーネントに指示することができます。たとえば、Slider にフォーカスが適用されているとき、Slider のトラックもフォーカスが適用されているように見える必要があります。`UIComponent.changeFocus()` メソッドが、`focused` プロパティーが変更されると、コンポーネント上に呼び出されるので注意してください。

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

逆に言えば、コンポーネントはクリックできる必要はあるけれども Tab 経由ではフォーカスを受け取ってはならないことがあります。たとえばドラッグできるパネルやその他の、マウスだけでコントロ

ールされる方が良いコンポーネントがこれに当たります。この場合、tabEnabled プロパティを false にしておくことができます。

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

4.5.2 複合コンポーネントのフォーカスの取得

複合コンポーネントは、ScrollingList、OptionStepper、またはButtonBar など、異なるコンポーネントで構成されたコンポーネントのことです。このコンポーネント自体はサブコンポーネントにマウス ハンドラを備えていても、自分自身のマウス ハンドラはいっさい持っていないことがよくあります。つまり Flash や Scaleform の Selection エンジンはアイテムにフォーカスを適用できず、内蔵のナビゲーション サポートはコンポーネントの識別に問題があり、誤ってその子コンポーネントを検証してしまうということになります。

複合であるにもかかわらず、単独のエンティティとしてそのコンポーネントに動作させるには、以下の手順を行います：

1. マウス ハンドラを持つすべてのサブコンポーネント (OptionStepper の矢印ボタンなど) で tabEnabled = mouseEnabled = focusable = false と設定します。
2. マウス ハンドラを持つすべてのサブコンポーネントで focusTarget プロパティを、コンテナ コンポーネントに設定します。必ず、コンテナ コンポーネントで focusable = true と設定します。
3. 必要な場合、サブコンポーネントの displayFocus プロパティを、コンテナ コンポーネントの changeFocus メソッド内で、true に設定します。

これで、サブコンポーネントにフォーカスが適用されると、そのフォーカスはコンテナ コンポーネントに移動するようになります。

4.6 入力処理

Scaleform CLIK コンポーネントは Flash 8 MovieClip クラスから派生しているので、ユーザーの入力を受け取ると、他の MovieClip インスタンスとほぼ同じように動作します。コンポーネントがマウス ハンドラを備えていれば、マウス イベントはコンポーネントがキャッチします。ただし、キーボード や同等のコントローラ イベントの処理方法に関しては、CLIK には大きな概念上の変更があります。

4.6.1 使用法とベスト プラクティス

マウス以外の入力はすべて、InputDelegate マネージャ クラス (scaleform.clik.managers.InputDelegate) がインターフェースします。InputDelegate は内部で、またはゲーム エンジンから入力コマンドの値をリクエストするかのいずれかで、入力コマンドを InputDetails オブジェクト (scaleform.clik.ui.InputDetails) に変換します。後者には、InputDelegate.readInput() メソッドをターゲット アプリケーションをサポートするように変更することが関係します。いったん InputDetails が作成されると、入力イベントは InputDelegate からディスパッチされます。

InputDetails は以下のプロパティで構成されます：

- 「キー」などのタイプ
- 押下されたキーのキー コードなどのコード
- ボタンのベクトルやキー押下のタイプなど入力についての追加情報である値。キーイベントは、キーが離される (key up) とキーが押される (key down) というアクションのどちらにも生成されます。また、相応する InputDetails のパラメータ値は、それぞれ、"keyUp"または "keyDown"となります。
- 「上」、「下」、「左」、「右」などのマッピングが可能な場合、そのような人が解読できる移動方向を定義する navEquivalent (ナビゲーション相当操作)。NavigationCode クラス (scaleform.clik.ui.NavigationCode) は、一般的なナビゲーション相当操作の便利な一覧を提供します。

FocusHandler は InputDelegate から発せられた InputEvents をリスンし、その後現在フォーカスされているコンポーネントから同じ InputEvent を再度発して、これをそのコンポーネントまたはそのイベントのバブルチェーンでの別のコンポーネントで処理するようにします。

このパスは handleInput() メソッドを実装した、表示リスト階層の上から順に並んだコンポーネントのリストです。

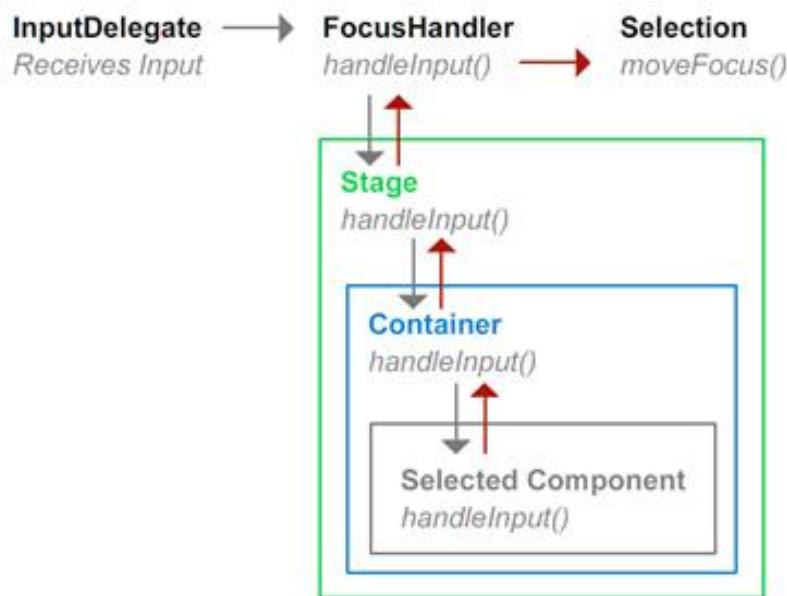


図 61: handleInput() チェーン

こうしておくと、そのイベントを受け取る他のコンポーネントが `event.isDefaultPrevented()` をチェックして、そのイベントがすでに処理されたかどうかを見極めることができます。

```

override public function handleInput(event:InputEvent):void {
    // Check whether the event has already been handled by another component.
    if (event.isDefaultPrevented()) { return; }
    var details:InputDetails = event.details;
  
```

```

        switch (details.navEquivalent) {
            case NavigationCode.ENTER:
                if (details.value == InputValue.KEY_DOWN) {
                    // Do something.
                    event.handled = true;
                }
                break;
            default:
                break;
        }
    }
}

