

Autodesk® Scaleform®

Scaleform CLIK AS3 User Guide

이 문서는 스케일폼 CLIK 프레임워크 및 이와 함께 제공되는 프리빌트 컴포넌트에 대한 상세한 구현설명을 포함하고 있다.

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

Table of contents

1	소개	1
1.1	개요	1
1.1.1	스케일폼 CLIK 에 포함된 것	1
1.2	UI 고려사항	2
1.3	컴포넌트 이해하기	3
1.3.1	점검가능 속성	4
1.4	프레임워크 기본	5
1.4.1	이벤트	5
1.4.2	Focus	6
2	프리빌트 컴포넌트(Prebuilt Components)	7
2.1	기본 버튼과 텍스트 타입(Basic Button and Text Types)	7
2.1.1	Button	9
2.1.2	CheckBox	15
2.1.3	Label	21
2.1.4	TextInput	24
2.1.5	TextArea	30
2.2	그룹 타입(Group Types)	34
2.2.1	RadioButton	35
2.2.2	ButtonGroup	41
2.2.3	ButtonBar	43
2.3	스크롤 타입(Scroll Types)	47

2.3.1	ScrollIndicator	47
2.3.2	ScrollBar	51
2.3.3	Slider	54
2.4	리스트 타입(List Types)	58
2.4.1	NumericStepper.....	60
2.4.2	OptionStepper	63
2.4.3	ListItemRenderer.....	67
2.4.4	ScrollingList	73
2.4.5	TileList	80
2.4.6	DropdownMenu	87
2.5	프로그레스 타입(Progress Types)	95
2.5.1	StatusIndicator.....	95
2.6	기타 다른 타입(Other Types)	97
2.6.1	Window.....	98
3	아트 세부설명	101
3.1	가장 좋은 연습	101
3.1.1	픽셀퍼펙트 이미지(Pixel-perfect Images).....	101
3.1.2	마스크(Masks)	102
3.1.3	애니메이션(Animations)	103
3.1.4	레이어와 그리기 기본요소(Layers and Draw Primitives)	103
3.1.5	복잡한 스킨	103
3.1.6	2 의 제곱	103
3.2	알려진 이슈와 추천하는 워크플로우	104
3.2.1	컴포넌트 복제	104
3.3	스키닝 예제	106
3.3.1	StatusIndicator 스키닝	107
3.4	폰트와 Localization.....	109
3.4.1	개요	110
3.4.2	폰트 내장	110
3.4.3	textField 에 폰트 내장	110

3.4.4	Scaleform Localization System	111
4	프로그래밍 세부사항.....	112
4.1	UIComponent	112
4.1.1	초기화.....	113
4.2	컴포넌트 상태	114
4.2.1	Button 컴포넌트	114
4.2.2	버튼이 아닌 상호작용 컴포넌트	116
4.2.3	비 상호작용 컴포넌트.....	116
4.2.4	특수한 경우	116
4.3	이벤트 모델.....	117
4.3.1	사용예와 가장 좋은 연습(Usage and Best Practices).....	117
4.4	실행 시 구성 요소 생성	119
4.5	포커스 다루기	119
4.5.1	사용예와 가장 좋은 연습(Usage and Best Practices).....	120
4.5.2	복합 컴포넌트에서 포커스 캡처	121
4.6	입력 다루기	121
4.6.1	사용예와 가장 좋은 연습(Usage and Best Practices).....	121
4.6.2	다중 마우스 커서	124
4.7	무효화.....	125
4.7.1	사용예와 가장 좋은 연습(Usage and Best Practices).....	125
4.8	컴포넌트 크기조절	126
4.8.1	Scale9Grid	127
4.8.2	Constraints	127
4.9	컴포넌트와 데이터 셋(Components and Data Sets).....	128
4.10	동적 애니메이션.....	129
4.10.1	사용예와 가장 좋은 연습(Usage and Best Practices).....	130
4.11	레이아웃 프레임워크	130
4.11.1	레이아웃	132

4.11.2	LayoutData	133
4.12	팝업 지원.....	135
4.12.1	사용예와 가장 좋은 연습(Usage and Best Practices)	135
4.13	끌어 놓기.....	136
4.13.1	사용예와 가장 좋은 연습(Usage and Best Practices)	136
5	사용예	138
5.1	기초	138
5.1.1	2 개의 textField 를 가진 ListItem 렌더러.....	138
5.1.2	픽셀별 스크롤 뷰	140
6	자주 묻는 질문.....	142
7	잠재적인 위험.....	144

1 소개

이 문서는 스케일폼 Common Lightweight Interface Kit (CLIK™)프레임워크와 컴포넌트에 대한 상세한 구현 가이드를 제공한다. 이 문서를 읽기 전에 일단 [Getting Started with Clik](#) 과 [Getting Started with Clik Buttons](#) 문서를 독파할 것을 강력하게 추천한다. 이들 2 개의 소개문서는 스케일폼 Clik 을 단계별로 설정하고 실행하는데 필요한 안내서다. 따라서 Clik 의 기본 코어 컨셉과 Clik 컴포넌트를 만들고 스킨을 입하는 상세한 학습을 제공해준다. 하지만, 고급 사용자라면 소개 문서를 학습하면서 이 문서를 참고하면 더 깊은 이해를 할 수 있을 것이다.

1.1 개요

스케일폼 Clik 은 액션스크립트 2.0(AS2) 사용자 인터페이스 요소, 라이브러리, 워크플로우 개선 툴 등을 통해서 Scaleform 4.0 사용자가 빠르게 콘솔, PC 그리고 모바일 게임에 화려하고 효율적인 UI 를 구현하게 해준다. 이 프레임워크의 주요 목적은 (CPU 와 메모리상에서)가볍고, 쉽게 스킨을 입힐 수 있고, 고수준의 커스터マイ즈가 가능하도록 하는 것이다. UI 코어 클래스 및 시스템의 기본 프레임워크 외에도 Clik 은 주로 UI 를 위한 프레임워크로 생각하는 것이 좋다. 하지만, 15 개 이상의 프리빌트 일반 인터페이스 요소(예, 버튼, 슬라이더, 텍스트 입력)와 함께 제공되기 때문에 개발자로 하여금 빠르고 쉽게 인터페이스를 만들고, 이들 인터페이스에 대한 빠른 반복 테스트를 할 수 있게 해준다.

1.1.1 스케일폼 Clik 에 포함된 것

컴포넌트: 확장성이 뛰어나고 간단한 UI 컨트롤

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

Classes: Core system APIs

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

Documentation and Example Files:

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 고려사항

좋은 UI의 제작을 향한 첫걸음은 컨셉을 종이나 마이크로소프트 비지오(Microsoft Visio®)와 같은 비주얼 도면 편집기(visual diagram editor)를 사용해 먼저 계획을 잡아보는 것이다. 두 번째는 플래시로 프로토타입을 만들어서 특정한 그래픽 디자인 작업에 들어가기 전에 인터페이스 옵션과 작업 흐름을 테스트하는 것이다. CLIK은 사용자들이 이러한 빠른 프로토타이핑과 완성에 필요한 반복을 할 수 있도록 하기 위해 디자인 되었다.

전면 UI의 구조를 잡는 것은 다양한 방법이 있다. 플래시에서는 비지오(Visio)나 기타 플로우차트 프로그램 같은 페이지 컨셉이 존재하지 않는다. 대신, 키프레임과 MovieClip이 이를 대신한다. 만약 모든 페이지가 동일한 플래시 파일에 있다면, 더 많은 메모리가 필요할 것이다. 하지만 페이지간에 전환은 더 빠를 것이다. 만약 각 페이지가 분리된 파일에 있다면 전체 메모리 사용량은 줄어들겠지만 로딩 타임이 길어질 것이다. 또한, 각 페이지를 분리해서 다루게 되면 여러명의 디자이너와 아티스트가 동일한 인터페이스에 대하여 동시에 작업할 수도 있다. 기술과 디자인적 조건 외에도 UI 프로젝트의 구조를 잡을 때는 업무의 흐름(workflow)도 함께 고려해야만 한다.

일반적인 데스크탑이나 웹어플리케이션과는 달리, 멋진 멀티미디어 게임 인터페이스를 만드는 방법이 매우 다양하다는 것을 이해하는 것은 매우 중요하다. 모든 엔진이나, 혹은 여러 플랫폼에 따라서는, 살짝 다른 접근 방식을 취함으로써 더 낫게(혹은 더 나쁘게) 사용될 수 있다. 예를 들어서 하나의 플래시 파일에 다중 페이지 인터페이스를 넣는 경우를 생각해보자. 이렇게 하면 관리도 쉽고, 페이지간 이동도 쉬울 것이다. 하지만 메모리 사용량이 늘어나 구형 콘솔 혹은 모바일 기기에서는 부정적인 효과를 야기할 수도 있다. 단일 플래시 파일에 모든 것이 들어가 있다면 여러 명의 디자이너가 동시에 인터페이스의 다른 파트를 작업하는 것도 불가능하다. 만약 프로젝트가 복잡한 인터페이스를 필요로 하며, 대규모의 디자인 팀과 기타 특정 기술적 혹은 디자인적 요구사항들이 있다면, UI의 각 페이지를 별도의 플래시 파일로 관리하는 것이 좋을 것이다. 예를 들면, 화면이 요청에 따라 로드되었다가 언로드가 되는 경우 혹은 동적으로 업데이트가 되어 네트워크에 스트리밍이 되는 경우가 있을 수도 있다. 결과적으로 UI에 대한 에셋 관리는 UI의 레이아웃(layout)부터 업무흐름 적용 그리고 성능고려까지 항상 충분히 생각해봐야 할 필요가 있다.

스케일폼은 개발자들에게 대부분의 UI 작동을 플래시와 액션스크립트를 사용하기를 권장하지만 여기에는 완벽한 정답은 없으며 어떤 팀들은 어플리케이션 엔진의 스크립팅 언어(루아 혹은 언리얼스크립트)를 사용하여 무거운 짐의 대부분을 해결하는 것을 선호할 수도 있다. 이러한 경우에 플래시는 최소한의 액션스크립트와 플래시 파일 자체 내에서의 상호작용만 사용하여 최소한의 액션스크립트로 구성된 애니메이션 제공 툴로 주로 사용될 것이다.

개발자는 기술, 디자인 그리고 업무흐름에 필요한 사항들을 초기에 파악하는 것이 중요하다. 그 다음, 프로세스 전반에 대한 전제적인 접근 방법, 특히 프로젝트가 너무 깊숙히 진행되기 전에 지속적으로 검토하여 부드럽고 성공적인 결과를 얻을 수 있도록 하여야 한다.

1.3 컴포넌트 이해하기

일단, 개발자는 플래시 컴포넌트가 정확히 무엇인지 이해하는 것이 중요하다. 플래시는 컴포넌트라고 불리우는 기본 인터페이스 생성 도구와 빌딩블럭을 제공한다. 하지만, 이 문서에서 컴포넌트라고 할 때는 스케일폼에서 세계적으로 유명한 gSkinner.com 팀과 협작하여 개발한 CLIK 프레임워크를 사용해서 만들어진 프리빌트 컴포넌트를 말하는 것이다.

Grant Skinner 가 이끌고 있는 gskinner.com 은 플래시 CS4 의 컴포넌트를 만들 권한을

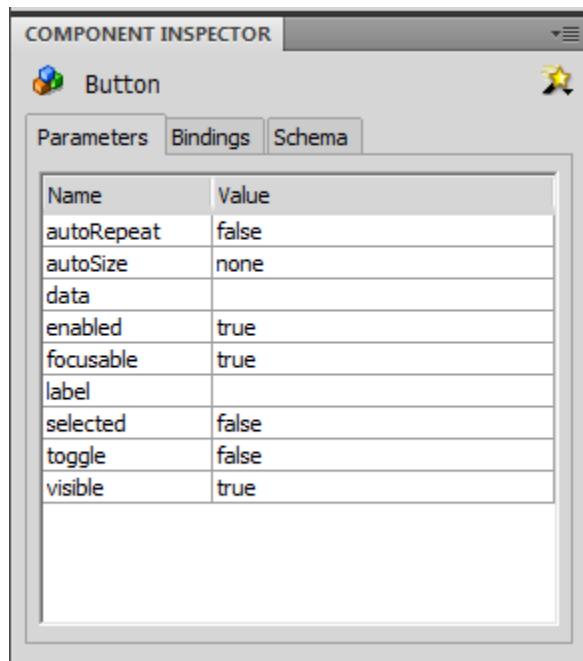
어도브로부터 받아 제작한 사람이며 세계 유수의 Flash team 중 하나로 알려져있다.

gskinner.com 에 대한 더 자세한 내용은 <http://gskinner.com/blog> 을 방문하기 바란다.

CLIK의 프리빌트 컴포넌트에 대한 더 깊은 이해가 필요하다면 Flash Studio 있는 CLIK 기본 팔렛을 열어보라. *Scaleform SDK\Resources\AS3\CLIK\components\CLIK_Components_AS3.fla*

1.3.1 점검가능 속성

컴포넌트의 중요한 속성은 플래시 IDE 컴포넌트 점검 패널(component inspector panel)의 파라메터 탭에서 설정 가능하다. CS4에서 이 패널을 열려면 상단 툴바에서 Window 드롭다운 메뉴를 눌러서 Component inspector panel을 클릭해서 사용 가능하도록 설정하면 된다. 물론 그냥 [Shift+F7]을 눌러도 된다. 이 패널의 점검가능 속성(inspectable properties)은 액션스크립트 프로그래밍에 익숙하지 않은 아티스트와 디자이너에게 컴포넌트의 작동과 기능을 설정하도록 해준다.



이들 속성을 변경하는 것은 컴포넌트를 포함한 SWF 파일을 퍼블리싱 한 다음에 알 수 있을 것이다. 플래시 IDE 스테이지에서 디자인 할 때는 CLIK 컴포넌트가 컴파일 된 클립이 아니기 때문에 변경사항이 눈에 보이지 않는다. 이는 컴포넌트에 손쉽게 접근하고, 스킨을 입히기 위해서 필요한 기능이다.

1.4 프레임워크 기본

1.4.1 이벤트

대부분의 구성 요소는 사용자 상호 작용, 상태 변경 및 focus 관리에 대한 이벤트를 생성합니다. 이러한 이벤트는 CLICK 구성 요소를 사용하여 가능한적인 사용자 인터페이스를 생성하는 데 중요합니다.

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- ***type***: 이벤트의 형식입니다.
- ***target***: 이벤트 대상입니다.
- ***currentTarget***: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- ***eventPhase***: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- ***bubbles***: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- ***cancelable***: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

ActionScript 3 이벤트 클래스에 대한 자세한 정보는 Adobe 의 ActionScript 3 참조 설명서에서 찾을 수 있습니다.

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

각 구성 요소에 의해 생성된 모든 이벤트의 목록은 이 사용자 설명서의 “미리 구축된 구성 요소” 섹션에서 해당 이벤트에 대한 고유한 프로퍼티에 따라 찾을 수 있습니다.

Scaleform 내에서 특정 내장 이벤트 형식은 Scaleform 익스텐션으로 대체됩니다. 예를 들어 Extensions.enabled 프로퍼티가 true 로 설정되면 Scaleform 은 항상 표준 MouseEvent 대신에 MouseEvent 이벤트를 확장하는 클래스인 MouseEventEx 를 생성합니다. MouseEventEx 는 다중 컨트롤러 및 오른쪽/중간 마우스 버튼 지원을 위한 프로퍼티를 추가합니다. 사용자는 수신한 이벤트가

MouseEventEx 인스턴스이고 그래서 이벤트 오브젝트를 익스텐션 형식에 보낸 경우에 확인할 수 있습니다.

1.4.2 Focus

FocusFlash 내에 focus 개념이 있습니다. 기본적으로 스테이지 focus 는 마우스 또는 키보드로 선택한 마지막 InteractiveObject 로 설정됩니다.

일반적으로, 포커스가 있는 컴포넌트만 키보드 이벤트를 받는다. 컴포넌트에 포커스를 설정하는 방법은 많다. 이 문서의 “프로그래밍 세부사항”에서 여러 가지 방법에 대하여 설명하고 있다. 대부분의 CLIK 컴포넌트는 마우스 좌측버튼이나 동등한 컨트롤이 컴포넌트 위에서 클릭하는 등의 상호작용이 발생하면 포커스를 받는다. TAB 과 Shift+TAB 키 (혹은 동등한 네비게이션 컨트롤)는 출력중인 컴포넌트 간에 포커스를 이동시킨다. 이는 대부분의 데스크탑 어플리케이션과 웹사이트에서의 행동과 동일하다. 비 CLIK 요소가 사용하는 포커스는 CLIK 컴포넌트에 의해서 사용됨을 주목하라. 이 말은 플래시 개발자가 CLIK 요소와 CLIK 이 아닌 요소 들을 동일한 쓴에 섞어서 사용할 수 있다는 뜻으로, 쓴의 포커스 사이를 이동하는 용도로 TAB 과 Shift+TAB 을 이용할 때 의도적으로 포커스를 갖게 할 수 있다는 것이다.

기본적으로 Button 은 엔터와 스페이스바 키에 반응한다. 마우스 커서를 Button 상에 있게 하거나 바깥으로 이동시키는 등의 행동은 컴포넌트에 영향을 미치며, 이는 마우스 드래그하는 경우도 마찬가지다.

콘솔 혹은 모바일에서는 개발자가 게임패드 컨트롤을 키보드나 마우스에 쉽게 맵핑할 수 있다. 예를 들면, Enter 키는 일반적으로 Xbox 360 혹은 PS3 콘솔 컨트롤러의 A 나 X 버튼에 맵핑한다. 이러한 맵핑은 CLIK 으로 제작한 UI 가 다양한 플랫폼에서 작동할 수 있도록 해준다.

2 프리빌트 컴포넌트(Prebuilt Components)

첫눈에 보면 CLIK 이 기본적인 프리빌트 UI 컴포넌트만 제공하는 것처럼 보인다. 하지만, CLIK 은 조금 더 화려한 컴포넌트와 인터페이스를 만드는 프레임워크를 제공하는 것이 진정한 목적이다. 개발자들은 (확장을 포함해서)자유롭게 CLIK 프레임워크에 기반한 커스텀 컴포넌트를 자신의 용도에 적합하게 만들 수 있다.

프리빌트 CLIK 컴포넌트는 기본 버튼부터 체크박스 그리고 리스트 박스나 드롭다운 메뉴, 모달 대화상자와 같은 복합 컴포넌트까지 표준 UI 기능 집합만 제공한다. 개발자들은 손쉽게 이를 표준 컴포넌트를 확장해서 기능을 추가하거나 커스텀 컴포넌트를 밑바닥부터 새로 만들 때 간단히 참고할 수 있다.

다음 장에서는 각각의 프리빌트 컴포넌트에 대하여 상세히 설명하도록 하겠다. 이들은 복잡도와 기능별로 그룹화 하였다. 각 컴포넌트는 다음과 같은 그룹으로 묶어서 설명할 것이다.

- **사용자 상호작용:** 사용자가 어떻게 컴포넌트와 상호작용할 것인가
- **컴포넌트 설정:** 플래시 저작환경에서 컴포넌트를 구축할 때 필요한 요소들
- **상태:** 컴포넌트가 기능하기 위해서 필요한 서로 다른 비쥬얼 상태(키프레임)
- **점검가능 속성:** 플래시 저작환경에 노출되어 있어서 코드를 사용하지 않고 손쉽게 특정 값을 설정할 수 있는 속성
- **이벤트:** 컴포넌트에 의해서 발생되어 UI 상의 다른 오브젝트에 전달될 수 있는 이벤트 리스트
- **팁과 트릭:** 컴포넌트와 관련하여 다양한 역할을 수행할 때 유용한 샘플코드

2.1 기본 버튼과 텍스트 타입(Basic Button and Text Types)

기본 형은 Button, CheckBox, Label, TextInput, TextArea 로 구성되어 있다. Button 은 대부분의 사용자 인터페이스의 뼈대를 구성하며, CheckBox 는 Button 으로부터 기능을 상속받는다. TextInput 과 TextArea 가 단일행과 다중행 textField 형인데 비해서, Label 은 정적 레이블형이다.



그림 1: *Mass Effect*의 Main Menu Button 예제

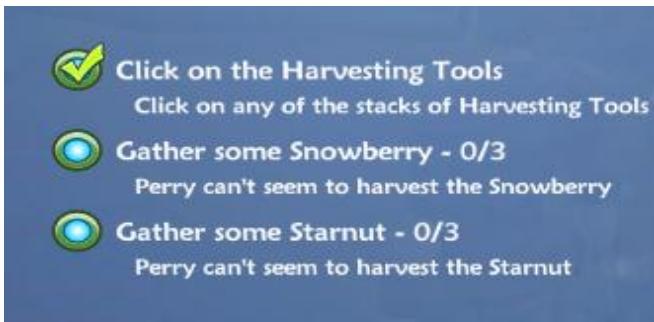


그림 2: *Free Realms*의 CheckBox 예제



그림 3: *Free Realms* 의 Label 예제

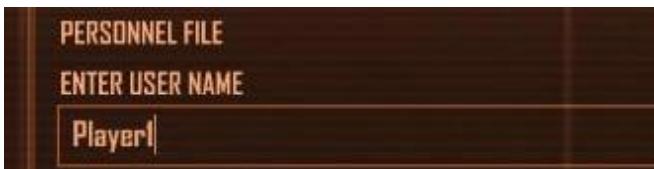


그림 4: *Crysis Warhead* 의 TextInput 예제

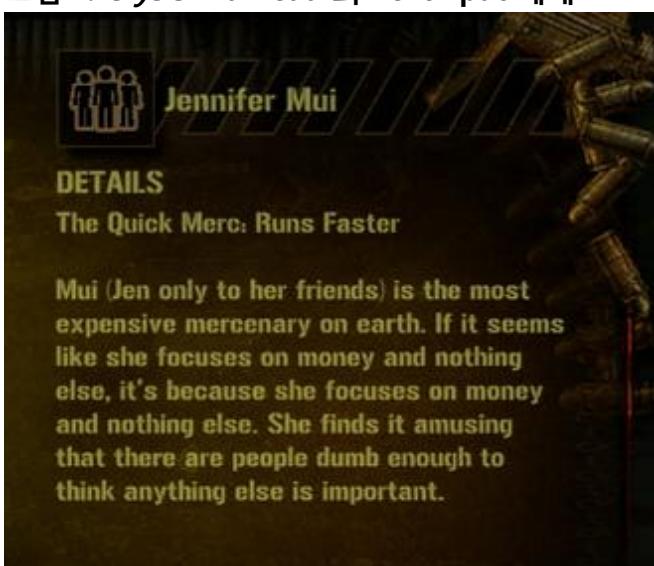


그림 5: *Mercenaries 2*의 TextArea 예제

2.1.1 Button



그림 6: 스킨 없는 버튼

CLIK 프레임워크의 기본 컴포넌트는 Button이며 클릭 가능한 인터페이스 컨트롤이 필요한 경우라면 언제든지 사용된다. 기본 Button 클래스(scaleform.clik.controls.Button)는 레이블 출력을 위해서 textField를 지원하며, 사용자 상호작용에 대하여 상태(stats)를 지원한다. Button은 직접 사용되거나 복합컴포넌트의 일부분으로 사용되는데, ScrollBar의 화살표나 Slider의 thumb이 대표적인 예다. 클릭으로 작동하는 대부분의 상호작용 컴포넌트는 Button으로 구성되거나 Button을 확장한 것이다.



그림 7: AnimatedButton, AnimatedToggleButton, ToggleButton

CLIK Button은 범용 버튼 컴포넌트이므로 마우스, 키보드 상호작용, 상태 등의 기타 기능을 제공함으로써 다용도 사용자 상호작용을 지원한다. 또한, 토글 기능과 애니메이션 상태를 지원한다. Button.fla 컴포넌트 소스파일의 ToggleButton, AnimatedButton, AnimatedToggleButton은 동일한 기본 클래스를 사용한다.

2.1.1.1 사용자 상호작용

Button 컴포넌트는 마우스나 기타 동등한 컨트롤러에 의하여 눌려진다. 포커스가 있을 경우에는 키보드에 의해서 눌릴 수도 있다.

2.1.1.2 컴포넌트 설정

CLIK Button 클래스를 사용하는 MovieClip 은 다음과 같은 부요소를 갖고 있어야만 한다.

- **textField**. (optional) TextField 타입. 버튼 레이블
- **focusIndicator**. (optional) MovieClip 형. 분리된 MovieClip 이 포커스 된 상태 출력에 사용된다. 실제 존재하는 경우에는 MovieClip 이 show 와 hide 라 명명된 프레임을 갖고 있어야 한다. 기본적으로 over 상태는 포커스 된 Button 컴포넌트를 표현할 때 사용된다. 하지만, 몇몇 경우에 아티스트는 포커스 된 상태와 마우스가 위에 있는 (mouse over)상태를 분할하고자 할 때 상당히 불편할 수 있다. focusIndicator 부요소는 이런 경우에 유용하다. 이 부요소를 추가하면 Button 상태가 포커스 상태에 영향을 받지 않는다. 컴포넌트 상태에 대한 더 자세한 정보는 다음 장을 참고하라.

2.1.1.3 상태

CLIK 버튼 컴포넌트는 사용자 작용에 근거한 서로 다른 상태를 지원한다. 이들 상태는 다음과 같다.