```

InputEvent は、その.bubble プロパティが変更されていない限り、ActionScript 3 イベントシステムが自動的に bubble します。ScrollingList のように、場合によってはコンポーネントは InputEvent を自身で処理を試みる前にこれを子供にパスしたいことがあります。たとえば、Enter キーを押した場合に ListItemRenderer は ButtonEvent.CLICK イベントと List を生成せねばならず、これがそのイベントを発して、何も行いません。)

```

override public function handleInput(event:InputEvent):void {
    if (event.isDefaultPrevented()) { return; }
    // Pass on to selected renderer first
    var renderer:IListItemRenderer = getRendererAt(_selectedIndex, _scrollPosition);
    if (renderer != null) {
        // Since we are just passing on the event, it won't bubble, and should properly
stopPropagation.
        renderer.handleInput(event);
        if (event.handled) { return; }
    }
    ...
}

```

また、イベント リスナーを InputDelegate.instance に追加することで、手動で入力イベントをリッスンし、入力を以下のように処理することも可能です。入力はそれでも FocusHandler に取得され、フォーカス階層を下に移動するので注意してください。

```

InputDelegate.instance.addEventListener(InputEvent.INPUT, handleInput);
function handleInput(e:InputEvent):void {
    var details:InputDetails = e.details;
    if (details.value = Key.TAB) { doSomething(); }
}

```

InputDelegate は予想される入力を管理するために、ゲームに合わせて調整する必要があります。デフォルトの InputDelegate はキーボード コントロールを処理して、矢印キー、さらに一般的なゲームのナビゲーション キーである W、A、S、D を方向性のあるナビゲーションの相当操作に変換します。

4.6.2 複数のマウス カーソル

Wiiなどの一部のシステムは複数のカーソル デバイスをサポートします。CLIKコンポーネント フレームワークは複数のカーソルをサポートしますが、1つのSWFでは複数のコンポーネントにフォーカスを適用することはできません。二人のユーザーが別々のボタンをクリックした場合、最後にクリックされたアイテムにフォーカスが適用されます。

このフレームワークでディスパッチされたすべてのマウス イベントは、controllerIdx プロパティを含んでいます。これはそのイベントを作成した入力デバイスのインデックスです。

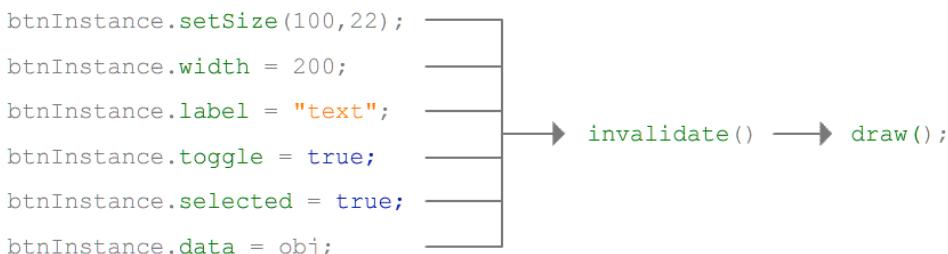
```
myButton.addEventListener(ButtonEvent.CLICK, onCursorClick);
function onCursorClick(event:ButtonEvent):void {
    trace(event.controllerIdx);
}
```

4.7 無効化

無効化は、複数のプロパティが変更になったときに、コンポーネントが再描画する回数を制限するメカニズムです。コンポーネントが無効化されると、自らを次のフレームに再描画します。これにより、開発者は一度に多くの変更をコンポーネントに適用して、一度だけコンポーネントを更新させることができます。コンポーネントが直ちに再描画する必要がある場合などわずかに例外もありますが、ほとんどの場合、無効化で十分です。

図 62: 特定のプロパティが変更になったときに、CLIKコンポーネントは自動的に無効化する。

4.7.1 使用法とベスト プラクティス



内部のコンポーネントの変更後に (通常 setter 関数が行います)、UIComponent.invalidate() を呼び出す必要があります。これは最終的にコンポーネントの UIComponent.draw() を呼び出します。

パフォーマンスを良くするために draw() 内部の倫理はサブセクションに分けられて、isValid() チェックでラップされており、これがコンポーネントのどのアスペクト (データ、サイズ、ステートなど) が変更されていないコンポーネントの部分の更新を避けるために現在無効であるかを識別します。

invalidate() が呼び出された場合、この方法はコンポーネントのすべてのアスペクトを無効にし、draw() での完全な再描画を強制的に行います。したがって、パフォーマンスを良好にするにはサブクラスは isValid() チェック内で draw() 論理をラップして同じパラダイムに従うべきで、invalidate() を呼び出し、invalidateData() や invalidateSize() のような特別な invalidate() 方法を使用して不必要的再描画を避けるべきです。

`invalidate()`の方法では、コンポーネントが複数の変更を 1 つの `draw()`呼び出しにバッチするために、次の段階の無効化 (Event.RENDER) 、または次のフレーム (Event.ENTER_FRAME) で `draw()` を呼び出します。既存のコンポーネントを使用している開発者は無効化を使う必要はありませんが、少なくとも認識しておくべきです。

コンポーネント内の何も以前に無効とマーク一付けされていなければ、`draw()` は呼び出されません。直ちに再描画する必要がある場合、開発者は `UIComponent.validateNow()` メソッドを使用することができます。

4.8 コンポーネントのスケーリング

Scaleform CLIK コンポーネントは以下の 2 つの方法でスケーリングを行います:

1. リフロー レイアウトを使用する。この場合コンポーネントのスケールはリセットされ、そのエレメントはオリジナルのサイズに合うようにサイズ変更されます。サブエレメントを含み、バックグラウンドのないコンポーネントはこの方法を使用します。
2. アスペクト比を維持するようにエレメントを逆にスケーリングする。この場合コンポーネントはスケーリングされたままですが、そのエレメントは逆にスケーリングされ、スケーリングされていないように見せます。この方法はグラフィック バックグラウンドと `scale9grid` を持つコンポーネントを拡大して、歪めることなく、コンテキストを内部でスケーリングすることができます。

一般的にコンポーネントには、Flash のスケール 9 グリッドに制約があることから、リフローの手法を使います。このため、開発者は Flash や Flex のコンポーネントと同じように、“スキン”のシンボルを作らなければなりません。リフローは、コンポーネントが同じようにスケールするサブコンポーネントを持っていれば上手く働きます。つまり、リフローは、コンテイナーやこれに似たエンティティに向いているということになります。

逆スケーリングの手法は、Scaleform が拡張的なスケール 9 グリッド機能を持っていることから、特に CLIK のためのようなものです。これは、ある種のコンポーネントに使われているようなレイヤーに集中したアプローチではなく、フレーム ステートを使った單一アセットを持つコンポーネントの作成に使うことができます。基本の CLIK コンポーネントは、もとより必要最小限に作られていて、通常は、背景一つ、ラベル一つ、オプションのアイコンまたはサブ・ボタン一つを持っています。そこで、逆スケーリングの手法にはぴったりなのです。しかしながら、逆スケーリングはコンテイナー的な組み立て（パネル レイアウト等）には適していません。こういった場合には、残りのサブエレメントと律則関係にあるような背景を一つ持ったリフロー手法をお薦めします。

コンポーネントは Flash IDE のステージでスケーリングすることも、`width` と `height` プロパティ、または `setSize()` メソッドを使ってダイナミックにスケーリングすることもできます。スケーリングされたコンポーネントの外観は、Flash IDE では正確に見えない場合もあります。これはコンポーネントを、LivePreview のない未コンパイルの MovieClip として渡してしまうことの限度です。スケーリングされたコンポーネントが、どのようにゲーム内で表示されるか正確な表現を得る唯一の方法は、Scaleform Player でこのムービーをテストすることです。CLIK エクステンションは、ワークフローのこの部分を向上する Launcher パネルをバンドルしています。

4.8.1 Scale9Grid

ほとんどのコンポーネントは 2 つ目のスケーリング方法を使用します。Scaleform Player では Scale9Grid 内部の MovieClip アセットは、大部分 scale9grid に従っていますが、Flash は MovieClip が含まれていればそのグリッドを破棄します。つまり、scale9grid が Flash Player で動作しない場合でも、Scaleform では完璧に動作する場合があるということです。

1 つ注意すべき点は、MovieClip に scale9grid が含まれている場合、サブエレメントもそのルールでスケーリングされるということです。Scale9grid をそのサブエレメントにも追加すれば、サブエレメントは親のグリッドを無視して、通常どおり描画するようになります。

4.8.2 Constraints

Constraints ユーティリティ クラス (scaleform.clik.utils.Constraints) は、スケーリングされるコンポーネント内のアセットのスケーリングと配置を支援します。このクラスを使うと開発者はステージ上にアセットを配置することはもちろん、アセットが親コンポーネントの端からの距離を維持するように保つことができます。たとえば、ScrollBar コンポーネントはトラックと下向き矢印ボタンを、ステージ上に置かれた位置に応じてサイズ変更し配置します。Constraints はコンポーネントで使用される両方のスケーリング メソッドで動作します。

以下のコードは ScrollBar アセットを configUI() メソッドの constraints オブジェクトに追加して、下向き矢印ボタンを最下部に揃え、トラックをスケーリングして親コンポーネントと共に拡大します。draw() メソッドにはこの constraints を更新するコードと、その結果登録されるエレメントが含まれています。この更新は draw() 内部で行われます。このメソッドはコンポーネントの無効化後、通常コンポーネントのサイズが変わった後で呼び出されるからです。

```
override protected function initialize():void {
    super.initialize();

    // The upArrow doesn't need a constraints, since it sticks to top left.
    if (downArrow) {
        constraints.addElement("downArrow", downArrow, Constraints.BOTTOM);
    }
    constraints.addElement("track", track, Constraints.TOP |
        Constraints.BOTTOM);
}

override protected function preInitialize():void {
    constraints = new Constraints(this, ConstrainMode.REFLOW);
}

protected function draw():void {
    constraints.update(_width, _height);
}
```

4.9 コンポーネントとデータのセット

データのリストが必要なコンポーネントは dataProvider による方法を使用します。dataProvider はデータの保管と取得のオブジェクトで、Scaleform CLIK IDataProvider クラス (scaleform.clik.interfaces.IDataProvider) で定義された API のすべて、または一部を公開します。

dataProvider による方法は直接プロパティにアクセスするのではなく、コールバックを伴なうリクエスト モデルを使用します。これにより dataProvider は必要に応じて、データのためにゲーム エンジンに接触することができます。これはメモリ上の利点と、大きなデータ セットを細かく、管理しやすいものに分ける機能を提供します。

dataProvider を使用するコンポーネントは、CoreList (ScrollingList、TileList)、OptionStepper、DropdownMenu を拡張するものが含まれています。

4.9.1 使用法とベスト プラクティス

フレームワークに含まれる DataProvider クラス (scaleform.clik.data.DataProvider) は静的 initialize() メソッドを使って、dataProvider メソッドを任意の ActionScript 配列に追加します。dataProvider を使用するコンポーネントは自動的に配列を初期化して、dataProvider による方法を使って自身にアクセスできるようにします。つまり、以下のシンタックスは静的に宣言された配列を、IDataProvider に記述されているメソッドで、完全に機能する dataProvider として初期化するという意味です：

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

dataProvider が実装するべきメソッドは以下のとおりです：

- *length* : プロパティ、または getter 関数のいずれかとして、データ セットの長さを返します。
- *requestItemAt* : dataProvider の特定の項目をリクエストします。OptionStepper など、常に 1 つの項目を表示するリスト コンポーネントが、通常は使用します。
- *requestItemRange* : 開始と終了インデックスで dataProvider の項目の範囲をリクエストします。ScrollingList など、複数の項目を表示するリスト コンポーネントが、通常は使用します。
- *indexOf* : 項目のインデックスを返します。
- *invalidate* : 変更があると、dataProvider にフラグを付け、データの新規の長さを提供します。また、このメソッドは "dataChange" イベントをディスパッチして、データが更新されたことをコンポーネントに通知しなければなりません。dataProvider は、公的にアクセス可能で、データ サイズを反映する length プロパティをサポートしなければなりません。

データの短いリストを必要とするインスタンスは、dataProvider として配列を使用することができます。開発者は ActionScript 2 のデータの保管が、アプリケーションにネイティブで保管するよりも、メモリとパフォーマンスの点では非常に負荷が高いということに気付くべきです。大型のデータ セットの場合、dataProvider を、ExternalInterface.call クラスにつなぎ、必要に応じてゲーム エンジンからデータをポーリングすることをお勧めします。