- **up** 혹은 기본 상태
- **over** 상태는 마우스 커서가 위에 있을 때, 혹은 포커스를 가졌을 때.
- **down** 상태는 버튼이 눌렸을 때
- **disabled** 상태

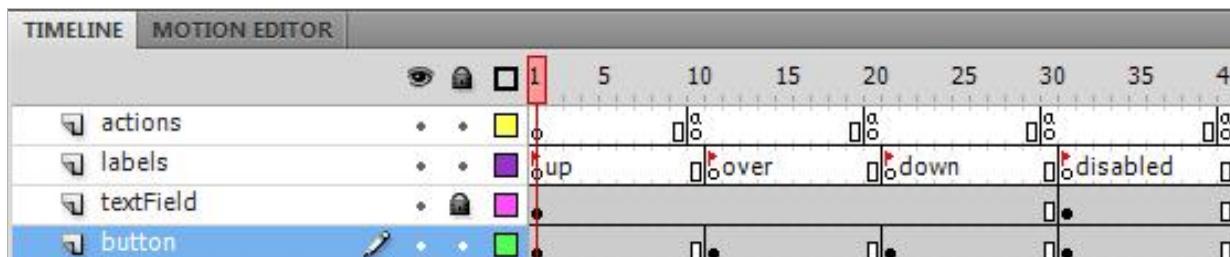


그림 8: Button 타임라인

이들 상태는 플래시 타임라인의 키프레임으로 나타난다. 또한, 이 상태들은 Button 컴포넌트가 정확히 작동하기 위한 최소한의 키프레임이다. 이들 상태 외에도 기능 확장을 위하여 복잡한 사용자 상호작용이나 애니메이션 전이(transition)를 처리하는 상태도 있다. 이에 대한 [세부자료는 Getting Started with CLIK Buttons](#) 문서를 참고하라.

2.1.1.4 점검가능 속성

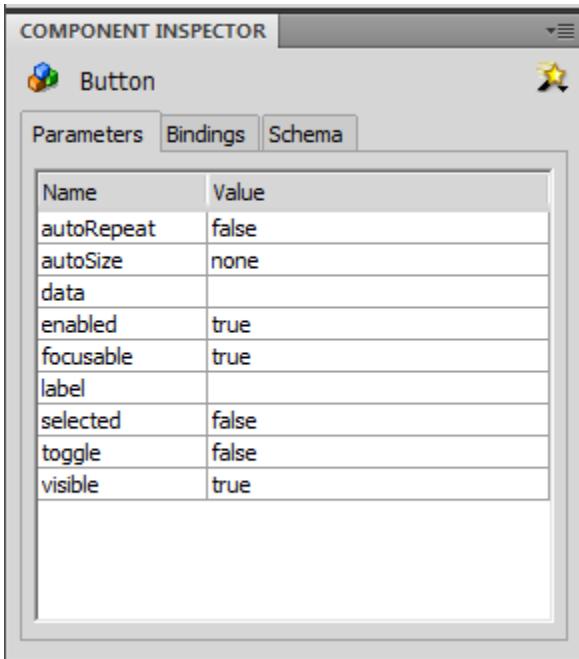


그림 9: CS4 component inspector 내의 Button 컴포넌트 점검가능 속성

Button 컴포넌트의 점검가능 속성은 다음과 같다.

autoRepeat	버튼을 누르고 유지할 때 "click" 이벤트를 보낼지 여부를 결정합니다.
autoSize	포함된 텍스트에 맞춰서 버튼 크기가 조절되고, 크기가 조절된 버튼을 정렬할 방향을 결정합니다. autoSize 프로퍼티를 TextFieldAutoSize.NONE 으로 설정하면 크기를 변경하지 않고 그대로 놔둡니다.
data	버튼 관련 데이터입니다. 이 프로퍼티는 버튼을 ButtonGroup 에서 사용할 때 특히 도움이 됩니다.
enabled	false 로 설정하면 버튼을 비활성화합니다. 비활성화한 구성 요소는 더 이상 사용자 입력을 받지 않습니다.
focusable	구성 요소에 대한 focus 관리를 활성화/비활성화합니다. focus 가능한 프로퍼티를 false 로 설정하면 tab 키, 방향 키 및 마우스 기반 focus 변경에 대한 지원을 제거합니다.

label	버튼의 라벨을 설정합니다.
selected	버튼의 선택된 상태를 설정합니다. 버튼은 선택과 비선택의 두 가지 마우스 상태 세트를 가질 수 있습니다. 버튼의 전환 프로퍼티가 true 일 때 선택된 상태는 버튼을 클릭하면 변경되지만, 선택된 상태는 전환 프로퍼티가 false 인 경우에도 ActionScript 를 사용해서 설정될 수 있습니다.
Toggle	버튼의 토글 모드를 설정한다. True 인 경우 토글버튼으로 작동한다.
Visible	false 인 경우 버튼을 숨긴다.

2.1.1.5 이벤트

모든 콜백은 단일 Event or Event subclass 인자를 받는데, 여기에는 이벤트에 관한 관련정보가 들어 있다. 다음 속성은 모든 이벤트에 대한 공통속성이다.

- **type**: The type of event.
- **target**: The event target.
- **currentTarget**: The object that is actively processing the Event object with an event listener.
- **eventPhase**: The current phase in the event flow. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- **bubbles**: Indicates whether an event is a bubbling event.
- **cancelable**: Indicates whether the behavior associated with the event can be prevented.

Button 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이기 속성(visible property)이 실행시 true 로 설정되었다.
ComponentEvent.HIDE	보이기 속성(visible property)이 실행시 false 로 설정되었다.
ComponentEvent.STATE_CHANGE	구성 요소의 상태가 변경되었습니다.
FocusHandlerEvent.FOCUS_IN	구성 요소가 focus 를 받았습니다.
FocusHandlerEvent.FOCUS_OUT	구성 요소가 focus 를 잃었습니다.
Event.SELECT	선택된 프로퍼티가 변경되었습니다.
MouseEvent.ROLL_OVER	마우스 커서가 버튼 위에 놓였습니다.

	<p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx로 보내야 합니다.</p> <p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx로 보내야 합니다.</p>
MouseEvent.ROLL_OUT	<p>마우스 커서가 버튼 밖으로 벗어났습니다.</p> <p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx로 보내야 합니다.</p> <p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx로 보내야 합니다.</p>
ButtonEvent.PRESS	<p>버튼이 눌렸습니다.</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p> <p><i>isKeyboard</i>: 이벤트가 키보드/게임패드에 의해 생성되었으면 true이고 이벤트가 마우스에 의해 생성되었으면 false입니다.</p> <p><i>isRepeat</i>: 이벤트가 autoRepeating 버튼을 길게 눌러 생성되었으면 true이고 버튼이 현재 반복 중이 아니면 false입니다.</p>
MouseEvent.DOUBLE_CLICK	<p>버튼이 두 번 클릭되었습니다. <i>doubleClickEnabled</i> 프로퍼티가 true인 경우에만 일어납니다.</p> <p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능).</p>

	<p>uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx로 보내야 합니다.</p> <p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx로 보내야 합니다.</p>
ButtonEvent.CLICK	<p>버튼이 클릭되었습니다.</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p> <p><i>isKeyboard</i>: 이벤트가 키보드/게임패드에 의해 생성되었으면 true이고 이벤트가 마우스에 의해 생성되었으면 false입니다.</p> <p><i>isRepeat</i>: 이벤트가 autoRepeating 버튼을 길게 눌러 생성되었으면 true이고 버튼이 현재 반복 중이 아니면 false입니다.</p>
ButtonEvent.DRAG_OVER	<p>마우스 커서가 버튼위로 드래그 되었다(왼쪽 버튼이 눌린 채로).</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p>
ButtonEvent.DRAG_OUT	<p>마우스 커서가 버튼 바깥으로 드래그 되었다(왼쪽 버튼이 눌린 채로).</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p>
ButtonEvent.RELEASE_OUTSIDE	<p>마우스 커서가 버튼 바깥으로 드래그 되었고, 그 상태로 마우스 왼쪽 버튼이 떼어졌다.</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능).</p>

uint 형식입니다.

이들 이벤트를 처리하려면 액션스크립트 코드가 필요하다. 다음 예는 버튼 클릭을 다루는 방법을 보여준다.

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

첫 번째 줄에서 이벤트 리스너를 click 이벤트에 대하여 설정하는데, 이때 버튼명은 myButton이고, 이벤트가 발생할 경우 onButtonPress 함수를 호출하도록 하고 있다. 다른 이벤트도 동일한 패턴으로 처리할 수 있을 것이다. 이벤트 핸들러(예제의 'e')에 제공되는 ButtonEvent 파라미터에는 이벤트에 대한 관련 정보가 포함됩니다. 파라미터의 이벤트 형식은 동일한 클래스이거나 리스너에 추가할 때 사용된 형식의 원형이어야 합니다.

2.1.2 CheckBox



그림 10: 스키너 없는 CheckBox

CheckBox (scaleform.clik.controls.CheckBox)는 Button 컴포넌트로서 클릭되었을 때 선택된 상태를 토글하도록 되어 있다. CheckBox는 true/false(부울값)의 변화를 나타낸다. 기능적으로는 ToggleButton과 동일하지만, 내부적으로 토글 속성을 처리한다.

2.1.2.1 사용자 상호작용

마우스나 동등한 키보드 컨트롤을 통해서 CheckBox 컴포넌트를 클릭하면 연속적으로 선택과 해제가 바뀐다. 다른 모든 경우의 경우는 Button과 동일하게 작동한다.

2.1.2.2 컴포넌트 설정

CLIK CheckBox 클래스를 사용하는 MovieClip 은 반드시 다음과 같이 명명된 부요소를 가져야 한다.

- **textField**: (optional) TextField 타입. 버튼 레이블
- **focusIndicator**: (optional) MovieClip 형. 분리된 MovieClip 이 포커스된 상태 출력에 사용된다.
실제 존재하는 경우에는 MovieClip 이 show 와 hide 라 명명된 프레임을 갖고 있어야 한다.

2.1.2.3 상태

토글 속성 때문에 CheckBox 는 selected 를 나타내는 추가적인 키프레임이 필요하다.

- **up** 혹은 기본 상태
- **over** 상태는 마우스 커서가 위에 있을 때, 혹은 포커스를 가졌을 때
- **down** 상태는 버튼이 눌렸을 때
- **disabled** 상태
- **selected_up** 혹은 기본상태
- **selected_over** 상태는 마우스 커서가 위에 있을 때, 혹은 포커스를 가졌을 때.
- **selected_down** 상태는 버튼이 눌렸을 때
- **selected_disabled** 상태

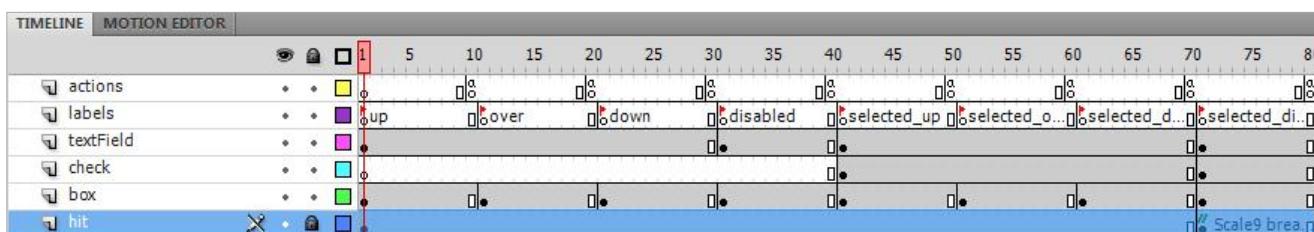
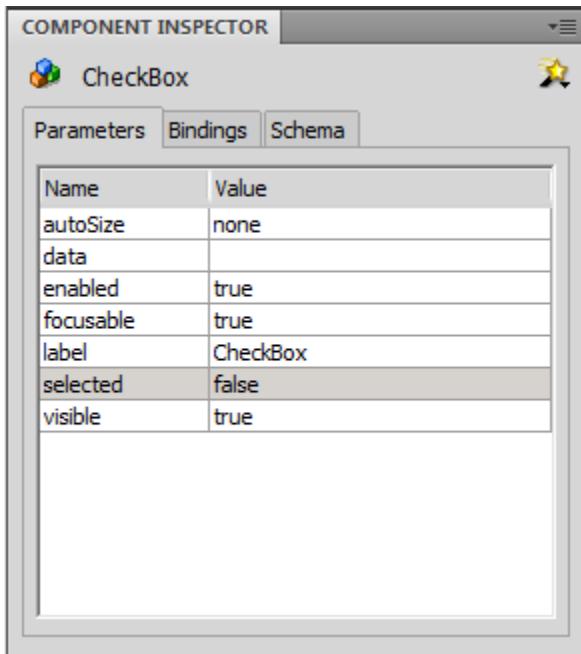


Figure 11: CheckBox timeline.

이들은 CheckBox 에서 필요한 최소한의 키프레임이다. Button 컴포넌트(결과적으로 CheckBox 컴포넌트도 포함)에서 지원되는 확장 상태 집합과 키프레임은 [Getting Started with CLIK Buttons](#) 문서를 참고하라.

2.1.2.4 점검가능 속성



Button 컨트롤에서 상속받기 때문에 CheckBox는 Button과 동일한 점검가능 속성을 가지고 있다. 단, toggle과 autoRepeat 속성은 제외된다.

autoSize	포함된 텍스트에 맞춰서 버튼 크기가 조절되고, 크기가 조절된 버튼을 정렬할 방향을 결정합니다. autoSize 프로퍼티를 autoSize=TextFieldAutoSize.NONE으로 설정하면 크기를 변경하지 않고 그대로 놔둡니다.
Data	버튼 관련 데이터입니다. 이 프로퍼티는 버튼을 ButtonGroup에서 사용할 때 특히 도움이 됩니다.
Enabled	false로 설정하면 버튼을 비활성화합니다. 비활성화한 구성 요소는 더 이상 사용자 입력을 받지 않습니다.
focusable	구성 요소에 대한 focus 관리를 활성화/비활성화합니다. focus 가능한 프로퍼티를 false로 설정하면 tab 키, 방향 키 및 마우스 기반 focus 변경에 대한 지원을 제거합니다.
Label	레이블 설정
selected	버튼의 선택된 상태를 설정합니다. 버튼은 선택과 비선택의 두 가지

	마우스 상태 세트를 가질 수 있습니다. 버튼의 전환 프로퍼티가 true 일 때 선택된 상태는 버튼을 클릭하면 변경되지만, 선택된 상태는 전환 프로퍼티가 false 일 경우에도 ActionScript 를 사용해서 설정될 수 있습니다.
Visible	false 면 버튼을 숨긴다

2.1.2.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- ***type***: 이벤트의 형식입니다.
- ***target***: 이벤트 대상입니다.
- ***currentTarget***: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- ***eventPhase***: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- ***bubbles***: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- ***cancelable***:
- 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

CheckBox 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이기 속성(visible property)이 실행시 true 로 설정되었다.
ComponentEvent.HIDE	보이기 속성(visible property)이 실행시 false 로 설정되었다.
FocusHandlerEvent.FOCUS_IN	버튼이 focus 를 받았습니다.
FocusHandlerEvent.FOCUS_OUT	버튼이 focus 를 잃었습니다.
Event.SELECT	선택된 프로퍼티가 변경되었습니다.

ComponentEvent.STATE_CHANGE	상태가 변경되었다.
MouseEvent.ROLL_OVER	<p>마우스 커서가 버튼 위로 왔다.</p> <p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를</p>
	<p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.</p>
MouseEvent.ROLL_OUT	<p>마우스 커서가 버튼 위를 벗어났다.</p> <p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를</p>
	<p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.</p>
ButtonEvent.PRESS	<p>버튼이 눌렸다.</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p> <p><i>isKeyboard</i>: 이벤트가 키보드/게임패드에 의해 생성되었으면 true이고 이벤트가 마우스에 의해 생성되었으면 false입니다.</p> <p><i>isRepeat</i>: 이벤트가 autoRepeating 버튼을 길게 눌러 생성되었으면 true이고 버튼이 현재 반복 중이 아니면 false입니다.</p>

MouseEvent.DOUBLE_CLICK	<p>버튼이 더블클릭 되었다. <i>doubleClickEnabled</i> 속성이 true 일 경우만 발생한다.</p> <p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를</p> <p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.</p>
ButtonEvent.CLICK	<p>버튼이 클릭 되었다.</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p> <p><i>isKeyboard</i>: 이벤트가 키보드/게임패드에 의해 생성되었으면 true이고 이벤트가 마우스에 의해 생성되었으면 false입니다.</p> <p><i>isRepeat</i>: 이벤트가 autoRepeating 버튼을 길게 눌러 생성되었으면 true이고 버튼이 현재 반복 중이 아니면 false입니다.</p>
ButtonEvent.DRAG_OVER	<p>마우스 커서가 버튼위로 드래그 되었다(왼쪽 버튼이 눌린채로).</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p>
ButtonEvent.DRAG_OUT	<p>마우스 커서가 버튼 바깥으로 드래그 되었다(왼쪽 버튼이 눌린채로).</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p>
ButtonEvent.RELEASE_OUTSIDE	<p>마우스 커서가 버튼 바깥으로 드래그 되었고, 그</p>

상태로 마우스 왼쪽 버튼이 떼어졌다. 숫자 타입.
0에서 3의 값
controllerIdx: 이벤트를 생성하는 데 사용되는
컨트롤러의 인덱스입니다(멀티 마우스 커서 환경
용으로만 적용 가능). uint 형식입니다.

다음 예제코드는 CheckBox 토글을 어떻게 다루는지를 보여준다.

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

2.1.3 Label



Figure 12: Unskinned Label.

CLIK Label 컴포넌트(scaleform.clik.controls.Label)는 간단히 말해서 MovieClip 심볼에 의해 랩핑(wrapping)되고 몇 가지 편리한 기능이 추가된 편집이 불가능한 표준 textField이다. 내부적으로 Label은 표준 textField처럼 동일한 속성 및 행동을 지원하지만 많이 사용하는 기능의 소량만 컴포넌트 자체에 의해 노출된다. Label의 실질적 textField에 대한 액세스는 사용자가 직접 속성을 변경하고자 할 때 가능하다. 아래에 설명된 특정한 예의 경우, 개발자는 Label 컴포넌트 대신 표준 textField를 사용할 수도 있다.

Label은 MovieClip 심볼이므로 textField와는 달리 그래픽 요소로 화려하게 치장할 수 있다. 심볼이기 때문에 textField 인스턴스와 같이 인스턴스마다 설정을 할 필요가 없다. 또한, Label은 동일한 결과를 얻기 위해 복잡한 AS2 코딩이 필요한 표준 textField와는 달리 타임라인에서 정의된 disabled 상태를 제공한다.

Label 컴포넌트는 기본적으로 제약(constraints)을 사용하는데 이는 스테이지에서 Label 인스턴스를 리사이징(resizing)하는 것은 런타임 때 아무런 시각적 효과가 나타나지 않는다는 뜻이다. 만약 textField의 리사이징이 필요한 경우 Label 보다는 표준 textField를 사용해야 할 것이다. 일반적으로 만약 텍스트 요소를 지속적인 사용할 필요가 없다면 Label 컴포넌트보다는 표준 textField를 사용하는 것이 더 가벼운 방법일 수 있다.

2.1.3.1 사용자 상호작용

Label에는 사용자 상호작용이 불가능하다.

2.1.3.2 컴포넌트 설정

CLIK Label 클래스를 사용하는 MovieClip은 반드시 다음과 같이 명명된 부요소를 가져야 한다.

- **textField**: TextField 타입. 레이블 텍스트

2.1.3.3 상태

CLIK Label 컴포넌트는 disabled 속성에 기반해서 2 가지 상태를 지원한다.

- **default** 혹은 사용가능 상태
- **disabled** 상태

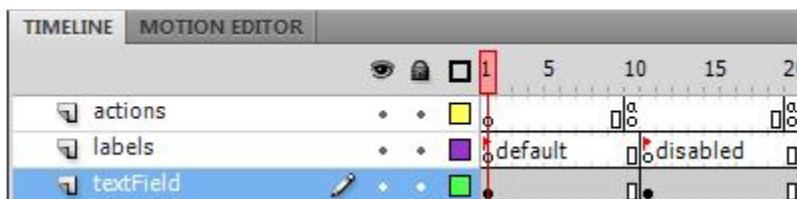
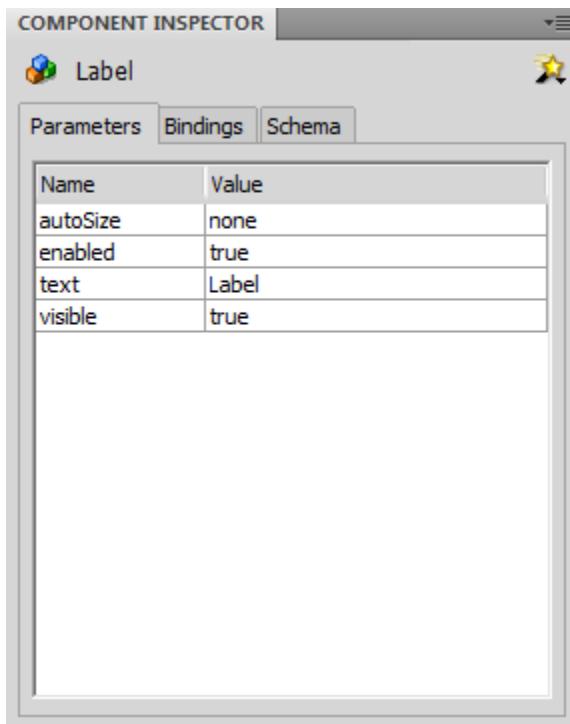


Figure 13: Label timeline.

2.1.3.4 점검가능 속성



Label 컴포넌트 점검가능속성은 다음과 같다.

Text	레이블의 텍스트 설정
Visible	false 로 설정하면 구성 요소를 숨깁니다.
enabled	false 로 설정하면 구성 요소를 비활성화합니다.
autoSize	포함된 텍스트에 맞춰서 라벨 크기가 조절되고, 크기가 조절된 버튼을 정렬할 방향을 결정합니다. autoSize 프로퍼티를 autoSize=TextFieldAutoSize.NONE 으로 설정하면 크기를 변경하지 않고 그대로 놔둡니다.

2.1.3.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- **type:** 이벤트의 형식입니다.

- ***target***: 이벤트 대상입니다.
- ***currentTarget***: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- ***eventPhase***: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- ***bubbles***: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- ***cancelable***:
- 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

Label 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true 로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false 로 설정되었습니다.
ComponentEvent.STATE_CHANGE	구성 요소의 상태가 변경되었습니다.

2.1.4 TextInput

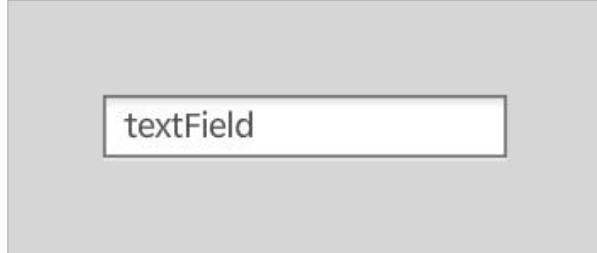


Figure 14: Unskinned TextInput.

TextInput (scaleform.clik.controls.TextInput)은 편집 가능한 textField 컴포넌트로서 텍스트 기반의 사용자 입력을 캡처한다. 본 컴포넌트는 Label 처럼 표준 textField 의 래퍼(wrapper)에 불과하므로 암호모드, 최고 문자수 및 HTML 텍스트 등과 같은 textField 와 동일한 기능을 지원한다. TextInput 의 textField 인스턴스에 직접 액세스하여 변경할 수 있는 다른 속성들에 비해 컴포넌트 자체에 의해 노출되는 이러한 속성은 소량에 불과하다.

TextInput 컴포넌트는 Label 을 사용하여 에디팅이 불가능한 텍스트가 출력이 될 수 있으므로 input 에서 사용하여야 한다. Label 처럼 개발자는 필요에 따라 TextInput 컴포넌트를 표준 textField 로

대체할 수 있다. 하지만, 복잡한 UI를 개발할 때, 특히 PC 어플리케이션 같은 경우, TextInput 컴포넌트는 표준 textField에 비해 더 값어치있는 기능을 제공한다.

TextInput은 표준 textField에서는 쉽게 얻을 수 없는 포커스와 disable 상태를 지원한다. 분리된 포커스 상태 때문에 TextInput은 표준 textField에는 포함되어 있지 않은 커스텀 포커스 표시기(indicator)를 지원한다. 표준 textField는 시각적 스타일을 변경하려면 복잡한 AS2 코딩이 필요하지만 TextInput의 시각적 스타일은 타임라인에서 간단하게 설정할 수 있다. TextInput의 검사 가능한 속성은 Flash Studio에 익숙하지 못한 디자이너와 프로그래머에게 쉬운 업무의 흐름을 제공한다. 개발자는 또한 커스텀 작동을 생성하기 위해 TextInput에 의해 발생한 이벤트를 쉽게 확인할 수 있다.

TextInput은 표준 선택과 HTML 포맷 텍스트의 다단락(multi paragraph)을 포함하여 textField에서 제공하는 잘라내기, 복사하기, 붙여넣기 등의 기능을 지원한다. 기본적으로 키보드 명령문으로 선택(Shift+방향키), 잘라내기(Shift+Delete), 복사하기(Shift+Copy) 그리고 붙여넣기(Shift+Insert) 등이 있다.

TextInput은 마우스 커서에 의한 roll over와 roll out 이벤트에 대한 하이라이트를 지원한다. 특별한 actAsButton 속성으로 두 개의 마우스 이벤트들로 작동하는 두 개의 추가적인 키프레임들을 제공하여 이 기능이 가능하도록 설정 할 수 있다. 이 프레임들은 "over"와 "out"으로 명명되었으며, 각각 rollOver와 rollOut 상태를 의미한다. 만약 actAsButton이 설정되었는데 "over"와 "out"프레임이 존재하지 않는다면 TextInput은 두 이벤트에 대응하여 "default"라는 키프레임으로 재생될 것이다. 이러한 프레임은 CLICK과 함께 제공된 프리빌트 TextInput 컴퍼넌트를 위해서 디폴트로 존재하지 않음에 주의한다. 그것들은 개발자들이 특별한 필요에 따라 추가하도록 의도된 것이다.

이 컴퍼넌트는 또한 아무런 값이 입력되지 않았거나 사용자가 입력했을 때의 default Text를 지원한다. defaultText 속성은 어떤 문자열도 입력할 수 있도록 해준다. Default Text의 테마(색깔과 모양)은 연한 회색(0xAAAAAA)을 띠고 이탤릭이 적용될 것이다. 사용자 지정 스타일은 새로운 TextFormat 객체를 defaultTextFormat 속성으로 할당하여 설정할 수 있다.

2.1.4.1 사용자 상호작용

TextInput을 클릭하면 포커스를 주게 되고 결과적으로 textField에 커서가 나타나게 된다. 이 커서가 나타나면 사용자는 키보드나 동등한 컨트롤러를 통해서 문자를 타이핑할 수 있다. 좌우 화살표키를

누르면 커서를 적절한 방향으로 이동시킨다. 왼쪽 화살표키가 눌렸을 때 이미 textField 좌측 끝에 도달해 있는 상태라면 좌측에 있는 컨트롤러 포커스가 이동할 것이다. 이는 오른쪽 화살표도 마찬가지다.

2.1.4.2 컴포넌트 설정

CLIK TextInput 클래스를 사용하는 MovieClip 은 다음과 같은 부요소를 갖고 있어야만 한다.

- **textField**: textField 타입.

2.1.4.3 상태

CLIK TextInput 컴포넌트는 포커스와 disabled 속성에 기반한 3 가지 상태를 지원한다.

- **default** 혹은 enabled 상태
- **focused** 상태, 일반적으로 textField 의 경계면에 하이라이트를 출력해서 상태를 나타낸다
- **disabled** 상태

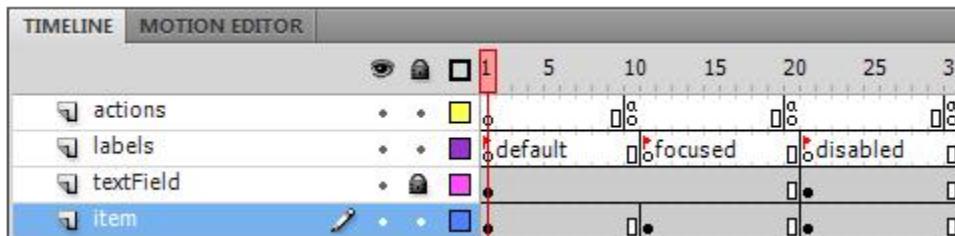
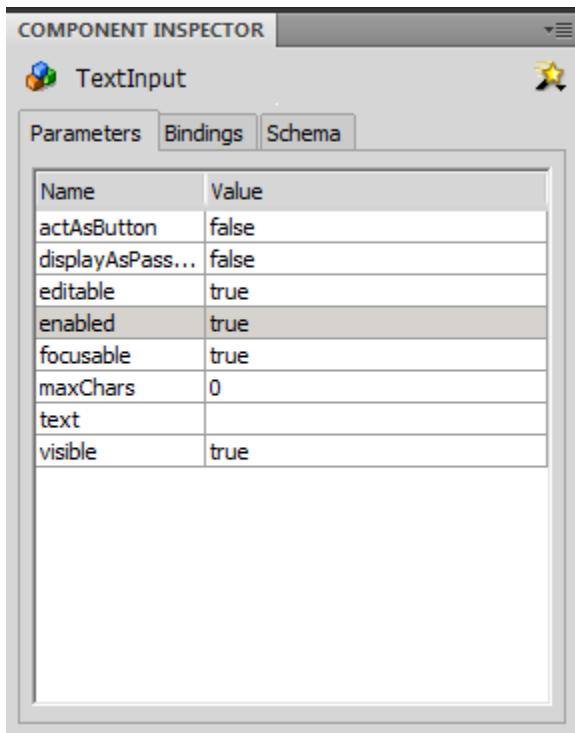


Figure 15: TextInput timeline.

2.1.4.4 Inspectable properties



TextInput 컴포넌트의 점검 가능 속성은 다음과 같다.

actAsButton

만약 true 이면, TextInput 은 포커스를 갖지 않을 때의 rollover 와 rollOut 상태를 지원하는 버튼과 비슷하게 동작할 것임. 마우스나 탭이 눌리는 것을 통해 한번 포커스를 갖게 되면, 그 TextInput 은 포커스를 잃을 때까지 정상 모드로 복귀함.

displayAsPassword

true 인 경우 실제 문자 대신 '*' 문자를 표시하도록 textField 를 설정합니다. textField 의 값은 사용자에 의해 입력되고 텍스트 프로퍼티에 의해 반환된 실제 문자가 됩니다.

Editable

false 로 설정하면 TextInput 을 편집할 수 없게 됩니다.

enabled

false 로 설정하면 구성 요소를 비활성화합니다.

focusable

구성 요소에 대한 focus 관리를 활성화/비활성화합니다.

focus 가능한 프로퍼티를 false 로 설정하면 tab 키, 방향

	키 및 마우스 기반 focus 변경에 대한 지원을 제거합니다.
maxChars	textField 에 입력될 수 있는 문자의 수를 0 보다 큰 숫자로 제한합니다.
Text	textField 의 초기 텍스트를 설정합니다.
Visible	false 로 설정하면 구성 요소를 숨깁니다.

2.1.4.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- ***type***: 이벤트의 형식입니다.
- ***target***: 이벤트 대상입니다.
- ***currentTarget***: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- ***eventPhase***: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- ***bubbles***: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- ***cancelable***: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

TextInput 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true 로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false 로 설정되었습니다.
FocusHandlerEvent.FOCUS_IN	구성 요소가 focus 를 받았습니다.
FocusHandlerEvent.FOCUS_OUT	구성 요소가 focus 를 잃었습니다.
Event.CHANGE	textField 의 내용이 변경되었다.
ComponentEvent.STATE_CHANGE	구성 요소의 상태가 변경되었습니다.
MouseEvent.ROLL_OVER	마우스 커서가 버튼 위로 왔다. <i>mouseIdx</i> : 이벤트를 생성하는 데 사용되는 마우스 커서의

인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.

buttonIdx: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.

MouseEvent.ROLL_OUT

마우스 커서가 버튼 위를 벗어났다

mouseIdx: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.

buttonIdx: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.

다음 코드는 textField 내용의 변경을 대기한다.

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

2.1.5 TextArea

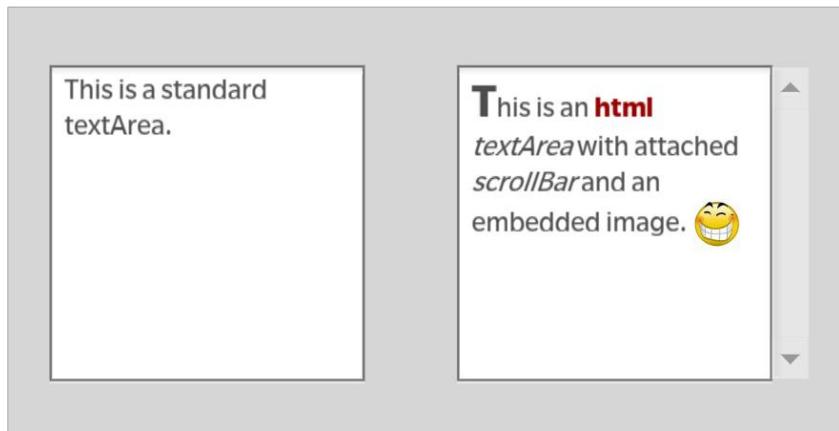


Figure 16: Unskinned TextArea.

TextArea (scaleform.clik.controls.TextArea)는 CLIK TextInput에서 파생되었기 때문에 동일한 기능과 속성 그리고 상태를 공유하지만 여러 줄의 에디팅 및 스크롤링이 가능한 입력을 위한 스크롤바(ScrollBar)를 추가 지원한다. TextInput과 TextArea 사이의 공유되는 특수 기능을 자세히 알아보려면 TextInput 설명을 참조하라.

Label 및 TextInput 처럼 TextArea 역시 표준 multiline textField의 래퍼(wrapper)이므로 HTML 텍스트, 단어 래핑(wrapping), 선택, 잘라내기, 복사하기, 붙여넣기 등의 표준 textField의 속성과 기능을 지원한다. 개발자는 TextArea 대신 표준 textField로 대체 사용할 수 있으나 광범위한 기능, 상태, 검사 가능한 속성 및 이벤트를 사용하려면 TextArea 컴포넌트를 사용할 것을 권장한다.

2.1.5.1 사용자 상호작용

TextInput을 클릭하면 포커스를 주게 되고 결과적으로 textField에 커서가 나타나게 된다. 이 커서가 나타나면 사용자는 키보드나 동등한 컨트롤러를 통해서 문자를 타이핑할 수 있다. 좌우 화살표키를 누르면 커서를 적절한 방향으로 이동시킨다. 화살표 키가 눌렸을 때 커서가 이미 경계면에 도달해 있는 상태라면 화살표 키 방향으로 인접한 컨트롤로 포커스가 이동된다.

2.1.5.2 컴포넌트 설정

CLIK TextArea 클래스를 사용하는 MovieClip은 다음과 같은 부요소를 갖고 있어야만 한다.

- **textField**: TextField type.

2.1.5.3 상태

부모인 TextInput 과 비슷하게, TextArea 컴포넌트는 포커스와 disabled 속성에 기반한 3 가지 상태를 지원한다.

- **default** 혹은 enabled 상태
- **focused** 상태, 일반적으로 textField 의 경계면에 하이라이트를 출력해서 상태를 나타낸다
- **disabled** 상태

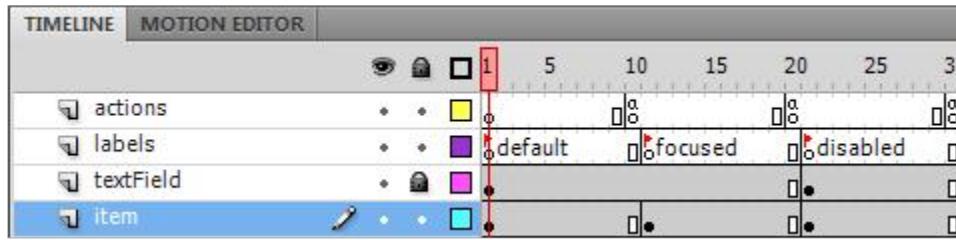
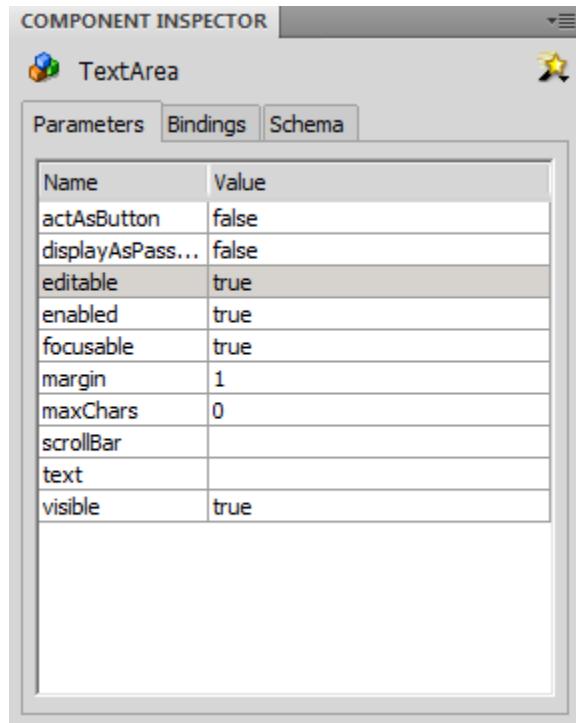


Figure 17: TextArea timeline.

2.1.5.4 Inspectable properties



TextArea 컴포넌트의 점검가능 속성은 TextInput 과 비교했을 때 password 속성이 삭제되었다는 것이 다르며, 또한 CLIK ScrollBar 컴포넌트와 관련된 속성이 추가 되어 있다. 이에 대해서는 2.4 장에서 설명할 것이다.

actAsButton	true 인 경우 focus 가 아닐 때 TextInput 이 버튼과 유사하게 동작하고 rollOver 및 rollOut 상태를 지원합니다. 마우스를 누르거나 탭을 통해 focus 되면 TextInput 은 focus 를 잃을 때까지 일반 모드로 되돌립니다.
displayAsPassword	true 인 경우 실제 문자 대신 '*' 문자를 표시하도록 textField 를 설정합니다. textField 의 값은 사용자에 의해 입력되고 텍스트 프로퍼티에 의해 반환된 실제 문자가 됩니다.
editable	false 라면 TextInput 이 편집 불가능하다.
enabled	false 로 설정하면 구성 요소를 비활성화합니다.
focusable	구성 요소에 대한 focus 관리를 활성화/비활성화합니다. focus 가능한 프로퍼티를 false 로 설정하면 tab 키, 방향 키 및 마우스 기반 focus 변경에 대한 지원을 제거합니다.
maxChars	textField 에 입력 가능한 문자수(0 보다 큰 숫자)
scrollBar	사용하려는 CLIK ScrollBar 컴포넌트의 인스턴스 명이나 ScrollBar 심볼의 링크 ID(이 경우에는 TextArea 에 의해서 인스턴스가 생성될 것이다)
scrollPolicy	auto 로 설정하면 텍스트를 스크롤하기 충분한 경우에만 스크롤 바가 나타난다. on 이면 스크롤바가 항상 나타나며, off 의 경우에는 절대로 나타나지 않는다. 이 속성은 ScrollBar 가 적용된 경우에만 영향을 미친다(ScrollBar 속성 참조).
Text	textField 의 초기 텍스트를 설정합니다.
Visible	false 로 설정하면 구성 요소를 숨깁니다.

2.1.5.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- ***type***: 이벤트의 형식입니다.
- ***target***: 이벤트 대상입니다.
- ***currentTarget***: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- ***eventPhase***: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- ***bubbles***: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- ***cancelable***:
- 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

TextArea 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true 로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false 로 설정되었습니다.
FocusHandlerEvent.FOCUS_IN	컴포넌트가 포커스를 받았다.
FocusHandlerEvent.FOCUS_OUT	컴포넌트가 포커스를 잃었다.
Event.CHANGE	textField 의 내용이 변경되었다
Event.SCROLL	텍스트 영역이 스크롤 되었다.
ComponentEvent.STATE_CHANGE	구성 요소의 상태가 변경되었습니다.
MouseEvent.ROLL_OVER	컴퍼넌트에 포커스가 없을 때 컴퍼넌트와 마우스 커서가 겹쳤음. <i>mouseIdx</i> : 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.
	<i>buttonIdx</i> : 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.
MouseEvent.ROLL_OUT	컴퍼넌트에 포커스가 없을 때 마우스 커서가 컴퍼넌트 밖으로 나갔음.

mouseIdx. 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.

buttonIdx. 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.

다음 코드는 TextArea 스크롤 이벤트를 대기한다.

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

2.2 그룹 타입(*Group Types*)

그룹 타입은 RadioButton, ButtonGroup, ButtonBar 컴포넌트로 구성되어 있다. ButtonGroup 은 Button 컴포넌트 그룹들을 관리하기 위한 특별한 로직을 가질 수 있는 관리자 타입니다. 이 컴포넌트는 비쥬얼 한 외관도 없을뿐더러 스테이지 상에 존재하지도 않는다. 하지만, ButtonBar 는 스테이지에 존재하면서 Button 그룹들을 관리할 수 있다. RadioButton 은 자동적으로 ButtonGroup 에 형제 버튼들을 그룹화 하는 특별한 Button 이다.

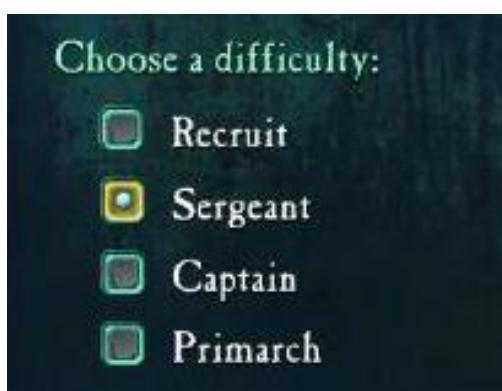


그림 18: *Dawn of War II*의 Radio ButtonGroup 예제

2.2.1 RadioButton



그림 19: 스키없는 RadioButton

RadioButton (scaleform.clik.controls.RadioButton)은 버튼 컴포넌트로서 일반적으로 단일 값을 변화시키거나 출력할 때 사용된다. RadioButton 은 집합 내에서 하나만 선택가능하며 다른 RadioButton 을 선택하면 기존에 선택된 컴포넌트는 선택이 해제된다.

CLIK RadioButton 은 CheckBox 컴포넌트와 매우 비슷하며 기능, 상태, 작동 등을 공유한다. 가장 큰 차이점은 RadioButton 이 그룹 속성을 지원한다는 것으로, 커스텀 ButtonGroup(다음 장에서 설명한다)을 적용할 수 있다. RadioButton 은 또한 토글속성을 기본적으로 지원하지 않는다는 것이다. 토글링은 이를 관리하는 ButtonGroup 에 의해서 수행되기 때문이다.

2.2.1.1 사용자 상호작용

마우스나 동등한 컨트롤을 통해서 RadioButton 컴포넌트를 클릭하면 선택되지 않았을 경우에 지속적으로 선택이 된다. 만약 RadioButton 이 선택되었고, 좀전에 클릭된 다른 RadioButton 과 동일한 ButtonGroup 에 소속되어 있다면, 이전에 선택 된 RadioButton 의 선택이 해제된다. 다른 모든 경우에 대해서는 RadioButton 는 Button 과 동일하게 작동한다.

2.2.1.2 컴포넌트 설정

CLIK RadioButton 클래스를 사용하는 MovieClip 은 반드시 다음과 같이 명명된 부요소를 가져야 한다.

- **textField:** (optional) TextField 타입. 버튼 레이블

- ***focusIndicator***. (optional) MovieClip 형. 분리된 MovieClip 이 포커스 된 상태 출력에 사용된다. 실제 존재하는 경우에는 MovieClip 이 show 와 hide 라 명명된 프레임을 갖고 있어야 한다.

2.2.1.3 상태

CheckBox 와 비슷하게 RadioButton 은 선택과 해제의 토글이 가능하기 때문에 적어도 다음과 같은 상태가 필요하다.

- ***up*** 혹은 기본 상태
- ***over*** 상태는 마우스 커서가 위에 있을 때, 혹은 포커스를 가졌을 때
- ***down*** 상태는 버튼이 눌렸을 때
- ***disabled*** 상태
- ***selected_up*** 혹은 기본상태
- ***selected_over*** 상태는 마우스 커서가 위에 있을 때, 혹은 포커스를 가졌을 때.
- ***selected_down*** 상태는 버튼이 눌렸을 때
- ***selected_disabled*** 상태

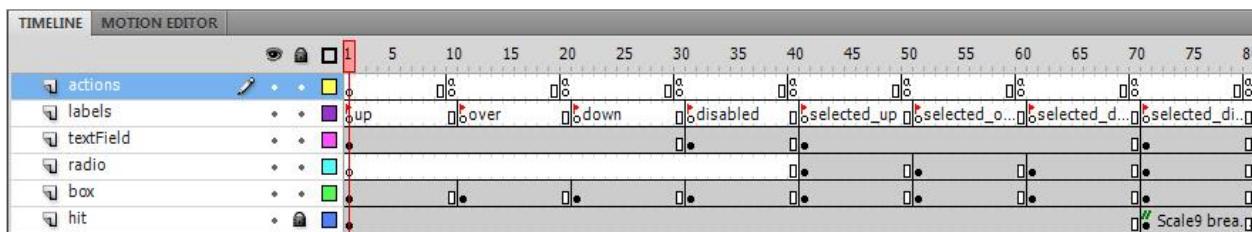
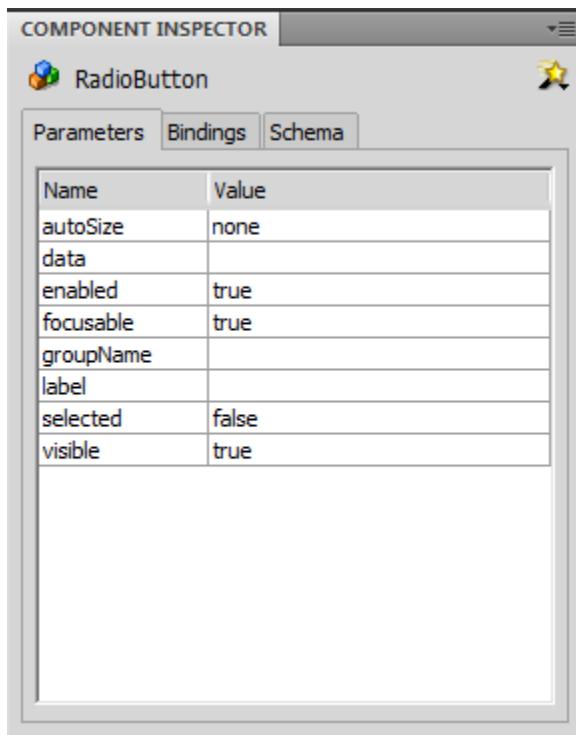


그림 20: RadioButton 타임라인.

이들은 RadioButton 에서 필요한 최소한의 키프레임이다. Button 컴포넌트(결과적으로 RadioButton 컴포넌트도 포함)에서 지원되는 확장 상태 집합과 키프레임은 [Getting Started with CLIK Buttons](#) 문서를 참고하라.

2.2.1.4 점검가능 속성



RadioButton 는 Button 에서 파생되었기 때문에 toggle 과 disableFocus 속성을 제외하면 동일한 속성을 갖는다.

autoSize	포함된 텍스트에 맞춰서 버튼 크기가 조절되고, 크기가 조절된 버튼을 정렬할 방향을 결정합니다. autoSize 프로퍼티를 autoSize=TextFieldAutoSize.NONE 으로 설정하면 크기를 변경하지 않고 그대로 놔둡니다.
Data	버튼 관련 데이터입니다. 이 프로퍼티는 버튼을 ButtonGroup 에서 사용할 때 특히 도움이 됩니다.
enabled	false 로 설정하면 버튼을 비활성화합니다. 비활성화한 구성 요소는 더 이상 사용자 입력을 받지 않습니다.
focusable	구성 요소에 대한 focus 관리를 활성화/비활성화합니다. focus 가능한 프로퍼티를 false 로 설정하면 tab 키, 방향 키 및 마우스 기반 focus 변경에 대한 지원을 제거합니다.

groupName	존재하거나 RadioButton 에 의해 자동으로 생성되어야 하는 ButtonGroup 인스턴스의 이름입니다. RadioButton 에 의해 생성되는 경우 새 ButtonGroup 은 RadioButton 의 컨테이너 안에 존재하게 됩니다. 예를 들어 RadioButton 이 _root 에 존재하면 ButtonGroup 오브젝트는 _root 에 생성됩니다. 같은 그룹에서 사용하는 모든 RadioButton 은 한 세트에 속하게 됩니다.
Label	버튼의 라벨을 설정합니다.
selected	버튼의 선택된 상태를 설정합니다. 버튼은 선택과 비선택의 두 가지 마우스 상태 세트를 가질 수 있습니다. 버튼의 전환 프로퍼티가 true 일 때 선택된 상태는 버튼을 클릭하면 변경되지만, 선택된 상태는 전환 프로퍼티가 false 일 경우에도 ActionScript 를 사용해서 설정될 수 있습니다.

2.2.1.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- **type**: 이벤트의 형식입니다.
- **target**: 이벤트 대상입니다.
- **currentTarget**: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- **eventPhase**: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- **bubbles**: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- **cancelable**: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

RadioButton 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true 로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false 로 설정되었습니다.