4.10 ダイナミック アニメーション

Scaleform CLIK には、Flash 8 Tween クラスに似た動作を備えていますが、完全に Scaleform と互換性があるカスタムの Tween クラス (scaleform.clik.motion.Tween) があります。どちらの Tween クラスも mx.transitions.easing.* パッケージのものなど、同じタイプのイージング関数をサポートします。

Scaleform CLIK にはカスタムの Tween クラス (scaleform.clik.motion.Tween) があり、このビヘイビアは他の同等の Tween クラスと類似していますが、Scaleform に完全に準拠しています。CLIK Tween は fl.transitions.easing.* package の下にあるような標準のイージング関数をすべてサポートしています。

Tween を開始するには、新規の Tween() を生成し、長さ（ミリ秒単位）、ターゲット Sprite、Tween するプロパティ、Tween すべき値、その他の Tween パラメーターとなるプロパティを持った Object を Tween に与えます。次のプロパティー/パラメーターがサポートされています。

- *ease*: 関数タイプ。イージング関数。
- *easeParam*: Object タイプ。イージング関数にパスできるその他のパラメーター。
- *onComplete*: 関数タイプ。このクロージャーは Tween 終了時に呼び出される。
- *onChange*: 関数タイプ。このクロージャーは Tween がその値を更新するときに呼び出される（各ティック/フレーム）。
- *data*: Object タイプ。Tween にアタッチしたい任意のカスタムデータ（これは Tween のビヘイビアには何ら影響を与えません）。
- *nextTween*: Tween タイプ。別の Tween にチェーンしたい場合に使用。チェーンされた Tween は現在の Tween 終了時に pause=false に設定されます。
- *frameBased*: ブール値タイプ。リアルタイムではなく、フレームベースのタイミングを使用してください。
- *delay*: Number タイプ。.Tween を x ミリ秒遅らせます。
- *fastTransform*: ブール値。ディスプレオブジェクトプロパティにマトリックス数学を使用します。
- *paused*: ブール値タイプ。Tween が一時停止されたかどうか。

ただし、これらのトゥイーン メソッドは MovieClip 毎に複数のプロパティをサポートして、同じオブジェクトのさまざまなプロパティを同時に変更できるようにします。次の章の例は、同時に複数のプロパティに影響するトゥイーンの作成方法を示しています。

4.10.1 使用法とベスト プラクティス

Tween のインスタンスは、ターゲットの MovieClip を MovieClip の現在のプロパティ値を、関数パラメータリストに指定した値に tween します。

Tween 終了時に通知を受けるには、Tween のコンストラクターの 4 つめのパラメーターとしてパスされている Object の onComplete プロパティとして関数参照を生成、追加してください。このコールバックは Tween 終了時に呼び出され、これを呼び出した Tween に 1 つのパラメーターをパスします。

```
import fl.transitions.easing.*;
import scaleform.clik.motion.Tween;

// Perform a 1 second tween from the current horizontal position and
```

```
// alpha values to the ones specified
var myTween:Tween = new Tween(300,
    myMovieClip,
    { x:100, y:100, alpha:0 },
    { ease:Strong.easeOut,
    onComplete:onCompleteCallback} );

function onCompleteCallback(t:Tween):void { // Do something. }
```

4.11 レイアウトフレームワーク

CLIK レイアウトフレームワークはデベロッパーが複数の要素のある動的なレイアウトを生成し、簡単に複数の解決に適合させる援助をします。いくらか単純ですが、CLIK レイアウトフレームワークは複数の解決とプラットフォームに対する UI を生成するデベロッパーを援助し、CLIK 内でカスタムレイアウトシステムを開発する基礎を提供するはずです。

注:Scaleform v4.0.14 は Clik AS3 レイアウトフレームワークとクラスの最初にリリースで、API は変更される可能性があります。CLIK AS3 レイアウトフレームワークについてご質問、問題、ご提案、コメントがございましたら、ご遠慮なく Scaleform Developer Center 経由でお問い合わせください。

次の 2 つのクラスが Clik レイアウトフレームワークのコアを形成します。

- **scaleform.clik.layout.Layout:**同じ親の下で .layoutData プロパティーを適切に実装する任意の Sprite のレイアウトを管理するレイアウト。
- **scaleform.clik.layout.LayoutData:**特定の Sprite がどのようにレイアウトされるべきかに関する情報を含むデータ構造。このデータは関連する Layout インスタンスで読み込まれ、レイアウトの生成に使用されます。

CLIK では新規レイアウト生成には複数の方法があります。最も簡単な新規レイアウト生成方法は次の通りです。

1. Sprites/MovieClip を生成し、これらを Stage で DisplayObjectContainer に加える。
2. これらの Sprites/MovieClip の .layoutData プロパティーを、LayoutData クラスの新規インスタンスで定義する。
3. 各 Sprite/MovieClip に対する LayoutData オブジェクトを好きなように入れる。
4. Layout クラスの新規インスタンスを生成する。ステップ 1 で Sprites/MovieClip が使用したのと同じ DisplayObjectContainer にこれを加える。

このシナリオでは、Layout インスタンスのプロパティーが何も変更されなければ、Layout は、.layoutData プロパティーを配置決めのために親の幅と高さを四角形 (x、y、幅、高さ) を用いて定義する親の中のすべての Sprite と MovieClip を管理します。

SDK にはレイアウトフレームワークの 2 つのデモが Scaleform¥Resources¥AS3¥CLIK¥demos¥ディレクトリにあります。最初のデモ、Layout_Main は Scaleform SDK Browser で開くことのできる Clik Component Browser あります。Layout_Main はサイズ変更可能な MovieClip 内での簡単な

Layout の生成をデモします。もう一つの Layout_FullScreen_Demo は、Stage のフルサイズを使えるように FxPlayer のスタンドアロン インスタンスで実行されるように設計されています。Layout_FullScreen_Demo は、フルスクリーンのコンテンツを複数の解決でどのように Layout できるか、また Layout システムが様々なパラメーターをどのように処理するかをデモします。

4.11.1 Layout

Layout はサイズやスケールの変化するコンテンツ内で、要素のレイアウトに使用できるコンポーネントです。Layout は、同じ親内に存在する Sprite（とそのサブクラス）のみを管理できます。つまり、Layout インスタンスそのものの子供としては要素を追加してはならず、その Layout インスタンスと同じレベルで追加しなければなりません。たとえば、同じレベルで 2 つの子供を持つ

(myWindow.myMovieClip と myWindow.layoutInstance) と、layoutInstance は myMovieClip を管理できます。

Layout は LayoutData クラスのインスタンスで.layoutData プロパティーを定義した Sprite のみを管理します。複数の Layout が同じ親に存在することもできますが、これには各 Layout に独自の.identifier プロパティー セットがあり、同じレベルの Spritit 内のすべての LayoutData もまたそれらの.layoutIdentifier プロパティーを定義する必要があります。

Layout が Stage に加えられたとき、Layout は親の下にあるすべての子供を見、これらに定義済みの.layoutData があるかどうかをチェックします。ここから、Layout は親に加えられた新しい子供を追跡して、これらの子供が Stage に加えられる前に.layoutData を定義しておれば、その Layout の管理する Sprite/MovieClip のリストに加えます。管理された Sprite/MovieClip のオフセットと配置順序は、これらが Stage に最初に加えられるときにだけ更新されることにご注意ください。更新された位置からオフセットを再度計算し直したい場合、またはリストを再度並べ替えたい場合には、次の関数を使用してください。

public function reset():void

これは Layout の管理する Sprite のリストをクリアし、LayoutData のある新しい Sprite を探し、これらの新しい Sprite を一からオフセット/フローを計算し直してリセットします。

パブリック関数 resortManagedSprites():void

管理された Sprite のリストを、レイアウトの適用される順序を定義するその layoutData.layoutIndex プロパティーで並べ替えます。この関数は、Layout の初期設定後に何らかの layoutIndex が変更された場合に呼び出します。

ユーザーは Flash ファイルのライブラリの Symbol に Layout クラスを付加でき、アーティストがデザイン時にレイアウトを Stage に配置できるようにできます。Layout が Symbol に結びつけられておらず（しかしその代わりにコードを使用して幅と高さを 0 にしてインスタンシエートされている）場合、その Layout のサイズはデフォルトでは親のサイズに設定されます。これは Layout の.rect プロパティーを設定して変更可能で、このプロパティーは Layout がその管理する要素のレイアウトに使用するエリアを定義します。

理想的には、Layout は親の最後の要素として加えられるべきで、こうするとその他の Sprite/MovieClip 用の LayoutData すべてがすでに適切に定義されているようになります。

4.11.1.1 パブリックプロパティー

rect	レイアウトの「サイズ」。内部の要素はこのプロパティーの x、y、幅、高さにしたがってレイアウトされます。Layout の width != 0 (Symbol に結びつけていて、Stage 上に配置されている) の場合、これは MovieClip の x、y、幅、高さを使用します。Layout の width == 0 (addChild())、ただしバックする Symbol なし) の場合、これは親の x、y、幅、高さを使用します。.rect プロパティーを使用してカスタムの Rectangle を割り当てることも可能です。
tiedToStageSize	この Layout のサイズがステージのサイズにマッチするように常に更新されておれば true、それ以外は false。
tiedToParent	この Layout のサイズが親のサイズにマッチするように常に更新されておれば true、それ以外は false。
hidden	この Layout がランタイムに非表示にされるべき場合は true、それ以外は false。これは Layout が、ランタイムで直ちに visible = false に設定されるような目に見える背景またはプレースホルダー画像を持つことを可能にします。

4.11.2 LayoutData

特定の Sprite に対するレイアウトを定義するデータ。有効な scaleform.clik.layout.Layout インスタンスと共に使用する必要があります。

4.11.2.1 パブリックプロパティー

alignH	この Sprite に対する水平配置。有効値 : LayoutMode.ALIGN_NONE、LayoutMode.ALIGN_LEFT、LayoutMode.ALIGN_RIGHT。
alignV	この Sprite に対する垂直配置。有効値 : LayoutMode.ALIGN_NONE、LayoutMode.TOP、LayoutMode.BOTTOM.property から TextFieldAutoSize.NONE はサイズ変更なし。
offsetH	Layout または relativeToH Sprite の端からの水平オフセット。これが変更されない (-1) 場合、Layout はアーティストがデザイン時に定義する、Stage 上の Layout/Sprite からの元の水平オフセットに自動的に設定します。
offsetV	Layout または relativeToV Sprite の端からの垂直オフセット。これが変更されない (-1) 場合、Layout は、アーティストがデザイン時に定義するような Stage 上の Layout/Sprite からの元の垂直オフセットに自動的に設定します。
offsetHashH	様々なアスペクト比に対するユーザーの定義する水平オフセットのハッシュテーブル。これらの値は Layout がステージのサイズに結びつけられている場合に(Layout.tiedToStageSize == true) クエリされます。たとえば、 "LayoutData.offsetHashH[LayoutData.ASPECT_RATIO_4_3] = 70;" は、Layout が Stage のサイズに結びつけられており、アスペクト比が現在 4:3 の場合に、この Sprite が 70px の水平オフセットを使用するようにします。
offsetHashV	様々なアスペクト比に対するユーザーの定義する垂直オフセットのハッシュテーブル。これらの値は Layout がステージのサイズに結びつけられている場合に(Layout.tiedToStageSize == true) クエリされます。

	たとえば、 "LayoutData.offsetHashV[LayoutData.ASPECT_RATIO_4_3] = 70;"は、Layout が Stage のサイズに結びつけられており、アスペクト比が現在 4:3 の場合に、この Sprite が 70px の垂直オフセットを使用するようにします。
relativeToH	この Sprite が水平基準であるべきだという Sprite のインスタンス名。null 値に置いておいた場合、この Sprite はレイアウト基準に配列されます。
relativeToV	この Sprite が垂直基準であるべきだという Sprite のインスタンス名。null 値に置いておいた場合、この Sprite はレイアウト基準に配列されます。
layoutIndex	この Sprite が同じ Layout 中の他の Sprite を基準にしてレイアウトされるべき順序を支配するインデックス。relativeToH または relativeToV を使用する場合は、Sprite の Layout がこの Sprite の前に更新されるよう設定してください。layoutIndex を変更なし (-1) のままにしておく場合、これはリストの最後に任意に付加されます。
layoutIdentifier	1 つの DisplayObjectContainer の中に複数の Layout が存在する場合に、この LayoutData オブジェクト中にどの Layout が関連づけられるべきかを定義する String。変更なし (null 値) のままにしておくと、同じ親の中のすべての Layout で自動的に管理されます。このプロパティが設定されている場合、これは Layout インスタンスの識別子のプロパティーにマッチする必要があります。