FocusHandlerEvent.FOCUS_IN	구성 요소가 focus 를 받았습니다.
FocusHandlerEvent.FOCUS_OUT	구성 요소가 focus 를 잃었습니다.
Event.SELECT	선택된 프로퍼티가 변경되었습니다.
ComponentEvent.STATE_CHANGE	구성 요소의 상태가 변경되었습니다.
MouseEvent.ROLL_OVER	<p>마우스 커서가 버튼 위에 놓였습니다.</p> <p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.</p> <p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.</p>
MouseEvent.ROLL_OUT	<p>마우스 커서가 버튼 밖으로 벗어났습니다.</p> <p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.</p> <p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.</p>
ButtonEvent.PRESS	<p>버튼이 눌렸습니다.</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p> <p><i>isKeyboard</i>: 이벤트가 키보드/게임패드에 의해 생성되었으면 true이고 이벤트가 마우스에 의해 생성되었으면 false입니다.</p> <p><i>isRepeat</i>: 이벤트가 autoRepeating 버튼을 길게 눌러</p>

	<p>생성되었으면 true이고 버튼이 현재 반복 중이 아니면 false입니다.</p>
MouseEvent.DOUBLE_CLICK	<p>버튼이 두 번 클릭되었습니다. <i>doubleClickEnabled</i> 프로퍼티가 true인 경우에만 일어납니다.</p> <p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx로 보내야 합니다.</p> <p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx로 보내야 합니다.</p>
ButtonEvent.CLICK	<p>버튼이 클릭되었습니다.</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p> <p><i>isKeyboard</i>: 이벤트가 키보드/게임패드에 의해 생성되었으면 true이고 이벤트가 마우스에 의해 생성되었으면 false입니다.</p> <p><i>isRepeat</i>: 이벤트가 autoRepeating 버튼을 길게 눌러 생성되었으면 true이고 버튼이 현재 반복 중이 아니면 false입니다.</p>
ButtonEvent.DRAG_OVER	<p>마우스 커서가 버튼위로 드래그 되었다(왼쪽 버튼이 눌린 채로).</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p>
ButtonEvent.DRAG_OUT	<p>마우스 커서가 버튼 바깥으로 드래그 되었다(왼쪽 버튼이 눌린 채로).</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의</p>

인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.

ButtonEvent.RELEASE_OUTSIDE

마우스 커서가 버튼 바깥으로 드래그 되었고, 그 상태로 마우스 왼쪽 버튼이 떼어졌다.

controllerIdx: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. 0에서 3의 값

다음 예제코드는 RadioButton 토글을 어떻게 다루는지를 보여준다.

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

2.2.2 ButtonGroup

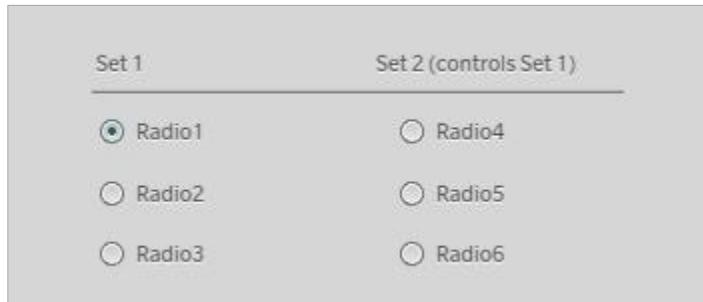


그림 21: 스킨없는 ButtonGroup.

CLIK ButtonGroup (scaleform.clik.controls.ButtonGroup)은 버튼 집합을 관리하기 때문에 매우 중요하다. 이 컴포넌트는 집합내에 있는 하나만 선택가능하고 나머지는 선택이 되지 않도록 해준다. 만약 사용자가 다른 버튼을 클릭하면 현재 선택되어 있던 버튼은 선택이 해제된다. CLIK Button 컴포넌트로부터 상속된 모든 (CheckBox 나 RadioButton 같은) 버튼은 ButtonGroup에 적용될 수 있다.

2.2.2.1 사용자 상호작용

ButtonGroup은 아무런 사용자 상호작용이 없다. 이는 ButtonGroup이 화면상에 보이지 않기 때문이다. 하지만 소속된 RadioButtons이 클릭되면 간접적으로 영향을 받는다.

2.2.2.2 컴포넌트 설정

CLIK ButtonGroup 클래스를 사용하는 MovieClip은 아무런 부요소를 필요로 하지 않는다. 이는 ButtonGroup이 화면상에 보이지 않기 때문이다.

2.2.2.3 상태

ButtonGroup은 스테이지 상에서 아무런 비쥬얼 형태를 갖지 않기 때문에 관련된 상태도 존재하지 않는다.

2.2.2.4 점검가능 속성

ButtonGroup은 스테이지 상에서 아무런 비쥬얼 형태를 갖지 않기 때문에 점검가능 속성도 존재하지 않는다.

2.2.2.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- ***type***: 이벤트의 형식입니다.
- ***target***: 이벤트 대상입니다.
- ***currentTarget***: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- ***eventPhase***: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- ***bubbles***: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- ***cancelable***: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

ButtonGroup 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

Event.CHANGE	그룹의 새로운 버튼이 선택되었다.
ButtonEvent.CLICK	그룹의 버튼이 클릭 되었다. <i>target</i> : 클릭된 버튼. CLIK Button 타입

다음 예는 ButtonGroup 의 어떤 버튼이 선택되었는지 결정하는 코드다.

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

2.2.3 ButtonBar

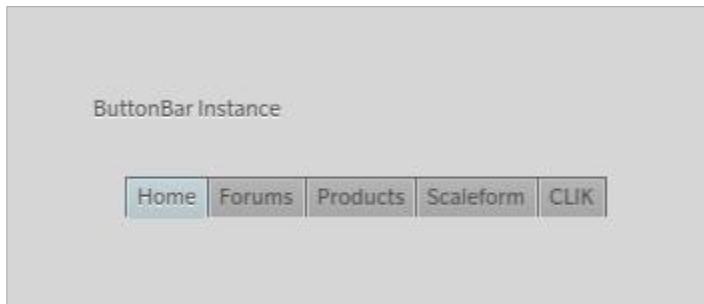


Figure 22: Unskinned ButtonBar.

ButtonBar (scaleform.clik.controls.ButtonBar)는 ButtonGroup 과 비슷한데, 비쥬얼한 모습을 갖고 있다는 점이 다르다. 또한, dataProvider(프로그래밍 세부사항 장 참조)에 기반해서 즉시 Button 인스턴스를 만들수도 있다. ButtonBar 는 동적 탭바(tab-bar)같은 UI 요소를 만들 때 유용하다. 이 컴포넌트는 dataProvider 를 적용함으로써 활성화 된다.

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

2.2.3.1 사용자 상호작용

ButtonBar 는 ButtonGroup 과 동일하게 작동한다. 그리고, 사용자가 직접적인 상호 작용은 못하지만 관리하는 Button 이 클릭되면 간접적으로 영향을 받는다.

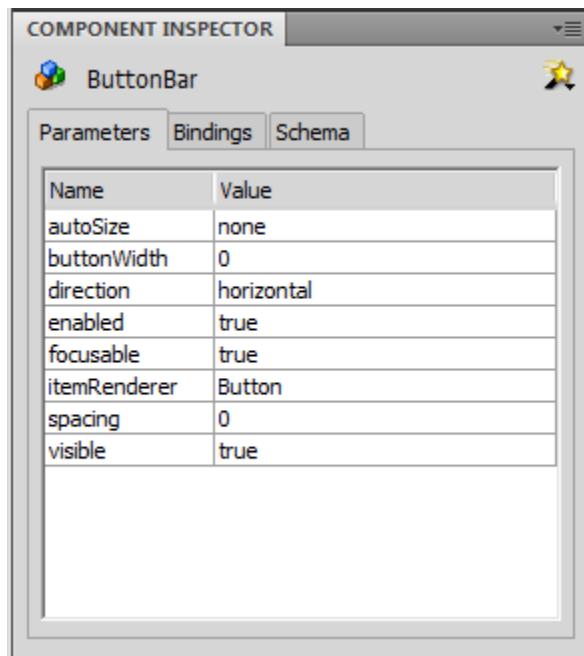
2.2.3.2 컴포넌트 설정

CLIK ButtonBar 클래스를 사용하는 MovieClip은 아무리 부요소가 필요없는데, 이는 ButtonBar가 실행시에 MovieClip을 생성하기 때문이다.

2.2.3.3 상태

CLIK ButtonBar는 아무런 비쥬얼 상태를 갖지 않는다. 이는 관리하는 버튼의 그룹 상태를 출력하기 때문이다.

2.2.3.4 점검가능 속성



ButtonBar가 아무런 내용물을 갖지는 않지만(플래시 IDE의 스테이지에서 조그만 원으로 나타날 뿐이다), 몇가지 점검가능 속성 값을 갖는다. 주된 용도는 ButtonBar로 생성된 Button 인스턴스의 위치설정에 관련된 것이다.

autoSize true인 경우 표시되는 라벨에 맞도록 버튼 인스턴스의 크기를 변경합니다.

Direction 버튼 배치입니다. 수평은 버튼 인스턴스를 옆으로 배치하고, 수직은 버튼 인스턴스를 서로의 위로 쌓습니다.

buttonWidth 버튼 인스턴스의 공통 넓이를 설정합니다. autoSize를 true로 설정하면

	이 프로퍼티가 무시됩니다.
enabled	false 로 설정하면 버튼을 비활성화합니다. 비활성화한 구성 요소는 더 이상 사용자 입력을 받지 않습니다.
focusable	구성 요소에 대한 focus 관리를 활성화/비활성화합니다. focus 가능한 프로퍼티를 false 로 설정하면 tab 키, 방향 키 및 마우스 기반 focus 변경에 대한 지원을 제거합니다.
itemRenderer	Button 컴포넌트 심볼의 링크 ID. 이 심볼은 ButtonBar 에 적용되는 데이터에 기반해서 필요할 때 인스턴스화 된다.
spacing	Button 인스턴스들 간의 공간. 현재 direction 에만 영향을 미친다(direction 항목 참고).
Visible	false 로 설정하면 ButtonBar 를 숨깁니다.

2.2.3.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- **type**: 이벤트의 형식입니다.
- **target**: 이벤트 대상입니다.
- **currentTarget**: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- **eventPhase**: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- **bubbles**: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- **cancelable**: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

ButtonBar 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true 로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false 로 설정되었습니다.
FocusHandlerEvent.FOCUS_IN	버튼이 focus 를 받았습니다.

FocusHandlerEvent.FOCUS_OUT	버튼이 focus 를 잃었습니다.
IndexEvent.INDEX_CHANGE	<p>그룹에서 새 버튼이 선택되었습니다.</p> <p><i>index</i>: ButtonBar 의 선택된 인덱스입니다. int 형식입니다. 버튼 개수에서 1 을 뺀 -1 이 값입니다(현재 선택된 항목이 없는 경우).</p> <p><i>lastIndex</i>: ButtonBar 의 이전에 선택된 인덱스입니다. int 형식입니다. 버튼 개수에서 1 을 뺀 -1 이 값입니다(이전에 선택된 항목이 없는 경우).</p> <p><i>data</i>: 선택된 dataProvider 의 데이터 값입니다. AS3 Object 형식입니다.</p>
ButtonBarEvent.BUTTON_SELECT	<p>그룹에서 새 버튼이 선택되었습니다.</p> <p><i>index</i>: ButtonBar 의 선택된 인덱스입니다. Int 형식입니다. 버튼 개수에서 1 을 뺀 -1 이 값입니다(현재 선택된 항목이 없는 경우).</p> <p><i>renderer</i>: ButotnBar 내에서 버튼 인스턴스가 현재 선택되었습니다. 버튼 형식입니다.</p>

ButtonBar 내에서 Button 인스턴스가 클릭되었을 때를 탐색하는 예

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

2.3 스크롤 타입(Scroll Types)

스크롤 타입은 ScrollIndicator, ScrollBar, Slider 컴포넌트로 구성되어 있다. ScrollIndicator 는 상호작용 없이 썬(thumb)을 사용해서 간단하게 타겟 컴포넌트의 스크롤 인덱스를 보여주고, ScrollBar 는 스크롤 위치를 조정하도록 사용자 상호작용을 지원한다. ScrollBar 는 4 개의 버튼으로 구성되어 있는데, 썬(thumb 혹은 grip), 트랙(track), 위쪽 화살표, 아래쪽 화살표가 그것이다. Slider 는 ScrollBar 와 비슷하지만 thumb 과 track 으로만 구성되며 타겟 컴포넌트의 요소 개수에 따라서 썬의 크기를 조절하지 않는다는 것이 다르다.

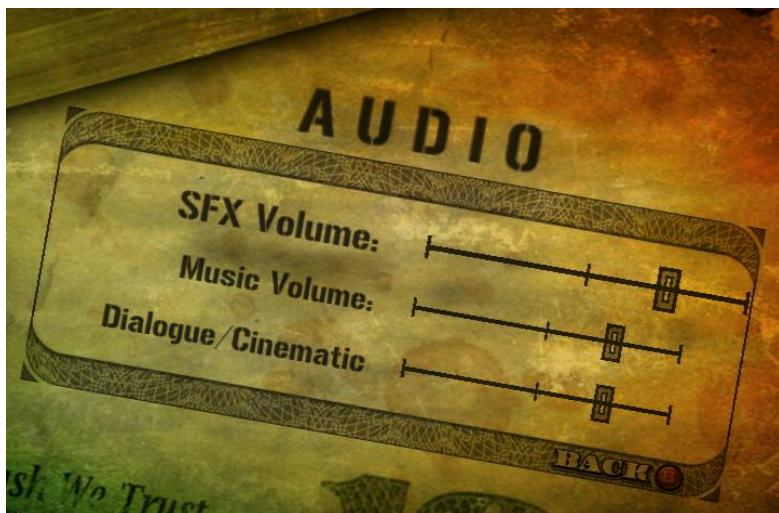


그림 23: *Mercenaries 2*의 자동 메뉴 Slider 예제

2.3.1 ScrollIndicator



그림 24: 스킨없는 ScrollIndicator.

CLIK ScrollIndicator(scaleform.clik.controls.ScrollIndicator)는 multiline textField 같은 다른 컴포넌트의 스크롤 위치를 보여준다. 스크롤 위치를 자동적으로 보여주기 위해서 textField에서 위치를 가리킬 수

있다. TextArea 를 포함한 모든 리스트 기반 컴포넌트는 scrollBar 속성을 갖기 때문에 ScrollIndicator 나 ScrollBar 인스턴스(혹은 링크 ID)에 의해서 위치를 가리킬 수 있다.

2.3.1.1 사용자 상호작용

ScrollIndicator 는 아무런 사용자 상호작용이 없다. 오직 보여줄 뿐이다.

2.3.1.2 컴포넌트 설정

CLIK ScrollIndicator 클래스를 사용하는 MovieClip 은 다음과 같은 부요소를 갖고 있어야만 한다.

- **thumb**: CLIK 버튼 타입. 스크롤 지시자 손잡이
- **track**: MovieClip 타입. 스크롤 지시자 트랙. 트랙의 경계는 손잡이가 이동할 수 있는 크기를 나타낸다.

2.3.1.3 상태

ScrollIndicator 는 명시적 상태가 없다. 자식요소의 상태를 사용할 뿐이다. 여기서 자식요소란 썸과 트랙 버튼 컴포넌트다.

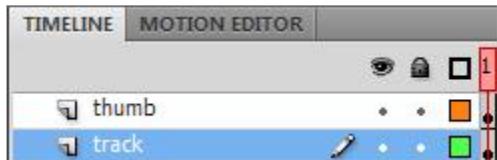
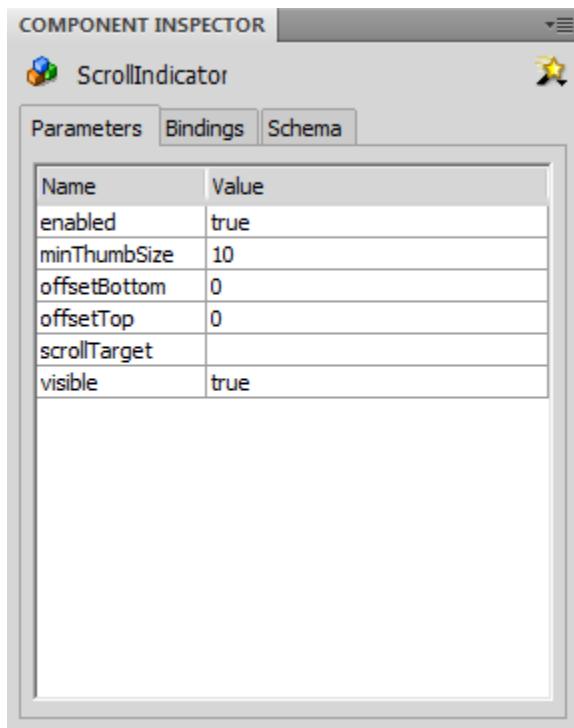


그림 25: ScrollIndicator 타임라인.

2.3.1.4 점검가능 속성



ScrollIndicator 의 점검가능 속성은 다음과 같다.

enabled	false 로 설정하면 구성 요소를 비활성화합니다. 비활성화한 구성 요소는 더 이상 사용자 입력을 받지 않습니다.
offsetTop	썸(Thumb)의 오프셋을 최 상단으로 설정함. 양수는 썸을 제일 높은 위치로 움직이게 한다.
offsetBottom	썸의 오프셋을 최 하단으로 설정한다. 양수는 썸을 제일 낮은 위치로 움직이게 한다.
scrollTarget	TextArea 나 일반 여러줄 textField 를 스크롤 타겟으로 해서 스크롤 이벤트에 대하여 자동적으로 응답하도록 한다. textField 타입이 아닌 경우에는 ScrollIndicator 속성을 수동으로 업데이트 해야 한다.
Visible	false 인 경우 컴포넌트를 숨긴다.

2.3.1.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- ***type***: 이벤트의 형식입니다.
- ***target***: 이벤트 대상입니다.
- ***currentTarget***: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- ***eventPhase***: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- ***bubbles***: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- ***cancelable***:
- 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

ScrollIndicator 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW 보이는 프로퍼티가 실행 시 true로 설정되었습니다.

ComponentEvent.HIDE 보이는 프로퍼티가 실행 시 false로 설정되었습니다.

Event.SCROLL ScrollIndicator 가 스크롤되었습니다.

다음은 스크롤 이벤트를 대기하는 코드다.

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

2.3.2 ScrollBar



그림 26: 스키 없는 ScrollBar.

CLIK ScrollBar (scaleform.clik.controls.ScrollBar)는 다른 컴포넌트의 스크롤 위치를 출력하고 제어한다. ScrollIndicator에 드래그 가능한 썸버튼을 추가하고, 옵션 상하 화살표버튼, 클릭 가능한 트랙을 추가해서 상호 작용 가능하게 한다.

2.3.2.1 사용자 상호작용

ScrollBar 썸은 마우스나 동등한 컨트롤러로 잡을 수 있으며 스크롤바 트랙의 경계 내에서 드래그 가능하다. 마우스 커서가 ScrollBar 위에 있을 때 마우스 휠을 움직이면 스크롤을 동작을 한다. 트랙을 클릭하면 2 가지 작동을 한다. 썸이 클릭 된 방향으로 연속적으로 스크롤 되거나, 즉시 그 지점으로 점프하는 것이다. 이러한 트랙모드는 점검가능 속성인 *trackMode*에 의해서 결정된다. *trackMode* 설정과 무관하게 [SHIFT]키와 함께 트랙을 클릭하면 썸을 즉시 그 위치로 이동시켜준다.

2.3.2.2 컴포넌트 설정

CLIK ScrollBar 클래스를 사용하는 MovieClip은 반드시 다음과 같이 명명된 부요소를 가져야 한다.

- **up 화살표:** Clik Button 타입. 스크롤 업 된 Button; 일반적으로 스크롤바 맨 위에 위치해 있다.
- **downArrow:** Clik Button 타입. 스크롤다운 된 Button; 일반적으로 스크롤바 맨 밑에 위치해 있다..
- **thumb:** Clik Button 타입. 스크롤바 손잡이
- **track:** Clik Button 타입. 스크롤바 트랙의 (scrollbar track) 경계는 손잡이(grip)가 이동할 수 있는 크기를 결정한다.

2.3.2.3 상태

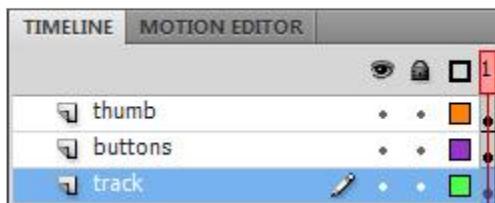
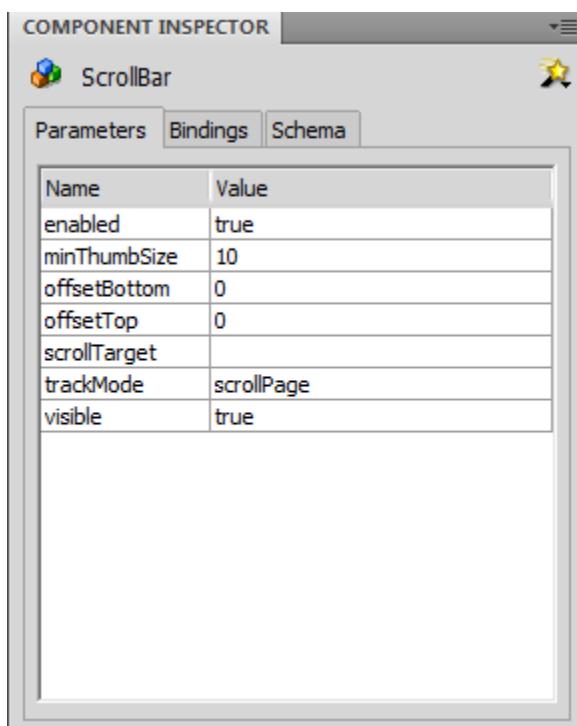


그림 27: ScrollBar timeline./ ScrollBar 타임라인

ScrollBar 는 스크롤지시자와 비슷하게, 명시적 상태가 없다. 자식요소의 상태를 이용할 뿐이다. 여기서 자식요소란 썸(thumb), up, down, 그리고 트랙 버튼 컴포넌트다.

2.3.2.4 점검가능 속성



ScrollBar 는 ScrollIndicator 의 점검가능 속성과 유사하며, 하나가 더 추가된다.

, trackMode:

enabled	false 로 설정하면 구성 요소를 비활성화합니다. 비활성화한 구성 요소는 더 이상 사용자 입력을 받지 않습니다.
offsetTop	썸(Thumb)의 오프셋을 최 상단으로 설정함. 양수는 썸을 제일 높은 위치로 움직이게 한다.

offsetBottom	썸의 오프셋을 최 하단으로 설정한다. 양수는 썸을 제일 낮은 위치로 움직이게 한다.
scrollTarget	TextArea 나 일반 여러줄 textField 를 스크롤 타겟으로 해서 스크롤 이벤트에 대하여 자동적으로 응답하도록 한다. textField 타입이 아닌 경우에는 ScrollIndicator 속성을 수동으로 업데이트 해야 한다.
trackMode	사용자가 트랙을 커서로 클릭했을 때 scrollPage 로 설정하면 커서가 떠어질 때까지 썸이 연속적으로 페이지 단위로 스크롤된다. scrollToCursor 로 설정하면 썸이 즉시 커서 지점으로 점프하며 썸을 드래그 모드로 전이 시킨다.
Visible	false 인 경우 컴포넌트를 숨긴다.

2.3.2.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- **type**: 이벤트의 형식입니다.
- **target**: 이벤트 대상입니다.
- **currentTarget**: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- **eventPhase**: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- **bubbles**: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- **cancelable**: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

ScrollBar 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW

보이는 프로퍼티가 실행 시 true 로

	설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false 로 설정되었습니다.
Event.SCROLL	ScrollBar 가 스크롤되었습니다.

다음은 스크롤 이벤트를 대기하는 코드다.

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

2.3.3 Slider



그림 28: 스키드없는 Slider.

Slider (scaleform.clik.controls.Slider)는 범위내의 수치값을 출력한다. 이때 썸은 현재 값을 나타내기도 하면서 이를 드래그해서 값을 수정할 수도 있다.

2.3.3.1 사용자 상호작용

Slider 썸은 마우스나 동등한 컨트롤러로 잡을 수 있으며 Slider 트랙의 경계 내에서 드래그 가능하다. 트랙을 클릭하면 썸을 커서 위치로 즉시 이동시킨다. 포커스가 있을 때 좌우 화살표키를 통해서 적절한 방향으로 썸을 이동 시킬수 있으며, [HOME], [END]키는 썸을 트랙의 시작점과 끝점으로 이동시킨다.

트래글 클릭하면 2 가지 작동을 한다. 썸이 클릭된 방향으로 연속적으로 스크롤 되거나, 즉시 그 지점으로 점프하는 것이다. 이러한 트랙모드는 점검가능 속성인 *trackMode*에 의해서 결정된다. *trackMode* 설정과 무관하게 [SHIFT]키와 함께 트랙을 클릭하면 썸을 즉시 그 위치로 이동시켜준다.

2.3.3.2 컴포넌트 설정

CLIK Slider 클래스를 사용하는 MovieClip 은 반드시 다음과 같이 명명된 부요소를 가져야 한다.

- **thumb**: CLIK Button 타입. 슬라이더 손잡이
- **track**: CLIK Button 타입. 슬라이더 트랙. 경계는 손잡이(grip)가 이동할 수 있는 크기를 결정한다.

2.3.3.3 상태

ScrollBar 와 ScrollIndicator 와 비슷하게 Slider 는 명시적인 상태가 없으며 자식 요소(thumb, track)의 상태를 사용한다.

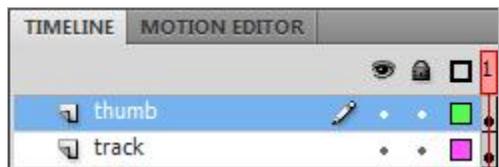
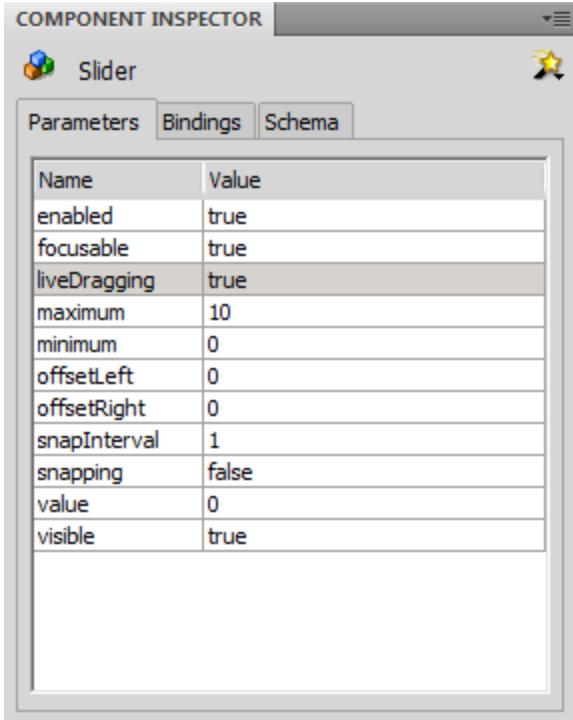


그림 29: Slider 타임라인

2.3.3.4 점검가능 속성



Slider의 점검가능 속성은 다음과 같다.

Enabled	false로 설정하면 버튼을 비활성화합니다. 비활성화한 구성 요소는 더 이상 사용자 입력을 받지 않습니다.
Focusable	구성 요소에 대한 focus 관리를 활성화/비활성화합니다. focus 가능한 프로퍼티를 false로 설정하면 tab 키, 방향 키 및 마우스 기반 focus 변경에 대한 지원을 제거합니다.
Value	Slider에 출력될 수치값
minimum	Slider 범위의 최소값
maximum	Slider 범위의 최대값
snapping	이 값이 true라면 썸이 snapInterval과 곱해진 값으로 스냅 된다.
snapInterval	스네핑 구간은 썸이 스냅 될 때 무엇과 곱해질지를 결정하는 것이다. snapping을 false로 하면 작동하지 않는다.
liveDragging	이 값이 true면 Slider가 썸을 드래그하는 중에 변경 이벤트를 발생시킨다. 만약 false라면 드래그가 완료되었을 때 한번만 변경 이벤트를 발생시킬 것이다.

offsetLeft	썸의 왼쪽에 대한 오프셋. 양수이면 슬라이더의 안쪽으로 움직임이 제한된다.
offsetRight	썸의 오른쪽에 대한 오프셋. 양수이면 슬라이더의 안쪽으로 움직임이 제한된다.
Visible	false로 설정하면 구성 요소를 숨깁니다.

2.3.3.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- **type**: 이벤트의 형식입니다.
- **target**: 이벤트 대상입니다.
- **currentTarget**: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- **eventPhase**: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- **bubbles**: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- **cancelable**: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

Slider 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false로 설정되었습니다.
ComponentEvent.STATE_CHANGE	구성 요소의 상태가 변경되었습니다.
FocusHandlerEvent.FOCUS_IN	버튼이 focus를 받았습니다.
FocusHandlerEvent.FOCUS_OUT	버튼이 focus를 잃었습니다.
SliderEvent.VALUE_CHANGE	슬라이더 값이 변경되었다.

다음은 Slider 값 변경을 대기하는 방법이다.

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

2.4 리스트 타입(List Types)

CLIK 리스트 타입은 NumericStepper, OptionStepper, ScrollingList, TileList, DropdownMenu 컴포넌트로 구성되어 있다. NumericStepper 를 제외한 이들 모든 컴포넌트는 출력되는 정보를 관리하기 위해서 DataProvider 와 함께 작업한다. ListItemRenderer 컴포넌트도 이 범주에 속하는데, 이는 ScrollingList 와 TileList 컴포넌트가 항목 리스트를 출력할 때 사용하기 때문이다.

NumericStepper 와 OptionStepper 는 동시에 하나의 요소만 출력한다. 하지만 ScrollingList 와 TileList 는 하나 이상을 출력할 수 있다. 뒤에나온 2 개의 컴포넌트는 ScrollIndicator 나 ScrollBar 컴포넌트도 지원할 수 있다. DropdownMenu 컴포넌트는 아이들 상태(idle state)일 때 하나의 요소를 출력하지만 ScrollingList 나 TileList 를 사용해서 더 많은 요소를 출력할 수도 있다.



그림 30: *Mercenaries 2*의 ScrollIndicator 가 있는 ScrollingList 사용 예제

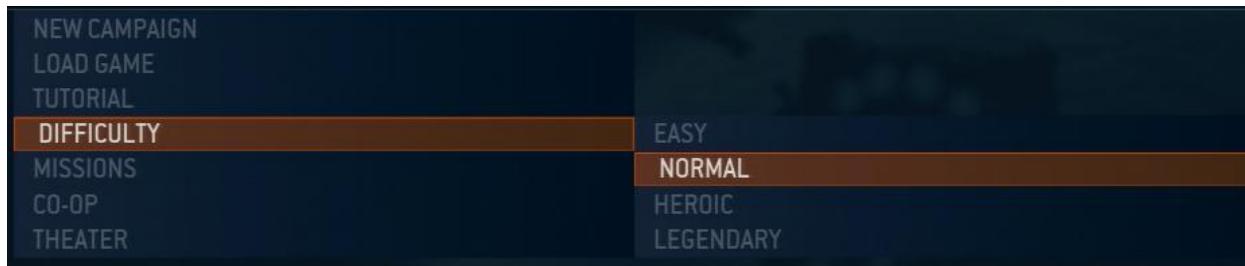


그림 31: *Halo Wars* 의 '난이도 설정' DropdownMenu 사용 예제

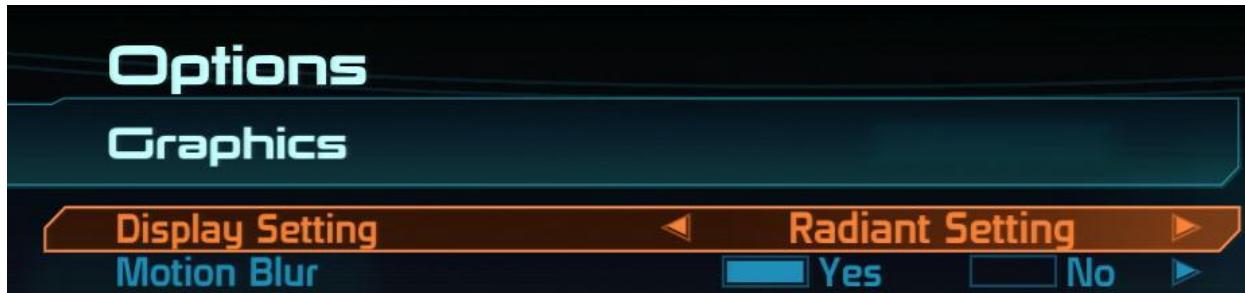


그림 32: from *Mass Effect* 의 '출력 설정' OptionStepper 사용 예제

2.4.1 NumericStepper



그림 33: 스킨없는 Numeric Stepper.

NumericStepper (scaleform.clik.controls.NumericStepper) 는 적용된 범위내의 단일 숫자를 추력하며, 임의의 간격에 기반한 수치 증감능력을 가지고 있다.

2.4.1.1 사용자 상호작용

NumericStepper 는 2 개의 화살표 버튼을 가지고 있으며 이 버튼을 마우스나 동등한 컨트롤로 클릭하면 수치값 변경을 제어할 수 있다. 포커스를 가졌을 때는 수치값을 키보드의 좌우화살표나 동등한 컨트롤에 의하여 변경 가능하다. 이를 키는 현재 값을 step 값에 따라서 증감시킨다. [HOME]과 [END]키를 누르거나 혹은 동등한 컨트롤에 의하여 수치값을 최대, 최소로 각각 변경시킬 수 있다.

2.4.1.2 컴포넌트 설정

NumericStepper 클래스를 사용하는 MovieClip 은 다음과 같은 부요소를 갖고 있어야만 한다

- **textField**: TextField 타입. 현재값을 보여준다.
- **nextBtn**: CLIK Button 타입. 이 버튼을 클릭하면 현재 값을 step 만큼 증가시킨다.
- **prevBtn**: CLIK Button 타입. 이 버튼을 클릭하면 현재 값을 step 만큼 감소시킨다.

2.4.1.3 상태

NumericStepper 컴포넌트는 포커스와 disabled 속성에 기반한 3 가지 상태를 지원한다.

- **default** 혹은 enabled 상태
- **focused** 상태, 일반적으로 textField 의 경계면에 하이라이트를 출력해서 상태를 나타낸다

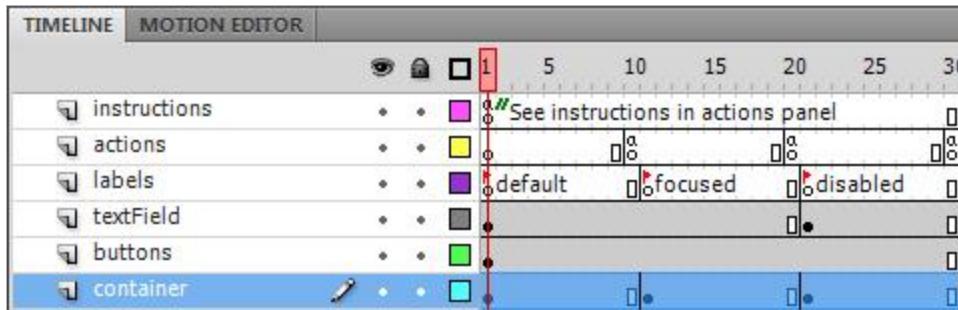
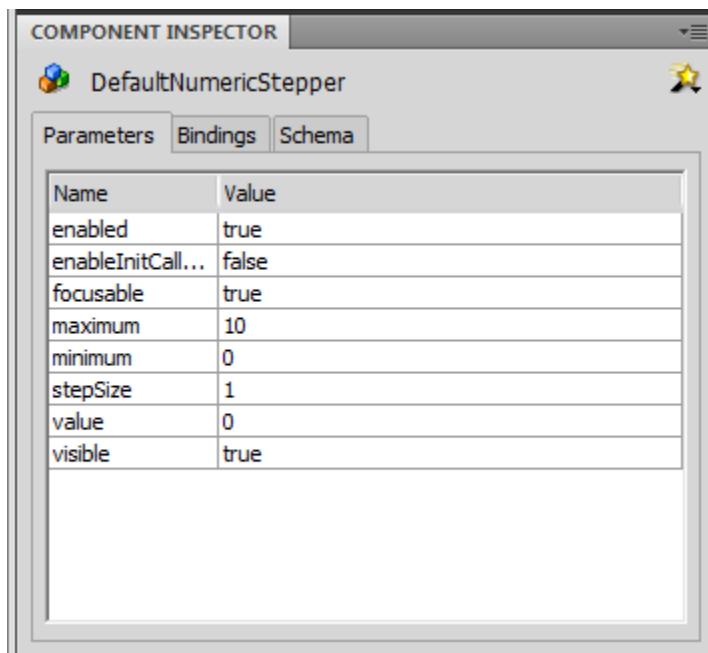


그림 34: NumericStepper 타임라인

2.4.1.4 점검가능 속성



NumericStepper로부터 유도된 MovieClip은 다음과 같은 속성을 갖는다.

Enabled	false로 설정하면 버튼을 비활성화합니다. 비활성화한 구성 요소는 더 이상 사용자 입력을 받지 않습니다.
Focusable	구성 요소에 대한 focus 관리를 활성화/비활성화합니다. focus 가능한 프로퍼티를 false로 설정하면 tab 키, 방향 키 및 마우스 기반 focus 변경에 대한 지원을 제거합니다.
minimum	NumericStepper 범위값의 최소치
maximum	NumericStepper 범위값의 최대치
stepSize	NumericStepper의 버튼 중 하나가 클릭될 때마다 NumericStepper 값이

	증가/감소되는 양입니다.
Value	NumericStepper 에 의하여 출력될 수치값
Visible	false 로 설정하면 구성 요소를 숨깁니다.

2.4.1.5 이벤트

모든 콜백은 단일 Object 인자를 받는데, 여기에는 이벤트에 관한 관련정보가 들어 있다. 다음 속성은 모든 이벤트에 대한 공통속성이다.

- **type**: 이벤트 타입
- **target**: 이벤트를 생성한 타겟

NumericStepper 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true 로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false 로 설정되었습니다.
ComponentEvent.STATE_CHANGE	구성 요소의 상태가 변경되었습니다.
FocusHandlerEvent.FOCUS_IN	구성 요소가 focus 를 받았습니다.
FocusHandlerEvent.FOCUS_OUT	구성 요소가 focus 를 잃었습니다.
ComponentEvent.STATE_CHANGE	구성 요소의 상태가 변경되었습니다.
IndexEvent.INDEX_CHANGE	NumericStepper 의 값이 변경되었습니다. <i>index</i> : NumericStepper 의 선택된 인덱스입니다. int 형식입니다. 버튼 개수에서 1 을 뺀 -1 이 값입니다(현재 선택된 항목이 없는 경우). <i>lastIndex</i> : NumericStepper 의 이전에 선택된 인덱스입니다. int 형식입니다. 버튼 개수에서 1 을 뺀 -1 이 값입니다(이전에 선택된 항목이 없는 경우). <i>data</i> : 선택된 dataProvider 의 데이터 값입니다. AS3 Object 형식입니다.

NumericStepper 의 값 변경을 대기

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

2.4.2 OptionStepper



그림 35: 스키 없는 OptionStepper.

OptionStepper (scaleform.clik.controls.OptionStepper)는 NumericStepper 와 비슷하게 단일 값을 출력하지만, 숫자 외의 값도 출력할 수 있다. OptionStepper 는 현재 값을 알아내기 위해서dataProvider 인스턴스를 사용한다. 따라서, 다른 타입의 요소를 임의의 개수만큼 지원할 수 있다. 출력되는 값은 OptionStepper 의 selectedIndex 속성을 코드를 사용해서 설정한다. dataProvider 는 다음과 같이 hem 를 통해서 적용된다.

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

2.4.2.1 사용자 상호작용

NumericStepper 와 비슷하게 OptionStepper 도 2 개의 화살표 버튼을 가지고 있으며 이 버튼을 마우스나 동등한 컨트롤로 클릭하면 수치값 변경을 제어할 수 있다. 포커스를 가졌을 때는 현재값을 키보드의 좌우화살표나 동등한 컨트롤에 의하여 변경 가능하다. 이들 키는 현재값을 직전과 직후 값으로 변경시킨다. [HOME]과 [END]키를 누르거나 혹은 동등한 컨트롤에 의하여 현재값을 dataProvider 내의 첫번째와 마지막 값으로 변경시킬 수 있다.

2.4.2.2 컴포넌트 설정

CLIK NumericStepper 클래스를 사용하는 MovieClip 은 반드시 다음과 같이 명명된 부요소를 가져야 한다.

textField: TextField 타입. 현재값을 보여준다.

- **nextBtn**: CLIK Button 타입. 이 버튼을 클릭하면 현재 값을 dataProvider 내의 다음 요소로 변경한다.
- **prevBtn**: CLIK Button 타입. 이 버튼을 클릭하면 현재 값을 dataProvider 내의 직전 요소로 변경한다.

2.4.2.3 상태

OptionStepper 컴포넌트는 포커스와 disabled 속성에 기반한 3 가지 상태를 지원한다.

- **default** 혹은 enabled 상태
- **focused** 상태, 일반적으로 textField 의 경계면에 하이라이트를 출력해서 상태를 나타낸다
- **disabled** 상태

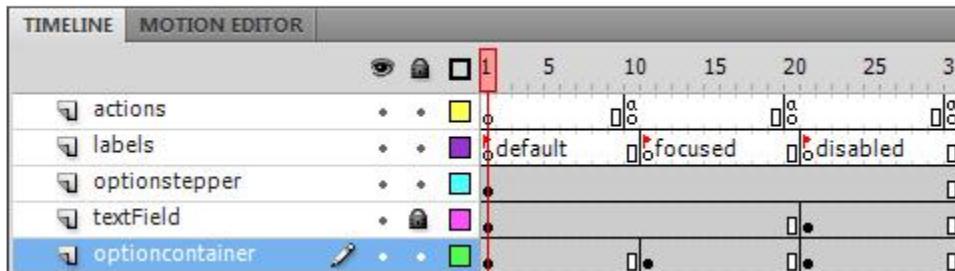
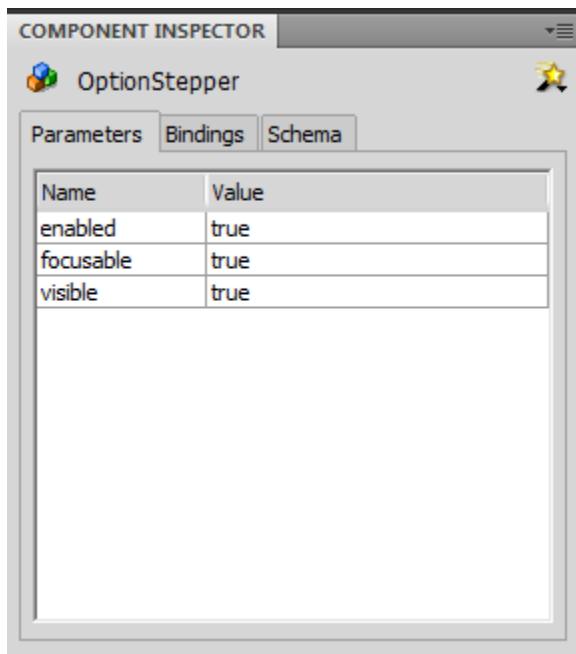


그림 36: OptionStepper 타임라인

2.4.2.4 점검가능 속성



OptionStepper로부터 유도된 MovieClip은 다음과 같은 속성을 갖는다.

Enabled	false로 설정하면 버튼을 비활성화합니다. 비활성화한 구성 요소는 더 이상 사용자 입력을 받지 않습니다.
Focusable	구성 요소에 대한 focus 관리를 활성화/비활성화합니다. focus 가능한 프로퍼티를 false로 설정하면 tab 키, 방향 키 및 마우스 기반 focus 변경에 대한 지원을 제거합니다.
Visible	false로 설정하면 구성 요소를 숨깁니다.

2.4.2.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- **type**: 이벤트의 형식입니다.
- **target**: 이벤트 대상입니다.
- **currentTarget**: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.

- **eventPhase**: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- **bubbles**: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- **cancelable**: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

OptionStepper 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true 로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false 로 설정되었습니다.
ComponentEvent.STATE_CHANGE	구성 요소의 상태가 변경되었습니다.
FocusHandlerEvent.FOCUS_IN	구성 요소가 focus 를 받았습니다.
FocusHandlerEvent.FOCUS_OUT	구성 요소가 focus 를 잃었습니다.
IndexEvent.INDEX_CHANGE	OptionStepper 의 값이 변경되었습니다. <i>index</i> : NumericStepper 의 선택된 인덱스입니다. int 형식입니다. 버튼 개수에서 1 을 뺀 -1 이 값입니다(현재 선택된 항목이 없는 경우). <i>lastIndex</i> : NumericStepper 의 이전에 선택된 인덱스입니다. int 형식입니다. 버튼 개수에서 1 을 뺀 -1 이 값입니다(이전에 선택된 항목이 없는 경우). <i>data</i> : 선택된dataProvider 의 데이터 값입니다. AS3 Object 형식입니다.

OptionStepper 의 값 변경을 대기하는 방법

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

2.4.3 ListItemRenderer



그림 37: 스킨없는 ListItemRenderer.

ListItemRenderer (`scaleform.clik.controls.ListItemRenderer`) 는 CLIK Button 클래스로부터 파생되었으며 Button 클래스를 확장해서 리스트관련 속성을 포함하고 있기 때문에 컨테이너 컴포넌트에 유용하다. 하지만, 독립적인 컴포넌트로 디자인된 것이 아니기 때문에 ScrollingList, TileList 그리고 DropdownMenu 와 함께 사용하여야 한다.

2.4.3.1 사용자 상호작용

ListItemRenderer 이 Button 컴포넌트로부터 파생되었으므로 Button 과 비슷한 사용자 상호작용 구조를 갖고 있다. 마우스 버튼을 사용해서 누를수 있으며, 마우스를 ListItemRenderer 위에서 이동하거나 벗어나게 하면 ListItemRenderer 에 영향을 주게 되며, 마우스 커서를 드래그할 수도 있다. 키보드나 동급 컨트롤러 상호작용은 ListItemRenderer 의 컨테이너 컴포넌트에 따라서 정의된다.

2.4.3.2 컴포넌트 설정

CLIK ListItemRenderer 클래스를 사용하는 MovieClip 은 반드시 다음과 같이 명명된 부요소를 가져야 한다.

- **textField**. (optional) TextField 타입. 리스트 항목 레이블
- **focusIndicator**. (optional) MovieClip 형. 분리된 MovieClip 이 포커스된 상태 출력에 사용된다. 실제 존재하는 경우에는 MovieClip 이 show 와 hide 라 명명된 프레임을 갖고 있어야 한다.

2.4.3.3 상태

컨테이너 컴포넌트 내부에서 선택될 수 있으므로, ListItemRenderer 는 선택된 상태를 표현하기 위한 *selected* 키프레임이 필요하다.

- ***up*** 혹은 기본 상태
- ***over*** 상태는 마우스 커서가 위에 있을 때, 혹은 포커스를 가졌을 때
- ***down*** 상태는 버튼이 눌렸을 때
- ***disabled*** 상태
- ***selected_up*** 혹은 기본상태
- ***selected_over*** 상태는 마우스 커서가 위에 있을 때, 혹은 포커스를 가졌을 때.
- ***selected_down*** 상태는 버튼이 눌렸을 때
- ***selected_disabled*** 상태

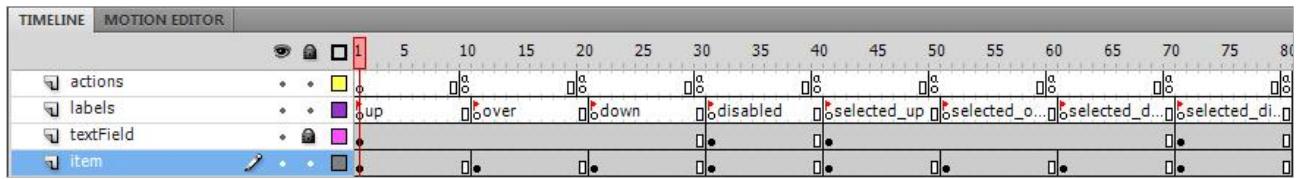
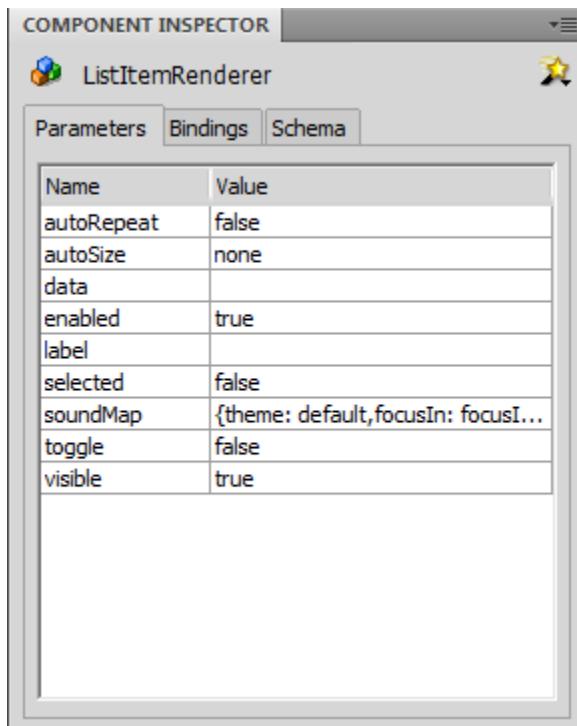


그림 38: ListItemRenderer 타임라인.

이들은 ListItemRenderer에서 필요한 최소한의 키프레임이다. Button 컴포넌트(결과적으로 ListItemRenderer 컴포넌트도 포함)에서 지원되는 확장 상태 집합과 키프레임은 [Getting Started with CLIK Buttons](#) 문서를 참고하라.

2.4.3.4 점검가능 속성



ListItemRenderer는 컨테이너 컴포넌트에 의해서 제어되며, 절대로 사용자가 수동으로 설정할 수 없기 때문에 Button의 점검가능 속성 중에서 작은 부분집합만 가진다.

autoRepeat	누르고 유지할 때 ListItemRender 가 "click" 이벤트를 보낼지 여부를 결정합니다.
autoSize	포함된 텍스트에 맞춰서 ListItemRender 크기가 조절되고, 크기가 조절된 ListItemRender 를 정렬할 방향을 결정합니다. autoSize 프로퍼티를 autoSize=TextFieldAutoSize.NONE 으로 설정하면 크기를 변경하지 않고 그대로 놔둡니다.
Data	ListItemRender 관련 데이터입니다. 이 프로퍼티는 ListItemRender 를 목록 구성 요소(ScrollingList/TileList)에서 사용할 때 특히 도움이 됩니다.
enabled	false 로 설정하면 구성 요소를 비활성화합니다. 비활성화한 구성 요소는 더 이상 사용자 입력을 받지 않습니다.
Label	ListItemRenderer 의 레이블을 설정한다.
Selected	ListItemRender 의 선택된 상태를 설정합니다. ListItemRender 는 선택과

	<p>비선택의 두 가지 마우스 상태 세트를 가질 수 있습니다.</p> <p>ListItemRender 의 전환 프로퍼티가 true 일 때 선택된 상태는 버튼을 클릭하면 변경되지만, 선택된 상태는 전환 프로퍼티가 false 인 경우에도 ActionScript 를 사용해서 설정될 수 있습니다.</p>
Toggle	<p>ListItemRender 의 전환 프로퍼티를 설정합니다. true 인 경우 ListItemRender 는 전환 버튼 역할을 수행합니다.</p>
Visible	false 로 설정하면 구성 요소를 숨깁니다.