4.12 ポップアップのサポート

Scaleform CLIK には PopUpManager クラス (scaleform.clik.managers.PopUpManager) が含まれていて、ダイアログやツールチップなどのポップアップをサポートします。Dialog コンポーネントはこの PopUpManager を使って自らのコンテンツを表示します。

4.12.1 使用法とベスト プラクティス

PopUpManager は複数の静的メソッドを備えており、ポップアップの作成と管理を支援します。The createPopUp() メソッドは任意の MovieClip シンボル (CLIK コンポーネントを含みます) へのリンク ID を使って、ポップアップを作成する場合に使用できます。最初のパラメーターは、ユニークなリンク ID に対する String です。2 番目のパラメーター、initProperties は、新たな PopUp が生成されたときに Object のプロパティーのコピーされるような Object をユーザーが提供できるようにします。3 つめと 4 つめのパラメーターは、新しいポップアップに対してそのポップアップの親を基準にして、それぞれ x と y 座標を定義します。relativeTo パラメーター、Sprite タイプは別の Sprite を基準にしたポップアップの配置に使用できます。

```
import scaleform.clik.managers.PopUpManager;
PopUpManager.createPopUp("PopupLinkageID", {alpha:.5}, 100, 100
myMovieClip);
```

Scaleform エクステンション InteractiveObjectEx.setTopmostLevel()を使って、ポップアップが常にステージ上で他のすべてのエレメントの上に表示されることが保証されるようになります。このスキームは、ライブラリ スコープに関する問題のため、ルート レベルでポップアップを作成する代わりに採用されました。子 SWF が親の内部にロードされていて、親のコンテキストに存在するパスのライブラリで定義されたシンボルのインスタンスを作成しようとした場合、親は子のライブラリにアクセスできないので、シンボル ルックアップ エラーが発生します。残念ですが、この問題をうまく回避する方法はなく、topmostLevel スキームが最善の方法です。

InteractiveObjectEx.setTopmostLevel()はポップアップ以外のものにも適用できます。ステージのこのようなエレメントの Z オーダーが、維持されます。したがって、ポップアップの上にカスタムのカーソルを描画するために、そのカーソルを最も高い深度、またはレベルで作成して、その topmostLevel プロパティを true に設定することができます。

4.13 ドラッグ&ドロップ

DragManager クラス (scaleform.clik.managers.DragManager) は、ドラッグ操作の開始と管理をサポートします。このクラスはシングルトンと同様に動作し、唯一のオブジェクトにアクセスするインスタンス プロパティを提供します。このオブジェクトを使用すると、開発者はドラッグ操作の開始/停止を行うことができます。

ドラッグを開始するには、新しいドラッグを基準にしたプロパティーで DragEvent.DRAG_STATE を発します。DragManager.handleStartDragEvent()はこのイベントを受け取り、このイベントのプロパティーに基づいたドラッグを開始します。このドラッグはその後 MouseEvent.MOUSE_UP を受け取るまで DragManager が管理し、DragManager.handleEndDragEvent()が呼び出されます。

4.13.1 使用法とベスト プラクティス



Figure 63: DragDemo in action.

DragManager は、UI を通して使用できる IDragSlots との効率的な通信のために関数を定義する IDragSlot インターフェイスに依存します。DragManager はドラッグの管理を行うだけです。この DragManager を使って、本当の意味でドラッグ & ドロップをサポートするコンポーネントを作成するかどうかは開発者次第です。

Scaleform CLIK にはサンプルの IDragSlot 実装が同梱されており、これは DragSlot (scaleform.clik.controls.DragSlot) と呼ばれています。

この DragSlot は UIComponent を拡張するので、CLIK コンポーネントとして分類されます。このクラスは DragManager を補完する固有の機能を含んでいます。DragSlot はドラッグされる MovieClip/Sprite/Bitmap を含むように設計されています。しかし、DragSlot はまた多くの Button のビヘイビアを模倣し、これが DragSlot を Button としても使用することを可能にしています。

DragSlot にはドラッグ タイプのコンセプトが含まれています。このようなドラッグ タイプは、コンポーネントがドロップ操作を許可/拒否できるように DragSlot ごとに構成することができます。このようなドラッグ タイプを補完するためには、DragSlot は dragBegin と dragEnd のイベント リスナーも DragManager にインストールします。これにより DragTarget は、ドロップ操作を受け入れるか否かに関わらず、表示することができます。

究極的には DragSlot は単なる IDragSlot のサンプル実装に過ぎず、その多くの関数はスタブアウトされて、データのバックエンドとより緊密に通信するサブクラスに置き換えられています。完全に機能する DragSlot サブクラスのサンプルについては、MMO UI Kit を参照してください。ここでは DragSlot を UI を通してドラッグできるコンテンツのベースクラスとして用いています。

5 使用例

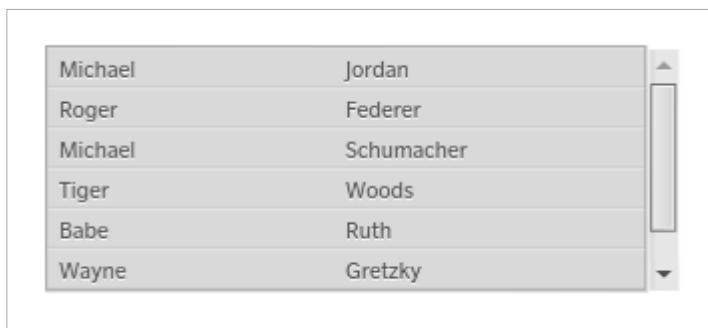
これ以降の章には、Scaleform CLIK コンポーネントを使って実装された使用例のサンプルが含まれています。

5.1 基礎編

以下のサンプルは範囲と複雑さにおいては単純ですが、一般的なタスクを実行するための Scaleform CLIK フレームワークの使い勝手の良さを表しています。

5.1.12 つの textField を備えた ListItem Renderer

図 64: 2 つのラベルを含むリスト項目を表示した ScrollingList



この ScrollingList コンポーネントは、デフォルトで ListItemRenderer を使って行内容を表示します。ただし、ListItemRenderer は 1 つの textField しかサポートしません。リスト項目が複数の textField を表示するケースはいくつもあります。さらにアイコンなどの textField 以外のリソースを表示することもあります。このサンプルは 2 つの textField をリスト項目に追加する方法を表しています。