2.4.3.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- **type**: 이벤트의 형식입니다.
- **target**: 이벤트 대상입니다.
- **currentTarget**: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- **eventPhase**: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- **bubbles**: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- **cancelable**: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

ListItemRenderer 구성 요소에 의해 생성된 이벤트는 아래와 같습니다. 이벤트 옆에 나열된 프로퍼티는 공통 프로퍼티에 추가적으로 제공됩니다.

일반적으로 ListItemRenderer 가 보낸 이벤트를 수신하는 것보다는 사용자가 목록 자체에 이벤트 리스너를 추가하는 것이 좋습니다. 자식 ListItemRenderer 와 상호 작용이 발생하면 목록은 ITEM_PRESS 및 ITEM_CLICK 을 포함한 ListEvent 를 보냅니다. 사용자는 EventListener 하나를 목록의 각 ListItemRenderer 에 추가하는 대신에 EventListener 하나를 마우스 클릭 이벤트(ListEvent.ITEM_CLICK)에 대한 목록에 추가할 수 있습니다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true 로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false 로 설정되었습니다.
ComponentEvent.STATE_CHANGE	구성 요소의 상태가 변경되었습니다.
FocusHandlerEvent.FOCUS_IN	구성 요소가 focus 를 받았습니다.
FocusHandlerEvent.FOCUS_OUT	구성 요소가 focus 를 잃었습니다.
Event.SELECT	선택된 프로퍼티가 변경되었습니다.
MouseEvent.ROLL_OVER	<p>마우스 커서가 버튼 위에 놓였습니다.</p> <p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.</p> <p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.</p>
MouseEvent.ROLL_OUT	<p>마우스 커서가 버튼 밖으로 벗어났습니다.</p> <p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.</p> <p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx 로 보내야 합니다.</p>
ButtonEvent.PRESS	<p>버튼이 눌렸습니다.</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p> <p><i>isKeyboard</i>: 이벤트가 키보드/게임패드에 의해 생성되었으면 true 이고 이벤트가 마우스에 의해</p>

	<p>생성되었으면 false입니다.</p> <p><i>isRepeat</i>: 이벤트가 autoRepeating 버튼을 길게 눌러 생성되었으면 true이고 버튼이 현재 반복 중이 아니면 false입니다.</p>
MouseEvent.DOUBLE_CLICK	<p>버튼이 두 번 클릭되었습니다. <i>doubleClickEnabled</i> 프로퍼티가 true인 경우에만 일어납니다.</p> <p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로 이벤트를 MouseEventEx로 보내야 합니다.</p> <p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로 이벤트를 MouseEventEx로 보내야 합니다.</p>
ButtonEvent.CLICK	<p>버튼이 클릭되었습니다.</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p> <p><i>isKeyboard</i>: 이벤트가 키보드/게임패드에 의해 생성되었으면 true이고 이벤트가 마우스에 의해 생성되었으면 false입니다.</p> <p><i>isRepeat</i>: 이벤트가 autoRepeating 버튼을 길게 눌러 생성되었으면 true이고 버튼이 현재 반복 중이 아니면 false입니다.</p>
ButtonEvent.DRAG_OVER	<p>마우스 커서가 버튼위로 드래그 되었다(왼쪽 버튼이 눌린 채로).</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p>
ButtonEvent.DRAG_OUT	마우스 커서가 버튼 바깥으로 드래그 되었다(왼쪽

버튼이 눌린 채로).

controllerIdx: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.

ButtonEvent.RELEASE_OUTSIDE

마우스 커서가 버튼 바깥으로 드래그 되었고, 그 상태로 마우스 왼쪽 버튼이 떼어졌다.

controllerIdx: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.

2.4.4 ScrollingList

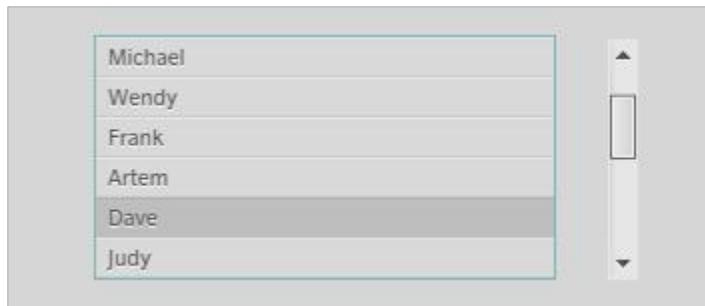


그림 39: 스킨없는 ScrollingList.

ScrollingList (scaleform.clik.controls.ScrollingList)는 내부 요소를 스크롤 할 수 있는 리스트 컴포넌트다. 이 컴포넌트는 직접 리스트 아이템을 인스턴스화 할 수도 있고, 혹은 스테이지상에서 존재하는 리스트 아이템을 사용할 수도 있다. ScrollIndicator 나 ScrollBar 컴포넌트는 이 리스트 컴포넌트에 부착되어서 스크롤관련 피드백을 전달해 주기도 한다. 이 컴포넌트는dataProvider 를 경유하게 되는데, dataProvider 는 다음과 같이 코드를 통해서 제공된다.

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

기본적으로 ScrollingList 는 ListItemRenderer 를 사용한다. 따라서 ListItemRenderer 이 FLA 파일 라이브러리에 있어야 작동한다. 그렇지 않으면 itemRenderer 점검가능 속성이 다른 컴포넌트로 교체된다. 자세한 설명은 점검 가능 속성을 참고하자.

2.4.4.1 사용자 상호작용

리스트 아이템이나 부착된 ScrollBar 인스턴스를 클릭하면 포커스를 ScrollingList로 이동시킨다. 포커스가 있으면 키보드의 상하 화살표를 누르거나 동급의 컨트롤을 사용해서 리스트 선택을 하나씩 스크롤 할 수 있다. 아무런 요소도 선택되지 않았다면 최상위 요소가 자동적으로 선택된 상태가 된다. 마우스 휠은 ScrollingList 경계의 최상위에 커서가 있는 경우 리스트를 스크롤 한다.

리스트 경계 내에서의 스크롤 작동은 ScrollingList의 *wrapping* 속성에 의해서 결정된다(이 속성은 점검가능 속성이 아니다). *wrapping*이 normal이면 선택상태가 리스트의 시작부분이나 끝부분일 경우에 포커스가 컴포넌트로 벗어나게 된다. 만약, *wrapping*이 wrap이면 시작부분과 끝부분에서 wrap 된다. *wrapping*이 stick인 경우에는 선택이 끝부분에 도착하면 멈추게 되고 인접한 컴포넌트로 포커스가 이동하지도 않는다.

[Page Up]과 [Page Down]키를 누르거나 동급의 컨트롤을 하게 되면 스크롤을 페이지 단위로 한다. [Home]과 [End]키, 혹은 동급의 컨트롤에 의해서 리스트를 시작요소와 마지막요소로 각각 스크롤시킬 수 있다. 부착된 ScrollBar 컴포넌트와의 상호작용은 ScrollingList에 영향을 미친다. 자세한 내용은 ScrollBar 항목을 참고하자.

개발자는 게임패드 컨트롤을 키보드와 마우스 컨트롤로 쉽게 맵핑할 수 있다. 예를 들면, 키보드의 방향키는 일반적으로 콘솔 게임용 D-패드에 맵핑할 수 있다. 이러한 맵핑은 CLIK에서 제작한 UI를 여러 플랫폼에서 작동할 수 있게 해준다.

2.4.4.2 컴포넌트 설정

ScrollingList는 명명된 부요소가 필요 없다. 하지만 스테이지에서 ScrollingList 컴포넌트 인스턴스를 위치시키고 크기 조절하려면 보이는 배경이 있는 것이 작업에 도움이 된다.

2.4.4.3 상태

ScrollingList 컴포넌트는 포커스와 disabled 속성에 기반한 3 가지 상태를 지원한다.

- **default** 혹은 enabled 상태
- **focused** 상태, 일반적으로 컴포넌트의 경계면에 하이라이트를 출력해서 상태를 나타낸다

- **disabled** 상태

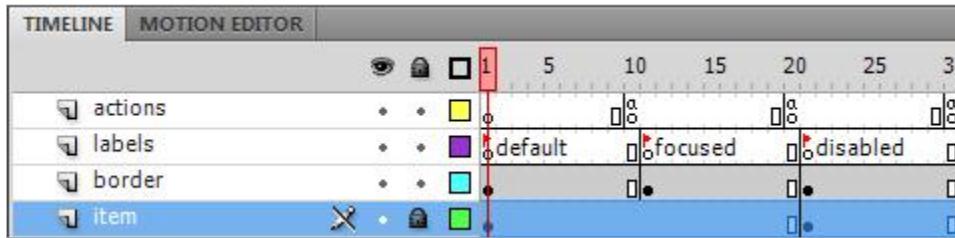
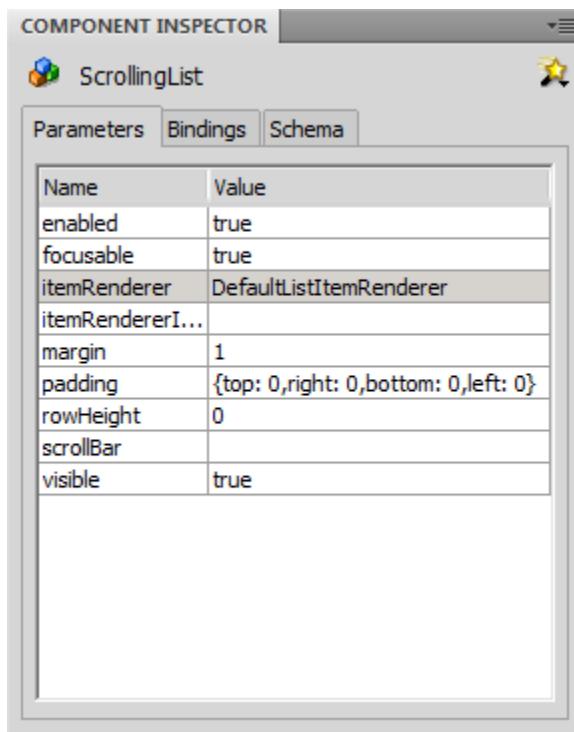


그림 40: ScrollingList 타임라인

2.4.4.4 점검가능 속성



ScrollingList로부터 유도된 MovieClip은 다음과 같은 속성을 갖는다.

Enabled	false로 설정하면 버튼을 비활성화합니다. 비활성화한 구성 요소는 더 이상 사용자 입력을 받지 않습니다.
Focusable	구성 요소에 대한 focus 관리를 활성화/비활성화합니다. focus 가능한 프로퍼티를 false로 설정하면 tab 키, 방향 키 및 마우스 기반 focus 변경에 대한 지원을 제거합니다.
itemRenderer	ListItemRenderer의 심볼 이름입니다. 목록 항목 인스턴스를

	내부에 생성하는 데 사용됩니다. <i>itemRendererInstanceName</i> 프로퍼티가 설정되면 영향이 없습니다(즉, 외부 <i>ListItemRenderer</i> 를 사용하는 경우).
itemRendererInstanceName	ScrollingList 컴포넌트와 함께 사용될 외부 리스트 아이템의 접두어(prefix). 스테이지의 리스트 아이템 인스턴스는 이 속성값으로 접두어가 붙는다. 이 값을 'r'로 설정하면 이 컴포넌트와 함께 사용되는 모든 리스트 아이템 인스턴스는 'r1', 'r2', 'r3',...값을 가져야 한다. 첫번째 아이템은 숫자 1을 가져야 한다.
Margin	리스트 컴포넌트와 내부적으로 생성된 리스트 아이템의 경계 여유값. 이 값은 <i>itemrendererInstanceName</i> 속성이 설정되면 효과가 없다.
padding	목록 항목에 대한 추가 패딩입니다. 이 값은 <i>itemRendererInstanceName</i> 프로퍼티가 설정되면 영향이 없습니다(즉, 외부 <i>ListItemRenderer</i> 를 사용하는 경우). 패딩은 자동으로 생성된 ScrollBar에 영향을 주지 않습니다.
rowHeight	내부에 생성된 목록 항목 인스턴스의 높이입니다. 이 값은 <i>itemRendererInstanceName</i> 프로퍼티가 설정되면 영향이 없습니다(즉, 외부 <i>ListItemRenderer</i> 를 사용하는 경우).
Visible	false로 설정하면 구성 요소를 숨깁니다. 부착된 ScrollBar 또는 다른 외부 <i>ListItemRenderer</i> 를 숨기지 않습니다.

2.4.4.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- **type**: 이벤트의 형식입니다.

- ***target***: 이벤트 대상입니다.
- ***currentTarget***: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- ***eventPhase***: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- ***bubbles***: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- ***cancelable***:
- 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

ScrollingList 컴포넌트에 의해서 생성되는 이벤트는 아래 표를 참고하라. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true 로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false 로 설정되었습니다.
ComponentEvent.STATE_CHANGE	구성 요소의 상태가 변경되었습니다.
FocusHandlerEvent.FOCUS_IN	구성 요소가 focus 를 받았습니다.
FocusHandlerEvent.FOCUS_OUT	구성 요소가 focus 를 잃었습니다.
ListEvent.INDEX_CHANGE	<p>선택된 인덱스가 변경되었습니다.</p> <ul style="list-style-type: none"> • <i>itemRenderer</i>: 두 번 클릭된 목록 항목입니다. IListItemRenderer 형식입니다. • <i>itemData</i>: 목록 항목과 연결된 데이터입니다. 이 값은 목록의dataProvider 에서 얻습니다. ActionScript 오브젝트 형식입니다. • <i>index</i>: 목록의 새로 선택된 인덱스입니다. int 형식입니다. 목록 항목 개수에서 1 을 뺀 -1 이 값입니다(selectedIndex 없음). • <i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러/마우스의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능).
ListEvent.ITEM_PRESS	목록 항목이 눌렸습니다.

- *itemRenderer*: 두 번 클릭된 목록 항목입니다.
IListItemRenderer 형식입니다.
- *itemData*: 목록 항목과 연결된 데이터입니다. 이
값은 목록의dataProvider에서 얻습니다.
ActionScript 오브젝트 형식입니다.
- *index*: 목록의 새로 선택된 인덱스입니다. int
형식입니다. 목록 항목 개수에서 1을 뺀 -1이
값입니다(selectedIndex 없음).
- *controllerIdx*: 이벤트를 생성하는 데 사용되는
컨트롤러/마우스의 인덱스입니다(멀티 마우스
커서 환경 용으로만 적용 가능).

ListEvent.ITEM_CLICK

목록 항목이 클릭되었습니다.

- *itemRenderer*: 두 번 클릭된 목록 항목입니다.
IListItemRenderer 형식입니다.
- *itemData*: 목록 항목과 연결된 데이터입니다. 이
값은 목록의dataProvider에서 얻습니다.
ActionScript 오브젝트 형식입니다.
- *index*: 목록의 새로 선택된 인덱스입니다. int
형식입니다. 목록 항목 개수에서 1을 뺀 -1이
값입니다(selectedIndex 없음).
- *controllerIdx*: 이벤트를 생성하는 데 사용되는
컨트롤러/마우스의 인덱스입니다(멀티 마우스
커서 환경 용으로만 적용 가능).

ListEvent.ITEM_DOUBLE_CLICK

목록 항목이 두 번 클릭되었습니다.

- *itemRenderer*: 두 번 클릭된 목록 항목입니다.
IListItemRenderer 형식입니다.
- *itemData*: 목록 항목과 연결된 데이터입니다. 이

	<p>값은 목록의 dataProvider에서 얻습니다. ActionScript 오브젝트 형식입니다.</p> <ul style="list-style-type: none"> • <i>index</i>: 목록의 새로 선택된 인덱스입니다. int 형식입니다. 목록 항목 개수에서 1을 뺀 -1이 값입니다(selectedIndex 없음). • <i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러/마우스의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능).
ListEvent.ITEM_ROLL_OVER	<p>마우스 커서가 목록 항목 위에 놓였습니다. <i>itemRenderer</i>: 두 번 클릭된 목록 항목입니다. IListItemRenderer 형식입니다.</p> <ul style="list-style-type: none"> • <i>itemData</i>: 목록 항목과 연결된 데이터입니다. 이 값은 목록의 dataProvider에서 얻습니다. ActionScript 오브젝트 형식입니다. • <i>index</i>: 목록의 새로 선택된 인덱스입니다. int 형식입니다. 목록 항목 개수에서 1을 뺀 -1이 값입니다(selectedIndex 없음). • <i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러/마우스의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능).
ListEvent.ITEM_ROLL_OUT	<p>The mouse cursor has rolled out of a list item.</p> <ul style="list-style-type: none"> • <i>itemRenderer</i>: 두 번 클릭된 목록 항목입니다. IListItemRenderer 형식입니다. • <i>itemData</i>: 목록 항목과 연결된 데이터입니다. 이 값은 목록의 dataProvider에서 얻습니다. ActionScript 오브젝트 형식입니다. • <i>index</i>: 목록의 새로 선택된 인덱스입니다. int

형식입니다. 목록 항목 개수에서 1 을 뺀 -1 이
값입니다(selectedIndex 없음).

- *controllerIdx*: 이벤트를 생성하는 데 사용되는
컨트롤러/마우스의 인덱스입니다(멀티 마우스
커서 환경 용으로만 적용 가능).

다음은 리스트 아이템 클릭을 대기하는 코드다.

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

2.4.5 TileList

TileList (scaleform.clik.controls.TileList)는 ScrollingList 와 비슷하게 내부요소를 스크롤 하는 리스트 컴포넌트다. 이 컴포넌트는 직접 리스트 아이템을 인스턴스화 할 수도 있고, 혹은 스테이지상에서 존재하는 리스트 아이템을 사용할 수도 있다. ScrollIndicator 나 ScrollBar 컴포넌트는 이 리스트 컴포넌트에 부착되어서 스크롤관련 피드백을 전달해 주기도 한다. TileList 와 ScrollingList 의 차이점은 TileList 가 다중 행과 열을 동시에 지원한다는 점이다. 리스트 아이템 선택은 모든 4 가지 방향으로 이동할 수 있다. 이 컴포넌트는dataProvider 를 경유하게 되는데, dataProvider 는 다음과 같이 코드를 통해서 제공된다.

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

기본적으로 TileList 는 ListItemRenderer 를 사용한다. 따라서 ListItemRenderer 이 FLA 파일 라이브러리에 있어야 작동한다. 그렇지 않으면 itemRenderer 점검가능 속성이 다른 컴포넌트로 교체된다. 자세한 설명은 점검가능 속성을 참고하자.

2.4.5.1 사용자 상호작용

리스트 아이템이나 부착된 ScrollBar 인스턴스를 클릭하면 포커스를 TileList로 이동시킨다. 포커스가 있으면 키보드의 상하 화살표를 누르거나 동급의 컨트롤을 사용해서 다중행 리스트 선택을 하나씩 횡방향으로 스크롤 할 수 있다. 또한, 키보드의 좌우 화살표를 누르거나 동급의 컨트롤을 사용해서 다중열 리스트 선택을 하나씩 종방향으로 스크롤 할 수 있다. 만약 TileList가 다중행렬을 가진 경우에는 4 가지 화살표키나 동급의 컨트롤을 사용해서 리스트 선택이 가능하다. 아무런 요소도 선택되지 않았다면 최상위 요소가 자동적으로 선택된 상태가 된다. 마우스 휠은 TileList 경계의 최상위에 커서가 있는 경우 리스트를 스크롤한다.

[Page Up]과 [Page Down]키를 누르거나 동급의 컨트롤을 하게 되면 스크롤을 페이지 단위로 한다. [Home]과 [End]키, 혹은 동급의 컨트롤에 의해서 리스트를 시작요소와 마지막 요소로 각각 스크롤시킬 수 있다. 부착된 ScrollBar 컴포넌트와의 상호작용은 ScrollingList에 영향을 미친다. 자세한 내용은 ScrollBar 항목을 참고하자.

2.4.5.2 컴포넌트 설정

TileList는 명명된 부요소가 필요없다. 하지만 스테이지에서 TileList 컴포넌트 인스턴스를 위치시키고 크기 조절하려면 보이는 배경이 있는 것이 작업에 도움이 된다.

2.4.5.3 상태

TileList 컴포넌트는 포커스와 disabled 속성에 기반한 3 가지 상태를 지원한다.

- **default** 혹은 enabled 상태
- **focused** 상태, 일반적으로 컴포넌트의 경계면에 하이라이트를 출력해서 상태를 나타낸다
- **disabled** 상태

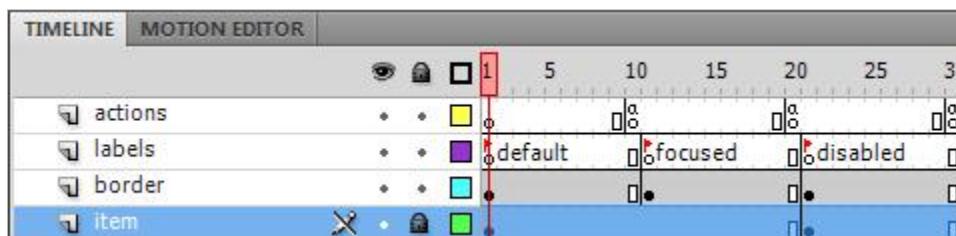
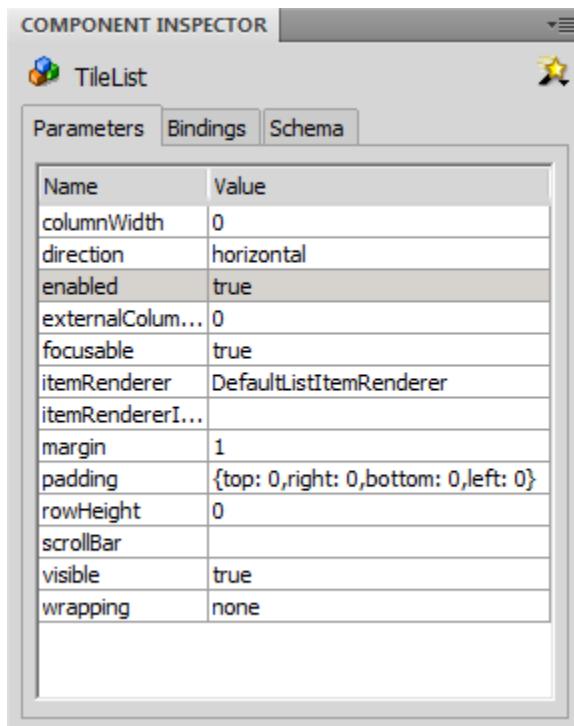


그림 41: TileList 타임라인

2.4.5.4 점검가능 속성



TileList로부터 유도된 MovieClip은 다음과 같은 속성을 갖는다.

columnWidth	내부적으로 생성된 리스트 아이템 인스턴스의 폭. <i>rendererInstanceName</i> 속성이 설정되면 효과가 없다.
Direction	스크롤 방향입니다. 열과 행의 의미는 이 값에 따라 변경되지 않습니다.
enabled	Disables the component if set to false. Disabled components will no longer receive user input.
externalColumnCount	<i>rendererInstanceName</i> 속성이 설정되면 이 값이 외부 렌더러에서 사용되는 컬럼 개수의 TileList를 통지하기 위해서 사용된다.
focusable	Enable/disable focus management for the component. Setting the focusable property to false will remove support for tab key, direction key and mouse based focus changes.
itemRenderer	ListItemRenderer의 심볼 이름입니다. 목록 항목 인스턴스를 내부에 생성하는 데 사용됩니다.

	<i>itemRendererInstanceName</i> 프로퍼티가 설정되면 영향이 없습니다(즉, 외부 ListItemRenderer 를 사용하는 경우).
itemRendererInstanceName	이 ScrollingList 구성 요소와 함께 사용하는 외부 목록 항목 렌더러의 접두사입니다. 스테이지의 목록 항목 인스턴스는 이 프로퍼티 값과 함께 접두사가 붙어야 합니다. 이 프로퍼티의 값이 'r'로 설정되면 이 구성 요소와 함께 사용되는 모든 목록 항목 인스턴스는 다음 값을 가져야 합니다. 'r1', 'r2', 'r3',... 첫 번째 항목에는 숫자 1 이 있어야 합니다.
Margin	목록 구성 요소와 내부에 생성된 목록 항목 경계 사이의 여백입니다. 이 값은 <i>itemRendererInstanceName</i> 프로퍼티가 설정되면 영향이 없습니다(즉, 외부 ListItemRenderer 를 사용하는 경우).
Padding	목록 항목에 대한 추가 패딩입니다. 이 값은 <i>itemRendererInstanceName</i> 프로퍼티가 설정되면 영향이 없습니다(즉, 외부 ListItemRenderer 를 사용하는 경우). 패딩은 자동으로 생성된 ScrollBar 에 영향을 주지 않습니다.
rowHeight	내부에 생성된 목록 항목 인스턴스의 높이입니다. 이 값은 <i>itemRendererInstanceName</i> 프로퍼티가 설정되면 영향이 없습니다(즉, 외부 ListItemRenderer 를 사용하는 경우).
Visible	false 로 설정하면 구성 요소를 숨깁니다. 부착된 ScrollBar 또는 다른 외부 ListItemRenderer 를 숨기지 않습니다.

2.4.5.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- **type:** 이벤트의 형식입니다.

- ***target***: 이벤트 대상입니다.
- ***currentTarget***: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- ***eventPhase***: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- ***bubbles***: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- ***cancelable***: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

TileList 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	구성 요소의 보이는 프로퍼티가 실행 시 true 로 설정되었습니다.
ComponentEvent.HIDE	구성 요소의 보이는 프로퍼티가 실행 시 false 로 설정되었습니다.
ComponentEvent.STATE_CHANGE	구성 요소의 상태가 변경되었습니다.
FocusHandlerEvent.FOCUS_IN	구성 요소가 focus 를 받았습니다.
FocusHandlerEvent.FOCUS_OUT	The component has lost focus.
ListEvent.INDEX_CHANGE	<p>선택된 인덱스가 변경되었습니다.</p> <ul style="list-style-type: none"> • <i>itemRenderer</i>: 두 번 클릭된 목록 항목입니다. IListItemRenderer 형식입니다. • <i>itemData</i>: 목록 항목과 연결된 데이터입니다. 이 값은 목록의dataProvider 에서 얻습니다. ActionScript 오브젝트 형식입니다. • <i>index</i>: 목록의 새로 선택된 인덱스입니다. int 형식입니다. 목록 항목 개수에서 1 을 뺀 -1 이 값입니다(selectedIndex 없음). • <i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러/마우스의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능).

ListEvent.ITEM_PRESS	<p>목록 항목이 눌렸습니다.</p> <ul style="list-style-type: none"> • <i>itemRenderer</i>: 두 번 클릭된 목록 항목입니다. IListItemRenderer 형식입니다. • <i>itemData</i>: 목록 항목과 연결된 데이터입니다. 이 값은 목록의 dataProvider에서 얻습니다. ActionScript 오브젝트 형식입니다. • <i>index</i>: 목록의 새로 선택된 인덱스입니다. int 형식입니다. 목록 항목 개수에서 1을 뺀 -1이 값입니다(selectedIndex 없음). • <i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러/마우스의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능).
ListEvent.ITEM_CLICK	<p>목록 항목이 클릭되었습니다.</p> <ul style="list-style-type: none"> • <i>itemRenderer</i>: 두 번 클릭된 목록 항목입니다. IListItemRenderer 형식입니다. • <i>itemData</i>: 목록 항목과 연결된 데이터입니다. 이 값은 목록의 dataProvider에서 얻습니다. ActionScript 오브젝트 형식입니다. • <i>index</i>: 목록의 새로 선택된 인덱스입니다. int 형식입니다. 목록 항목 개수에서 1을 뺀 -1이 값입니다(selectedIndex 없음). • <i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러/마우스의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능).
ListEvent.ITEM_DOUBLE_CLICK	<p>목록 항목이 두 번 클릭되었습니다.</p> <ul style="list-style-type: none"> • <i>itemRenderer</i>: 두 번 클릭된 목록 항목입니다. IListItemRenderer 형식입니다.

- *itemData*: 목록 항목과 연결된 데이터입니다. 이 값은 목록의 dataProvider에서 얻습니다. ActionScript 오브젝트 형식입니다.
- *index*: 목록의 새로 선택된 인덱스입니다. int 형식입니다. 목록 항목 개수에서 1을 뺀 -1이 값입니다(selectedIndex 없음).
- *controllerIdx*: 이벤트를 생성하는 데 사용되는 컨트롤러/마우스의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능).

ListEvent.ITEM_ROLL_OVER

마우스 커서가 목록 항목 위에 놓였습니다.

- *itemRenderer*: 두 번 클릭된 목록 항목입니다. IListItemRenderer 형식입니다.
- *itemData*: 목록 항목과 연결된 데이터입니다. 이 값은 목록의 dataProvider에서 얻습니다. ActionScript 오브젝트 형식입니다.
- *index*: 목록의 새로 선택된 인덱스입니다. int 형식입니다. 목록 항목 개수에서 1을 뺀 -1이 값입니다(selectedIndex 없음).
- *controllerIdx*: 이벤트를 생성하는 데 사용되는 컨트롤러/마우스의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능).

ListEvent.ITEM_ROLL_OUT

마우스 커서가 목록 항목 밖으로 벗어났습니다.

- *itemRenderer*: 두 번 클릭된 목록 항목입니다. IListItemRenderer 형식입니다.
- *itemData*: 목록 항목과 연결된 데이터입니다. 이 값은 목록의 dataProvider에서 얻습니다.

ActionScript 오브젝트 형식입니다.

- *index*: 목록의 새로 선택된 인덱스입니다. int 형식입니다. 목록 항목 개수에서 1 을 뺀 -1 이 값입니다(selectedIndex 없음).
- *controllerIdx*: 이벤트를 생성하는 데 사용되는 컨트롤러/마우스의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능).

다음 코드는 TileList 가 포커스를 받았는지를 판단할 때 사용된다.

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

2.4.6 DropdownMenu

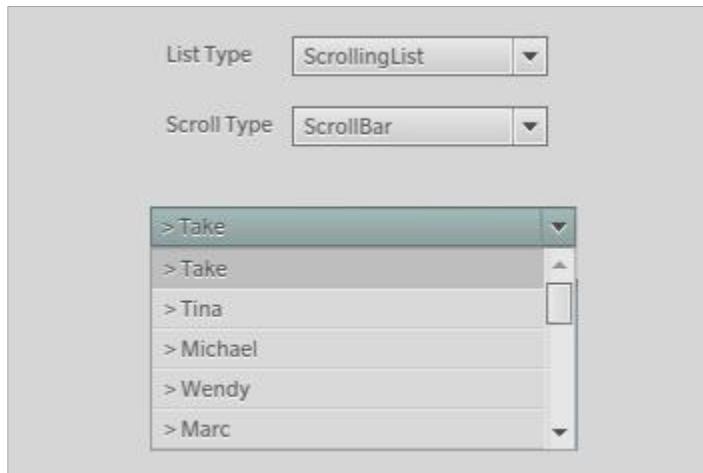


그림 42: 스키 없는 DropdownMenu.

DropdownMenu (scaleform.clik.controls.DropdownMenu)는 버튼과 리스트의 작동을 합친 컴포넌트다. 이 컴포넌트를 클릭하면 선택될 요소의 리스트를 연다. DropdownMenu 는 아이들 상태일때(idle state) 선택된 요소만 보여준다. ScrollingList 나 TileList 를 사용해서 설정가능하며, ScrollBar 나

ScrollIndicator 를 조합할 수도 있다. DropdownMenu 의 리스트 요소는 dataProvider 를 경유하게 되는데, dataProvider 는 다음과 같이 코드를 통해서 제공된다.

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

DropdownMenu 는 기본적으로 ScrollingList 컴포넌트를 사용해서 내용물을 관리한다. 따라서 ListItemRenderer 이 FLA 파일 라이브러리에 있어야 작동한다. 그렇지 않으면 dropdown 점검가능 속성이 다른 컴포넌트로 교체된다. 자세한 설명은 점검가능 속성을 참고하자.

또한, 기본적으로 DropdownMenu 는 리스트 요소에 대해서 scrollbar 를 붙이지 않는다. Scrollbar 나 ScrollIndicator 는 코드를 통해서 부착된다. 팁과 트릭 항목을 참고하자.

2.4.6.1 사용자 상호작용

DropdownMenu 인스턴스를 클릭하거나 [Enter]키를 누르거나 혹은 동등한 컨트롤을 통해서 선택가능한 요소 리스트를 열수 있다. 리스트가 열리면 포커스도 이동된다. 사용자는 ScrollingList, TileList, ScrollBar 에서 설명한 사용자 상호작용을 통해서 작업할 수 있다. 리스트 아이템을 클릭하면 선택하게 되고, 리스트를 닫고 선택된 아이템을 DropdownMenu 에 보여준다. 리스트 경계바깥을 클릭하면 자동적으로 리스트를 닫고 포커스가 DropdownMenu 컴포넌트로 복귀된다.

2.4.6.2 컴포넌트 설정

DropdownMenu 는 대부분의 기능을 Button 컴포넌트로부터 상속받는다. 결과적으로 DropdownMenu 클래스를 사용하는 MovieClip 은 다음과 같이 명명된 부요소를 가져야 한다. 리스트와 스크롤바는 동적으로 생성된다.

- **textField**. (optional) TextField 타입. 버튼 레이블
- **focusIndicator**. (optional) MovieClip 형. 분리된 MovieClip 이 포커스된 상태 출력에 사용된다. 실제 존재하는 경우에는 MovieClip 이 show 와 hide 라 명명된 프레임을 갖고 있어야 한다.

2.4.6.3 상태

DropdownMenu 는 열렸을 경우에 토클가능하다. 따라서 ToggleButton 이나 CheckBox 처럼 선택상태를 나타낼 수 있어야 한다.

- **up** 혹은 기본 상태
- **over** 상태는 마우스 커서가 위에 있을 때, 혹은 포커스를 가졌을 때
- **down** 상태는 버튼이 눌렸을 때
- **disabled** 상태
- **selected_up** 혹은 기본상태
- **selected_over** 상태는 마우스 커서가 위에 있을 때, 혹은 포커스를 가졌을 때.
- **selected_down** 상태는 버튼이 눌렸을 때
- **selected_disabled** 상태

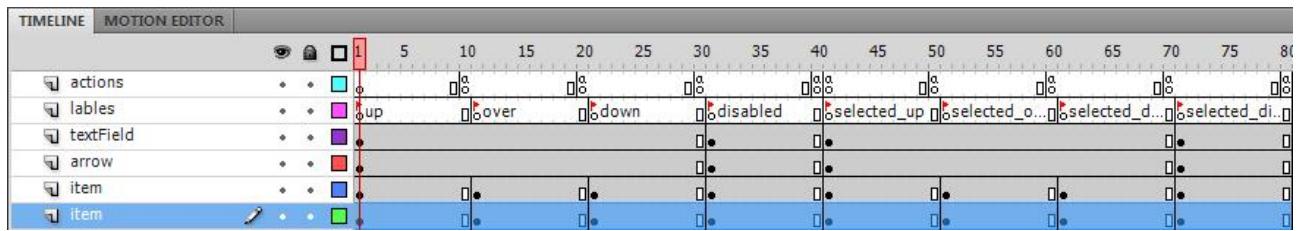
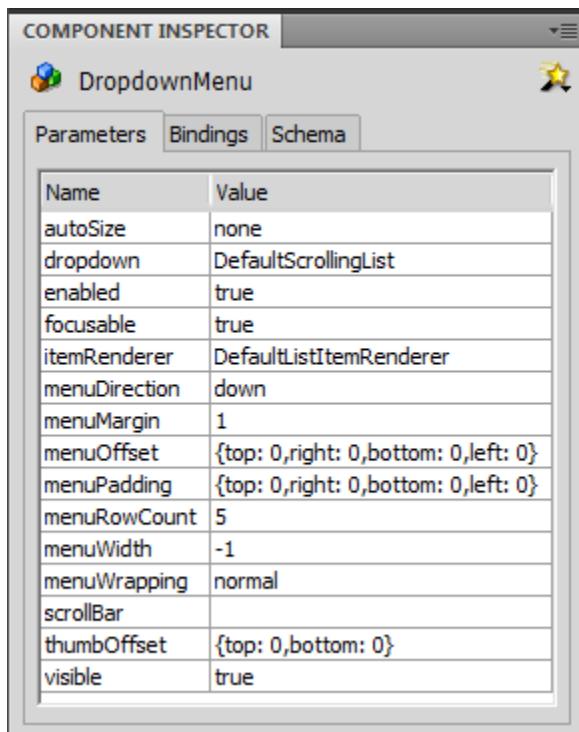


그림 43: DropdownMenu 타임라인

이들은 DropdownMenu에서 필요한 최소한의 키프레임이다. Button 컴포넌트(결과적으로 DropdownMenu 컴포넌트도 포함)에서 지원되는 확장 상태 집합과 키프레임은 [Getting Started with CLIK Buttons](#) 문서를 참고하라.

2.4.6.4 점검가능 속성



DropdownMenu 컴포넌트의 점검가능 속성은 다음과 같다.

autoSize	포함된 텍스트에 맞춰서 닫은 DropdownMenu 크기가 조절되고, 크기가 조절된 버튼을 정렬할 방향을 결정합니다. autoSize 프로퍼티를 autoSize="none"으로 설정하면 현재 크기를 변경하지 않고 그대로 놔둡니다.
Dropdown	DropdownMenu 컴포넌트에서 사용될 리스트 컴포넌트((ScrollingList 나 TileList)의 심볼명
Enabled	false 로 설정하면 구성 요소를 비활성화합니다.
Focusable	기본적으로 구성 요소는 사용자 상호 작용을 위해 focus 를 받을 수 있습니다. 이 프로퍼티를 false 로 설정하면 focus 를 획득하지 못합니다.
menuDirection	드롭다운 목록이 열릴 방향입니다. 유효한 값은 "up"과 "down"입니다.

menuMargin	내부적으로 만들어진 아이템들의 리스트와 컴퍼넌트 리스트의 경계선과의 여백. 이 여백은 자동적으로 발생된 스크롤바에 영향을 줌.
menuOffset	드롭다운 버튼 위치에서 드롭다운 목록의 수평 및 수직 오프셋입니다. 양수(+)인 수평 값은 목록을 드롭다운 버튼 수평 위치의 오른쪽으로 옮깁니다. 양수(+)인 수직 값은 목록을 버튼에서 옮깁니다.
menuPadding	목록 항목에 대한 상단, 하단, 왼쪽 및 오른쪽 추가 패딩입니다. 자동으로 생성된 scrollbar에 영향을 주지 않습니다.
menuRowCount	목록이 표시할 열의 개수입니다.
menuWidth	If set, this number will be enforced as the width of the drop down's list.
thumbOffset	Scrollbar 썸 상단 및 하단 오프셋입니다. 이 프로퍼티는 목록이 자동으로 scrollbar 인스턴스를 생성하지 않으면 영향이 없습니다.
Scrollbar	드롭다운 목록 스크롤 바의 심볼 이름입니다. 드롭다운 목록 인스턴스에 의해 생성됩니다. 값을 비우면 드롭다운 목록에 스크롤 바가 생기지 않습니다.
Visible	false로 설정하면 구성 요소를 숨깁니다

2.4.6.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- **type**: 이벤트의 형식입니다.
- **target**: 이벤트 대상입니다.
- **currentTarget**: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- **eventPhase**: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).

- **bubbles**: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- **cancelable**: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

DropdownMenu 컴포넌트에 의해서 생성되는 이벤트는 change 이벤트를 제외한 Button 컴포넌트와 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true 로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false 로 설정되었습니다.
ComponentEvent.STATE_CHANGE	버튼의 상태가 변경되었습니다.
Event.SELECT	선택된 프로퍼티가 변경되었습니다.
FocusHandlerEvent.FOCUS_IN	버튼이 focus 를 받았습니다.
FocusHandlerEvent.FOCUS_OUT	버튼이 focus 를 잃었습니다.
ListEvent.INDEX_CHANGE	선택된 인덱스가 변경되었습니다. <ul style="list-style-type: none"> • <i>itemRenderer</i>: 두 번 클릭된 목록 항목입니다. IListItemRenderer 형식입니다. • <i>itemData</i>: 목록 항목과 연결된 데이터입니다. 이 값은 목록의dataProvider 에서 얻습니다. ActionScript 오브젝트 형식입니다. • <i>index</i>: 목록의 새로 선택된 인덱스입니다. int 형식입니다. 목록 항목 개수에서 1 을 뺀 -1 이 값입니다(selectedIndex 없음). • <i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러/마우스의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능).
MouseEvent.ROLL_OVER	마우스 커서가 구성 요소 위에 놓였습니다. <i>mouseIdx</i> : 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로, 이벤트를 MouseEventEx 로 보내야 합니다.

	<p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로, 이벤트를 MouseEventEx로 보내야 합니다.</p>
MouseEvent.ROLL_OUT	<p>마우스 커서가 구성 요소 밖으로 벗어났습니다.</p> <p><i>mouseIdx</i>: uint를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). 숫자 형식입니다. Scaleform 전용으로, 이벤트를 MouseEventEx로 보내야 합니다.</p> <p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지 나타냅니다(zero base index). Scaleform 전용으로, 이벤트를 MouseEventEx로 보내야 합니다.</p>
ButtonEvent.PRESS	<p>DropdownMenu가 눌렸습니다.</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p> <p><i>isKeyboard</i>: 이벤트가 키보드/게임패드에 의해 생성되었으면 true이고 이벤트가 마우스에 의해 생성되었으면 false입니다.</p> <p><i>isRepeat</i>: 이벤트가 autoRepeating 버튼을 길게 눌러 생성되었으면 true이고 버튼이 현재 반복 중이 아니면 false입니다.</p>
MouseEvent.DOUBLE_CLICK	<p>구성 요소가 두 번 클릭되었습니다. doubleClickEnabled 프로퍼티가 true인 경우에만 일어납니다.</p> <p><i>mouseIdx</i>: 이벤트를 생성하는 데 사용되는 마우스 커서의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다. Scaleform 전용으로, 이벤트를 MouseEventEx로 보내야 합니다.</p> <p><i>buttonIdx</i>: 어느 버튼에 대해 이벤트가 생성되는지</p>

	<p>나타냅니다(zero base index). Scaleform 전용으로, 이벤트를 MouseEventEx로 보내야 합니다.</p>
ButtonEvent.CLICK	<p>DropdownMenu가 클릭되었습니다.</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p> <p><i>isKeyboard</i>: 이벤트가 키보드/게임패드에 의해 생성되었으면 true이고 이벤트가 마우스에 의해 생성되었으면 false입니다.</p> <p><i>isRepeat</i>: 이벤트가 autoRepeating 버튼을 길게 눌러 생성되었으면 true이고 버튼이 현재 반복 중이 아니면 false입니다.</p>
ButtonEvent.DRAG_OVER	<p>마우스 커서가 DropdownMenu 위로 드래그되었습니다(왼쪽 마우스 버튼이 눌려 있는 상태).</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p>
ButtonEvent.DRAG_OUT	<p>마우스 커서가 DropdownMenu 밖으로 드래그되었습니다(왼쪽 마우스 버튼이 눌려 있는 상태).</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p>
ButtonEvent.RELEASE_OUTSIDE	<p>마우스 커서가 버튼 밖으로 드래그되고 왼쪽 마우스 버튼이 해제되었습니다.</p> <p><i>controllerIdx</i>: 이벤트를 생성하는 데 사용되는 컨트롤러의 인덱스입니다(멀티 마우스 커서 환경 용으로만 적용 가능). uint 형식입니다.</p>

2.5 프로그래스 타입(*Progress Types*)

프로그래스 타입은 상태나 이벤트 혹은 액션의 진행을 출력하는데 사용된다. 스케일폼 CLIK 프레임워크는 이 범주에 해당하는 2 개의 컴포넌트를 제공한다: StatusIndicator 와 Progress Bar. StatusIndicator 는 이벤트나 액션의 상태를 출력하는데 사용되고, ProgressBar 는 StatusIndicator 와 동일한 의미지만, 다른 컴포넌트나 진행상황 따라 생성된 이벤트에 기반한 추가적인 기능을 제공한다.

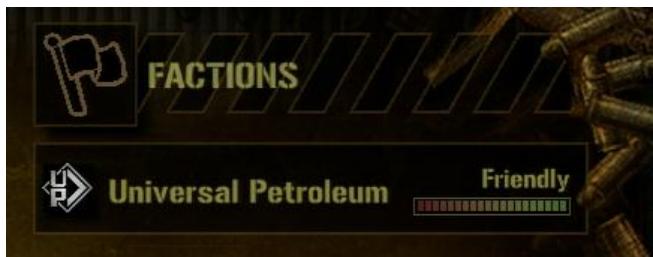


그림 44: *Mercenaries 2*의 팩션 StatusIndicator 사용예

2.5.1 StatusIndicator

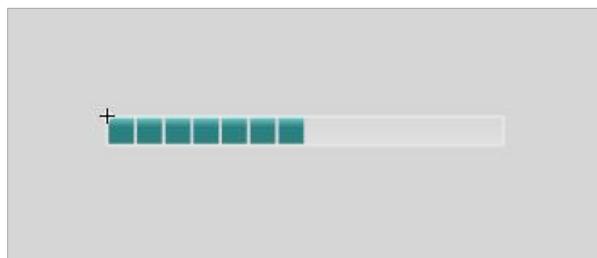


그림 45: 스킨없는 StatusIndicator.

StatusIndicator (`scaleform.clik.controls.StatusIndicator`) 컴포넌트는 이벤트나 액션의 상태를 타임라인을 사용해서 비쥬얼 지시자(indicator)형태로 보여준다. StatusIndicator 의 값은 최대값과 최소값을 보간해서 컴포넌트 타임라인상의 프레임 번호를 생성한다. 컴포넌트의 타임라인이 상태를 출력하기 위해서 사용되기 때문에 완벽하게 자유롭게 혁신적인 비쥬얼 지시자를 만들 수 있을 것이다.

2.5.1.1 사용자 상호작용

StatusIndicator 는 상호작용을 지원하지 않는다.

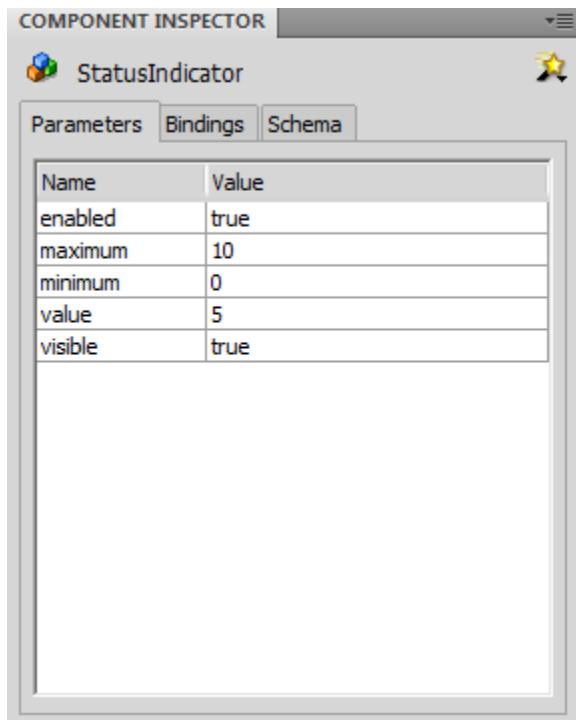
2.5.1.2 컴포넌트 설정

StatusIndicator 를 사용하는 MovieClip 은 명명된 부요소가 필요 없다. 대신, StatusIndicator 는 적어도 2 개의 프레임이 있어야 제대로 작동한다. stop() 명령을 첫 번째 프레임에 넣어서 프레임 재생을 막아야 함을 명심하라. StatusIndicator 는 value 속성에 의해 생성된 프레임으로 gotoAndStop() 을 사용해서 이동한다.

2.5.1.3 상태

StatusIndicator 컴포넌트는 상태가 없다. 컴포넌트의 프레임이 이벤트나 액션의 상태를 출력하는데 사용된다.

2.5.1.4 점검 가능 속성



StatusIndicator 로부터 파생된 MovieClip 은 다음과 같은 속성을 갖는다.

Visible	false 인 경우 컴포넌트를 숨긴다.
Enabled	true 인 경우 컴포넌트를 disable 한다.
Value	이벤트나 액션의 상태값. 최대값과 최소값을 보간해서 재생 될 프레임번호를 생성한다.

Minimum	타겟 프레임을 보간할 때 사용될 최소값
Maximum	타겟 프레임을 보간할 때 사용될 최대값

2.5.1.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- ***type***: 이벤트의 형식입니다.
- ***target***: 이벤트 대상입니다.
- ***currentTarget***: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- ***eventPhase***: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- ***bubbles***: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- ***cancelable***: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

StatusIndicator 는 특별한 이벤트를 발생시키지 않는다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true 로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false 로 설정되었습니다.

2.6 기타 다른 타입(Other Types)

스케일폼 CLIK 프레임워크는 특정 범주에 넣기 쉽지 않은 몇가지 컴포넌트들이 있다. 창 구성 요소로 콘텐츠를 표시하는 데 사용되거나 대화 상자로 사용될 수 있는 modal 및 modal-free 창을 표시할 수 있습니다.

DragSlot 은 드래그 앤 드롭 기능을 필요로 하는 UI에서 사용되도록 설계된 CLIK 구성 요소의 기본 구현입니다. 드래그 앤 드롭은 CLIK 내에서 사용자 정의 프레임워크로 구현되고 CLIK DragManager 의 인스턴스화가 필요합니다(scaleform.clik.managers.DragManager.as 참조).

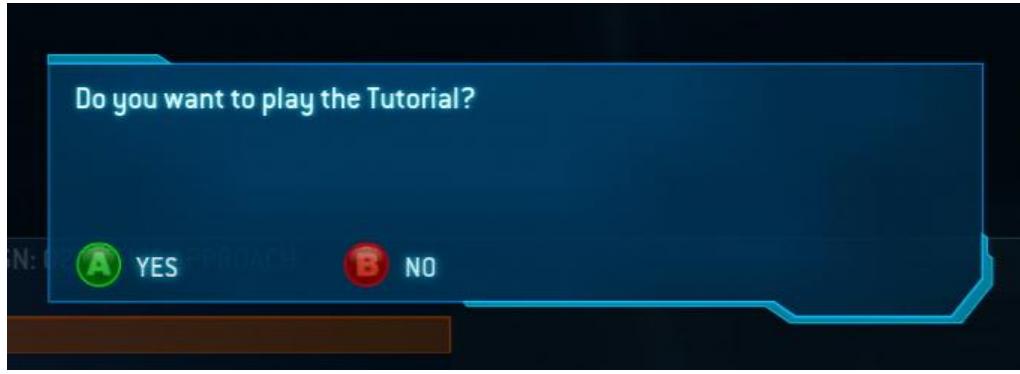


Figure 46: Window example from *Halo Wars*.