まず、必要条件を定義します。目的は 2 つの textField をサポートするカスタムの ListItemRenderer を作成することです。また、このカスタムの ListItemRenderer は ScrollingList と互換性があるべきです。計画ではリスト項目に付き 2 つの textField を持つ予定なので、データも单一文字列のリストだけではないはずです。このサンプルに以下のdataProvider を使用してみます：

```
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"}];
```

このdataProvider には 2 つのプロパティ fname と lname を持つオブジェクトが含まれています。これらの 2 つのプロパティは 2 つのリスト項目 textField に表示されます。

デフォルトの ListItemRenderer は 1 つの textField しかサポートしないので、2 つの textField をサポートする機能が必要となります。これを実現する最も簡単な方法は、ListItemRenderer クラスをサブクラス化することです。以下は MyItemRenderer というクラスに対するソース コードです。これは ListItemRenderer をサブクラス化して、2 つの textField の基本的なサポートに追加します。(このコードを作業中の FLA と同じディレクトリの *MyItemRenderer.as* というファイルに追加します):

```
import scaleform.clik.controls.ListItemRenderer;

class MyItemRenderer extends ListItemRenderer {

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

    public function MyItemRenderer() { super(); }

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

    override protected function updateAfterStateChange():void {
        super.updateAfterStateChange();
        textField1.text = data ? data.fname : "";
        textField2.text = data ? data.lname : "";
    }
}
```

ListItemRenderer の setData と updateAfterStateChange メソッドは、このサブクラスでオーバーライドされます。setData メソッドは、リスト項目がそのコンテナ リスト コンポーネント (ScrollingList、TileList など) から項目のデータを受け取るときは常に呼び出されます。ListItemRenderer では、このメソッドは 1 つの textField の値を設定します。MyItemRenderer は代わりに両方の textField の値を設定して、内部でその項目オブジェクトへのリファレンスも保管します。この項目オブジェクトは updateAfterStateChange で再利用されます。このメソッドは ListItemRenderer のステートが変更になると常に呼び出されます。このステートの変更によって、textField は自らの値の更新が必要になる場合があります。

ここまででは、複雑なリスト項目を持つdataProvider と、そのリスト項目のレンダリングをサポートする ListItemRenderer を定義してきました。すべてをこのリスト コンポーネントと組み合わせるために、シンボルを作成してこの新規の ListItemRenderer クラスをサポートしなければなりません。このサンプルの場合、これを実現する最速の方法は、ListItemRenderer シンボルを 'textField1' と 'textField2' の 2 つの textField を持つように変更することです。次に、このシンボルの識別子とクラスを MyItemRenderer に変更する必要があります。このリストで MyItemRenderer コンポーネントを使用するために、リスト インスタンスの itemRenderer 検証可能プロパティを 'ListItemRenderer' から 'MyItemRenderer' に変更します。

この FLA を実行します。リストは、dataProvider に設定された fname と lname プロパティという 2 つのラベルを表示するリスト エレメントを含んで表示されるはずです。

5.1.2 ピクセル単位のスクロール表示

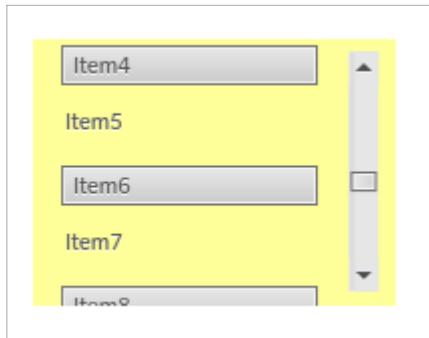


図 65: CLIK エレメントを含むコンテンツを持ったピクセル単位のスクロール表示

ピクセルごとのスクロールは、複雑なユーザー インターフェイスでは一般的な使用事例です。このサンプルは、CLIK ScrollBar コンポーネントを使ってこれを簡単に実現する方法を示しています。

まず始めにスクロール表示の新規のシンボルを作成します。これは、コンテンツのオフセットの算出に必要とされる計算を簡素化する便利なコンテナを提供します。このスクロール表示コンテナ内で、以下のレイヤーを上から下への順序でセットアップします。これらのレイヤーは必須ではありませんが、分りやすくするためにお薦めします：

- *actions* : スクロール表示を動作させる ActionScript コードが含まれています。
- *scrollbar* : CLIK ScrollBar のインスタンスが含まれています。このインスタンスを 'sb' とします。
- *mask* : 四角形、または MovieClip で定義されたマスク レイヤーです。レイヤー プロパティもマスクに設定します。
- *content* : スクロールするコンテンツが含まれます。
- *background* : マスクされていない領域を強調表示するバックグラウンドを含んだオプションのレイヤーです。

コンテンツ レイヤーに適切なエレメントを追加して、そのエレメントをすべて保有するシンボルを作成します。コンテンツをすべて保有するシンボルを作成することで、コンテンツをスクロールするというタスクがさらに容易になります。このコンテンツ インスタンスを 'content' として、その y 値を 0 に設定します。これはこのスクロール ロジックがさまざまなオフセットを明らかにする必要がないということを保証します。ただし、このようなオフセットが、作成するスクロール表示の複雑さに応じて、必要になる場合もあります。

ここまででは、簡単なスクロール表示の作成に必要な構造と、合わせて組み込む必要のある指定エレメントを定義してきました。code レイヤーの最初のフレームに以下の ActionScript コードを加えます：

```
// 133 is the view size (height of the mask)
sb.setScrollProperties(1, 0, content.height - 133);
```

```
sb.position = 0;

sb.addEventListener(Event.SCROLL, onScroll, false, 0, true);
function onScroll(e:Event) {
    content.y = -sb.position;
}
```

このスクロールバーは別のコンポーネントには接続されていないので、手動で構成する必要があります。setScrollProperties メソッドを使って、このスクロールバーをある位置単位でスクロールし、完全にコンテンツを表示するための最小値と最大値を設定します。この場合のスクロールバーの位置はピクセル単位です。このファイルを実行します。スクロール表示は、スクロールバーで操作することで変化するようになっているでしょう。

6 よく寄せられる質問

1. Flash Studio から発行すると、次のエラーメッセージが出ます。

メッセージ：

1152:A conflict exists with inherited definition scaleform.clik... in namespace public.

このエラーメッセージは、クラスの要素が 2 回定義されているために出ます。このエラーメッセージは FLA が "Automatically Declare Stage Instances" を無効にするように設定されていない場合に起き、これがどれかの ActionScript の定義と、FLA が発行されたときに自動的に生成される定義との間に競合を起こします。

CLIK アーキテクチャーはすべての Clik コンポーネントが正しく機能するためにはこの設定を無効にしておくことを必要としており、これは Stage 要素を明示的に定義することは、ActionScript クラスの厳密に型指定された子供に対しては通常必要（またこれがベストプラクティス）だからです。

FLA に対して "Automatically Declare Stage Instances" を無効化するには、File -> Publish Settings -> "Flash" tab -> Actionscript 3.0 Settings を開いてください。この際 "Automatically declare stage instances" のチェックは外しておいてください。この設定は Flash Studio のグローバルな設定ではなく、特定の FLA に対してのみ使用されることに注意してください。

2. Clik コンポーネントを別の.FLA のライブラリからインポートしています。そのコンポーネントのインスタンスを Stage に配置してその検査可能なプロパティーを変更すると、次のエラーが出ます。

Code:

1046:Type was not found or was not a compile-time constant:...

このエラーはコンポーネントの宣言が適切ではない場合に最も良く起ります。このエラーは一般的に次のステップで解決できます。

- コンポーネントに有効なインスタンス名が付いているようにする。
- コンポーネントが ActionScript で定義されているようにする。
- ActionScript の定義が正しいクラスを参照しているようにする。場合によって、これはコンポーネントのエクスポート名では無いことがあります。その代わり ActionScript クラス名を使用してください。たとえば、myButton:Button (scaleform.clik.controls.) を、myButton:MyButtonSymbol の代わりに使ってください。ここに MyButtonSymbol はインポートされたシンボルのエクスポート名、myButton はコンポーネントのインスタンス名です。

3. ライブラリ内に同じ Base Class から来る 2 つのコンポーネントがあります。SWF ファイルを発行すると、次のエラーメッセージが出ます。

Symbol 'MySymbol', Layer 'actions', Frame 10, Line 1 -- 1024:Overriding a function that is not marked for override.

Symbol 'MySymbol', Layer 'actions', Frame 20, Line 1 -- 1024:Overriding a function that is not marked for override.

Symbol 'MySymbol', Layer 'actions', Frame 30, Line 1 -- 1024:Overriding a function that is not marked for override.

これらのエラーは通常 Symbol の Class 名がその Base Class 名と同じであることから来ています。たとえば、Class が“Button”である Symbol と Base Class “scaleform.clik.controls.Button”が挙げられます。この場合この Symbol のタイムライン定義が Base Class とマージされます。そのため、同じ Base Class を共有する他のどの Symbol もこうしたタイムライン定義を継承し、望ましくないビヘイビアを引き起こしてしまいます。

この問題を解決するには、複数の Symbol の関連する Base Class があるかどうかを確かめ、Symbol の Class 名で Base Class の名前とマッチするものが無いようにしてください。同じ名前を共有する Symbol の Class を変更するとこの問題は解決します。

7 潜在的な危険

1. Symbol の Class を、他の Symbol が決してその Base Class を拡張することが無い限り、その Base Class と同じように名付けることを避ける。Symbol がその Base Class と同じ Class 名を持つ場合、そのタイムライン定義は Base Class とマージされ、これがこの Base Class のその他のサブクラスについて問題を生ずることがあります。

そのクラスにその他のサブクラスが存在することが無い場合、Base Class を Class として使用し、その Symbol に対しては Base Class を提供しないことを推奨します。こうすることで Base Class と Symbol は緊密に結合され、その Base Class のインスタンスを新たにインスタンシエートすると、Symbol は getDefinitionByName().のようなルックアップを使用すること無く自動的にこれに結びつけられます。

8 CLIK AS3 vs. CLIK AS2

This section outlines major differences, mostly programming related, between CLIK AS2 and CLIK AS3.

1. CLIK AS3 is designed to work in Flash Player.
2. Rather than using a custom EventDispatcher, CLIK AS3 uses ActionScript 3's native event system.
3. The invalidation system has been redesigned to help avoid unnecessary updates in draw(). Each component now stores an "invalid" table that tracks which aspects of the component are currently invalid. Default invalidation types are defined in scaleform.clik.constants.InvalidationType. Aspects can be marked as invalid using invalidate(), invalidate[InvalidationType](), or manually setting the property to true within the table.. Logic within draw() should be wrapped within an appropriate isInvalid() check to ensure that unnecessary updates to components are avoided.
4. invalidate() no longer uses a 1ms delay; instead, it uses the next stage invalidatation (Event.RENDER) or the next Event.ENTER_FRAME for the component and then calls validateNow().
5. Tween's syntax is no longer MovieClip.TweenTo; instead, var t:Tween = new Tween();
6. handleInput() is now tied to the native event system. This means that handleInput() now accepts one parameter of type InputEvent and the event will bubble up from the component that dispatched it rather than being passed down from FocusHandler.
7. handleInput() no longer returns a Boolean; instead, it should set event.handled = true; if the event was handled.
8. UIComponent has a new property, focusable, which can be used with the CLIK FocusManager to prevent focus from reaching a component. This is more or less a replacement for AS2's MovieClip.focusEnabled property.
9. enableInitCallback is no longer an inspectable of UIComponent. Instead, it should be set within a class's constructor or preInitialize() method.
10. DataProvider now requires that you provide the target array to DataProvider's constructor (myComponent.dataProvider = new DataProvider(myArray));, rather than simply setting myComponent.dataProvider = myArray;

11. ButtonBar will now only create Buttons that fit within its own size. If your ButtonBar's size is 200px wide and each Button is 150px wide, only one Button will be displayed. This allows users to resize their ButtonBar and have the Buttons within be added and removed dynamically.
12. UILoader has been removed. This functionality is now handled by the Flash MovieClip/Sprite and Loader classes.
13. UIComponent.SoundMap, introduced in CLIK 3.3, has been removed for CLIK AS3.