2.6.1 Window

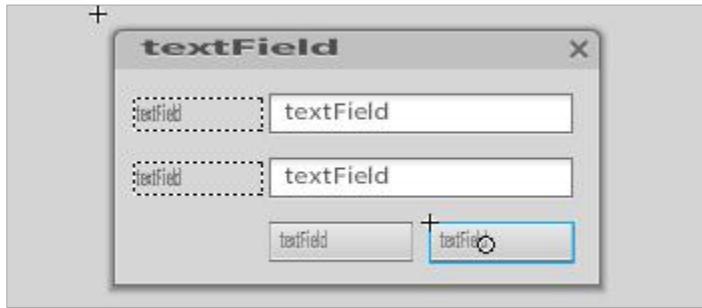


Figure 47: Sample unskinned Window.

CLIK 창 구성 요소(scaleform.clik.controls.Window)는 경고 대화 상자와 같은 창 또는 대화 상자 보기 를 표시합니다. 이 컨텐츠는 완전히 사용자가 정의하는 것이기 때문에 프리빌트 컴포넌트는 아무런 컨텐츠를 가질 수 없다. 하지만, 컴포넌트 심볼을 편집하면 간단하게 컨텐츠를 등록할 수 있다. PopUpManager 는 창을 대화 상자로 여는 기능(PopUpManager.showModal() 참조) 및 다중 대화 상자 동시 관리를 제공합니다.

2.6.1.1 사용자 상호작용

Window 의 사용자 상호작용은 생성하는 대화상자 내에서 정의된다.

2.6.1.2 컴포넌트 설정

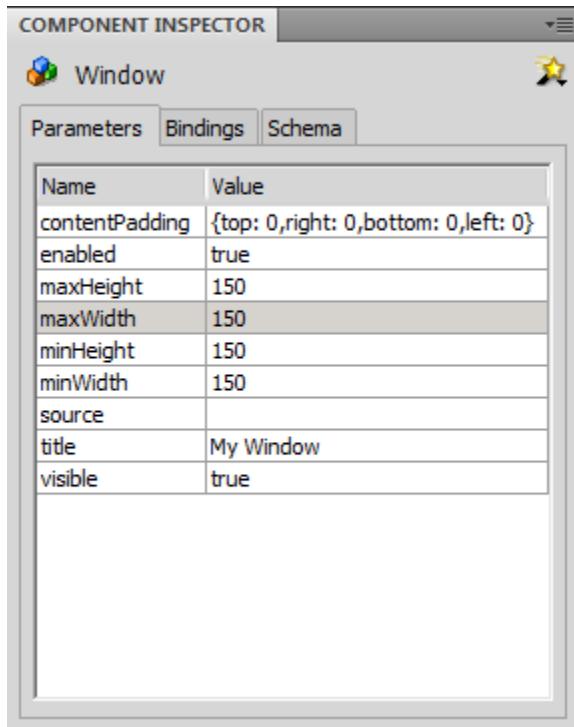
Window 클래스를 사용하는 MovieClip 은 다음과 같이 명명된 부요소가 필요하다.

- **closeBtn**: (옵션) CLIK 버튼 형식입니다. 창을 닫는 버튼입니다.
- **titleBtn**: (옵션) CLIK 버튼 형식입니다. 드래그할 수 있는 대화 상자용 제목 표시줄입니다.
- **okBtn**: (옵션) CLIK 버튼 형식입니다. 클릭하면 창이 닫히는 "Accept" 또는 "OK" 버튼입니다.
- **resizeBtn**: (옵션) CLIK 버튼 형식입니다. 사용자가 창을 누르고 드래그할 때 창 크기를 변경할 수 있게 하는 버튼입니다.
- **background**: (옵션) MovieClip 형식입니다. 창의 배경입니다. 창 크기를 변경하면 크기가 조정 및 변경됩니다.
- **hit**: (옵션) MovieClip 버튼 형식입니다. 창의 hitArea입니다.

2.6.1.3 상태

Dialog 컴포넌트는 상태가 없다. 대화상자 뷰에 출력되는 MovieClip은 자체 상태를 가질 수도 있고, 없을 수도 있다.

2.6.1.4 점검가능 속성



Window 컴포넌트로부터 파생된 MovieClip은 다음과 같은 점검가능 속성을 갖는다.

contentPadding	창으로 불러온 콘텐츠에 적용될 상단, 하단, 왼쪽 및 오른쪽 패딩입니다.
-----------------------	--

Enabled	false로 설정하면 버튼을 비활성화합니다.
maxHeight	크기 변경 버튼을 통해 크기를 변경할 때 창의 최대 높이입니다.
maxWidth	크기 변경 버튼을 통해 크기를 변경할 때 창의 최대 넓이입니다.
minHeight	크기 변경 버튼을 통해 크기를 변경할 때 창의 최소 높이입니다.
minWidth	크기 변경 버튼을 통해 크기를 변경할 때 창의 최소 넓이입니다.
Source	창으로 불러올 심볼의 보내기 이름입니다.
Title	창에 titleBtn 하위 구성 요소가 있는 경우, 이 문자열은 라벨로 사용됩니다.
Visible	false로 설정하면 구성 요소를 숨깁니다.

2.6.1.5 이벤트

모든 이벤트 콜백은 단일 이벤트 또는 이벤트에 대한 관련 정보가 포함된 이벤트 하위 클래스 파라미터를 받습니다. 다음 프로퍼티는 모든 이벤트에 공통입니다.

- **type**: 이벤트의 형식입니다.
- **target**: 이벤트 대상입니다.
- **currentTarget**: 이벤트 리스너로 Event 오브젝트를 처리하고 있는 오브젝트입니다.
- **eventPhase**: 이벤트 흐름에서 현재 단계입니다. (EventPhase.CAPTURING_PHASE, EventPhase.AT_TARGET, EventPhase.BUBBLING_PHASE).
- **bubbles**: 이벤트가 버블링 이벤트인지 여부를 나타냅니다.
- **cancelable**: 이벤트와 연관된 동작을 차단할 수 있는지 여부를 나타냅니다.

Window 컴포넌트에 의해서 생성되는 이벤트는 다음과 같다. 이벤트 다음에 나열된 속성들은 공통속성 외에 추가적으로 제공되는 것들이다.

ComponentEvent.SHOW	보이는 프로퍼티가 실행 시 true로 설정되었습니다.
ComponentEvent.HIDE	보이는 프로퍼티가 실행 시 false로 설정되었습니다.

대화상자의 제출(submit) 이벤트 다루는 법

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

3 아트 세부설명

이번 장에서는 아티스트가 스케일폼 CLIK 컴포넌트용 스킨을 개발하는 방법에 대하여 도와줄 것이다. 여기서는 컴포넌트에 스킨을 입히고, 애니메이션과 폰트를 내장하는 작은 샘플을 통해서 연습을하게 될 것이다.

3.1 가장 좋은 연습

이번 장은 스케일폼 CLIK 컴포넌트에 스킨을 입히는 가장 좋은 연습예제를 포함하고 있다.

3.1.1 픽셀퍼펙트 이미지(Pixel-perfect Images)

CLIK 컴포넌트에 벡터기반 스킨을 개발 할 때는 모든 애셋들이 픽셀퍼펙트 할 것을 강추한다. 픽셀퍼펙트란 애셋의 라인들이 플래시 그리드에 완벽하게 일치하는 것을 말한다.

그리드를 켜고 설정하는 법

1. 상단 플래시 메뉴에서 View 선택
2. Grid → Show Grid 선택
3. 상단 플래시 메뉴에서 View 다시 선택
4. Grid → Edit Grid 선택
5. 수평, 수직값에 1px 입력
6. OK 누름

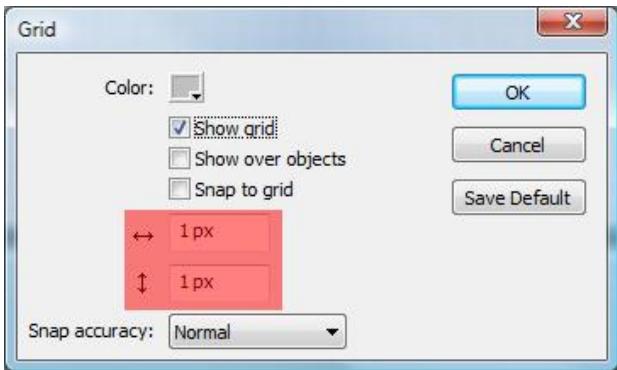


그림 48: Edit Grid 창.

이제 스테이지에 그리드가 보일 것이다. 각각의 그리드 상자는 정확히 1 픽셀을 나타낸다. 아트 애셋을 만들 때 그리드에 일치하도록 하라. 이렇게 하면 최종 SWF 가 뭉개지지(blur) 않을 것이다. 주의: 모든 이미지의 크기가 2 의 제곱이어야 한다. 그렇지 않으면 이미지가 뭉개질 것이다. 자세한 내용은 다음에 나오는 “2 의 제곱” 항목을 참고하라.

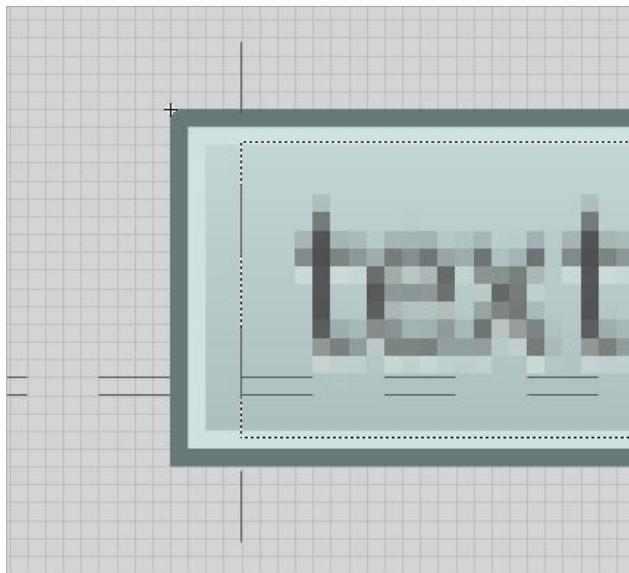


그림 49: 픽셀퍼펙트 벡터 그래픽 스킨

3.1.2 마스크(Masks)

플래시에서 마스크는 아티스트에게 실행시 그래픽의 일부분을 숨길 수 있게 해준다. 마스크는 애니메이션 효과에 가끔 사용된다. 하지만, 실제 마스크는 실행 시에 상당한 비용이 필요하다. 따라서, 스케일품에서는 가능하다면 마스크 사용을 피하도록 권고한다. 꼭 마스크를 사용해야 한다면 단일 디지트(single digit)로 묶어두고서 마스크가 있는 것과 없는 것의 성능테스트를 해보길 바란다.

플래시에서 사용 가능한 선택사항으로는 포토샵에서 알파블렌드로 마스크 될 영역이 있는 PNG 파일을 만드는 것이다. 하지만, 이 방법은 투명영역에 애니메이션이 없을 경우에만 작동한다.

3.1.3 애니메이션(Animations)

하나의 벡터 형태가 다른 형태로 몰프되는 애니메이션은 피하는 것이 최고다. 즉 사각형이 원으로 몰프되는 형태 말이다. 이러한 타입의 애니메이션은 처리 비용이 매우 비싸며, 모든 형태가 매 프레임마다 재계산 되어야 하기 때문이다.

가능하다면 벡터 그래픽에서도 스케일 애니메이션을 피하는 것이 좋은데 이 것은 추가 tessellation(벡터의 모양을 삼각형으로 변환해주는 작업)때문에 수행성능에 마이너스 영향을 미치기 때문이다. 가장 저렴한 애니메이션은 이러한 추가 tessellation 을 필요로 하지 않는 이동과 회전이다.

프로그램적인 tween 과 타임라인 기반 tween 선택은 피하는 것이 좋은데 이는 성능면에서 타임라인 tween 이 훨씬 저렴하기 때문이다. 타임라인 tween 은 원하는 프레임 레이트에서 부드러운 애니메이션이 될 수 있는 최소한의 키프레임 수를 유지하도록 하라.

3.1.4 레이어와 그리기 기본요소(Layers and Draw Primitives)

플래시에서 스킨을 만들 때는 가능한한 적은 레이어를 사용하라. 사용되는 모든 레이어는 적어도 하나의 그리기 기본요소를 추가한다. 그리기 기본요소가 많아질수록 더 많은 메모리가 필요하기 때문에 수행성능도 떨어지게 된다.

3.1.5 복잡한 스킨

복잡한 스킨에는 비트맵을 사용하는 것이 최고다. 하지만 간단한 스킨 벡터 그래픽은 메모리를 절약해 주기도 하며 품질 저하(blurring) 없이 어떠한 해상도에서도 확장이 가능하다.

3.1.6 2 의 제곱

모든 비트맵은 2 의 제곱 크기임을 명심하라. 2 의 제곱 비트맵 사용예는 다음과 같다.

- 16x16
- 32x32
- 64x64
- 128x128

- 128x256
- 256x256
- 512x256
- 512x512

3.2 알려진 이슈와 추천하는 워크플로우

이번 장에서는 아트관련 플래시 이슈 중에서 알려진 것들과 스케일폼 CLIK으로 작업할 때 추천하는 워크플로우를 다룬다.

3.2.1 컴포넌트 복제

플래시에서 컴포넌트 복제 시에 원본의 링크 정보와 컴포넌트 정의 정보가 복사되지 않는 문제가 알려져 있다. 링크 정보 없는 컴포넌트는 작동할 수 없기 때문에 이번 장에서는 이 문제에 대한 2 가지 해결책을 제공할 것이다.

3.2.1.1 컴포넌트 복제 (방법 1)

수정되지 않은(스킨 없는) CLIK 컴포넌트(예를 들어서 Button 같은 경우)를 외부 FLA 파일에서 목적 FLA 파일로 복제하는 가장 빠른 방법은 다음과 같다.

1. *CLIK_Components.fla* 파일을 오픈한다.
2. 파일의 라이브러리에서 컴포넌트(예 Button)를 목적 FLA로 복사한다. 이 과정은 원본 라이브러리에서 [마우스 우클릭→ *Copy*], 목적 FLA 라이브러리에서 [마우스 우클릭→ *Paste*]선택하면 된다.
3. 목적 FLA 라이브러리에 있는 컴포넌트를 마우스 우클릭하고 *Rename*을 선택하여 'Button'이 아닌 다른 이름으로 바꾼다.
4. 목적 FLA 라이브러리에 있는 컴포넌트를 다시 마우스 우클릭하고 *Properties*를 선택한다.
5. *Identifier* 항목을 변경해서 3 번에서 선택한 새로운 이름과 매치시킨다.
6. 라이브러리 창의 빈영역을 마우스 우클릭하여 *Paste*를 선택한다. 원본 컴포넌트의 새로운 복제본이 링크정보를 고스란히 포함하여 붙여넣기 될 것이다.

3.2.1.2 컴포넌트 복제 (방법 2)

동일 라이브러리에서 심볼(컴포넌트)을 복제할 때는 링크정보가 새로운 복제 심볼에 복사되지 않을 것이다. 이 정보는 컴포넌트가 작동하는데 필요하기 때문에 입력해줘야 한다.

1. 라이브러리 창에서 복제할 컴포넌트를 우클릭해서 *Properties*를 선택한다.
2. *Properties* 창에서 텍스트(예: scaleform.clik.controlsButton)를 더블클릭해서 *Class* 텍스트필드를 하이라이트 시킨다. 그리고나서 [CTRL+C]를 눌러서 복사한다.
3. *Cancel* 을 누른다.
4. 복제할 컴포넌트를 다시 우클릭해서 *Duplicate*를 선택한다.
5. *Duplicate Symbol* 창에서 *Export for ActionScript* 체크박스를 클릭한다.
6. *Class* 항목을 선택하고 [CTRL+V]를 눌러 링크정보를 텍스트필드에 붙여넣는다.
7. *OK*를 누른다
8. 라이브러리 창의 새로 복제된 컴포넌트에서 우클릭하여 *Component Definition*을 선택한다.
9. 빈 *Class* 텍스트필드를 클릭해서 [CTRL+V]를 눌러 링크정보를 붙여넣는다.
10. *OK*를 누른다

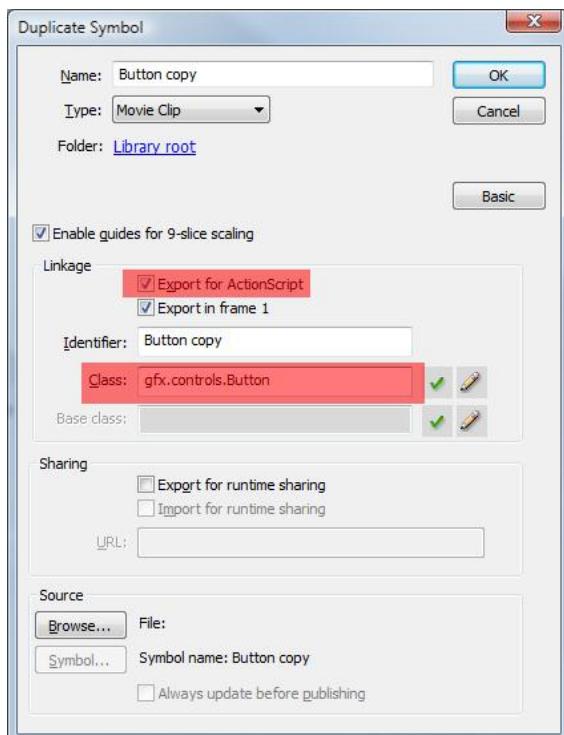


그림 50: 심볼 링크 복제 (붉은색 영역을 채워 넣어야 한다).

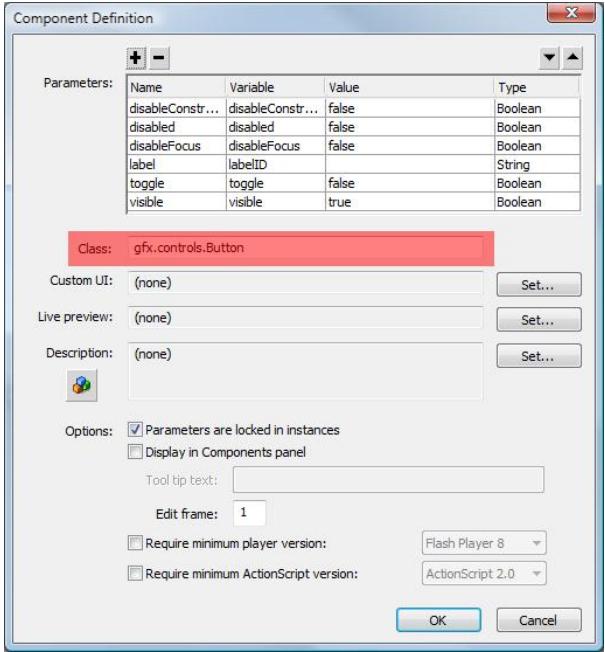


그림 51: 컴포넌트 정의 (클래스 부분을 채워 넣어야 한다).

3.3 스키닝 예제

스케일폼 CLIK 의 최대 장점은 비쥬얼과 함수 레이어의 분리에 있다. 이렇게 함으로써 프로그래머와 아티스트가 각기 독립적으로 작업을 할 수 있는 것이다. 룩앤파일(look and feel)을 손쉽게 커스터마이징 할 수 있다는 것이 이러한 분리의 중요한 장점이다.

프리빌트 CLIK 컴포넌트가 표준 룩앤파일을 가지고 있지만, 이는 사용자가 자신의 필요에 따라서 커스터마이징 하라는 의미다. 다음 장에서는 프리빌트 컴포넌트를 커스터마이징 하는 접근 방식에 대하여 다룰 것이다.

CLIK 컴포넌트는 표준 플래시 심볼만큼이나 스킨 입히기가 쉽다. FLA 라이브러리에 있는 컴포넌트를 간단하게 더블클릭하면 컴포넌트의 타임라인에 대한 접근할 수 있으며, 또한

- 플래시에 있는 기본 스킨 설정
- 플래시에서 밑바닥부터 커스텀 스킨 제작. 또는
- 포토샵이나 일러스트레이터에서 제작한 아트 애셋을 플래시로 임포트

스키닝에 관련된 상세한 학습서는 [Getting Started with CLIK](#) 을 참고하도록 하자.

3.3.1 StatusIndicator 스키닝

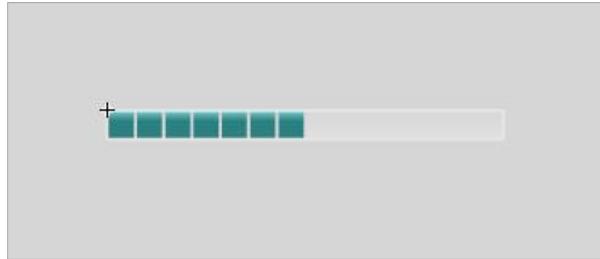


그림 52: 스킨없는 StatusIndicator

대부분의 다른 커포넌트들이 버튼에 기반하고 있기 때문에 비슷한 접근 방식을 취하는데 반해서 StatusIndicator 는 다른 접근방식이 필요한 독특한 컴포넌트다. StatusIndicator 를 열고 타임라인을 살펴보도록 하자. *indicator* 와 *track* 이라는 2 개의 레이어가 있다. *track* 은 배경 그래픽을 보여주는데 사용된다. *indicator* 레이어는 몇가지 키프레임을 사용하며, 또한 비트맵이사 벡터그래픽을 사용해서 낮은값에서 높은값으로의 증가단계를 나타내준다.

본 학습은 몇가지 비트맵 파일을 만들 것인데, 이 파일은 상태지시자를 나타내기 위해서 다중레이어로 구성된 단일 포토샵 파일이다. 여기서는 StatusIndicator 에 스킨을 입히는 한가지 길만 다를 것이지만 스테이지상에 그래픽 이미지를 만들고 조합하는 다양한 방법이 존재한다. 물론, 어떤 방식을 사용하든 StatusIndicator 가 적절하게 작동하기 위해서는 최종 결과물이 동일해야 한다.



그림 53: StatusIndicator PSD 파일

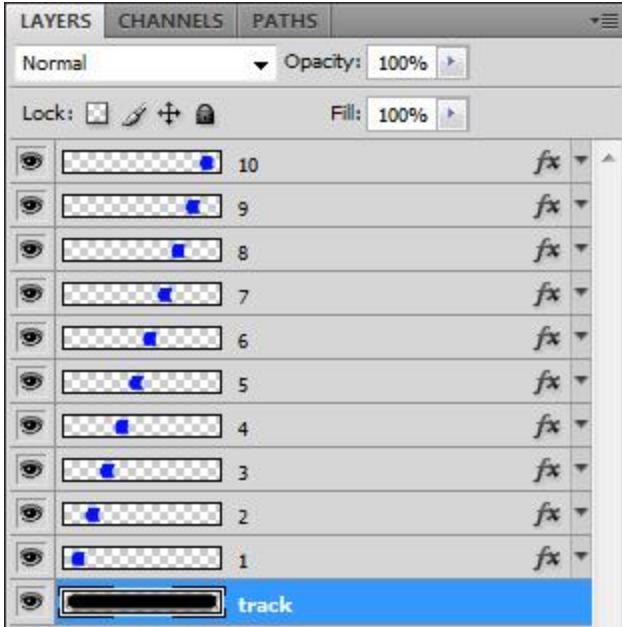


그림 54: StatusIndicator PSD 파일의 포토샵 레이어 설정

1. 위에 있는 그림처럼 포토샵에서 StatusIndicator 비트맵을 만든다. 레이어 상의 각 이미지 위치뿐만이 아니라 순서와 번호에도 주의하기 바란다. 배경은 투명이다.
2. PSD 파일을 저장한다.
3. 플래시에서 상단메뉴 *File → Import → Import to Stage*
4. StatusIndicator 스킨 PSD 파일 선택
5. *Import* 창에서 *Convert layers to 향목*을 *Layers*로 선택
6. *OK*버튼
7. PSD 파일의 각 레이어에 새로운 플래시 타임라인 레이어가 생성되었을 것이다. 위의 이미지를 사용했다면 10 개의 레이어가 생성되는데, 이는 PSD 파일에 10 개의 레이어가 있기 때문이다(각 프레임마다 하나의 레이어). 각 레이어는 1 부터 10 까지 레이블 되어 있는데, 레이어 1 이 제일 밑이고, 레이어 10 이 제일 위다. 레이어 1 을 선택한다.
8. 스테이지 상의 모든 비트맵 이미지에 선택 박스를 그린다.
9. 스키드 없는 트랙 상으로 이미지의 위치를 이동한다. 그리고 필요한 만큼 크기를 조절한다.
10. 레이어 1 의 첫번째 키프레임을 선택해서 프레임 6 으로 드래그한다.
11. 레이어 1 의 프레임 11 에서 마우스 우클릭→ *Insert Keyframe* 으로 새로운 키프레임 추가
12. 레이어 2 의 비트맵을 선택하고 [CTRL+X]로 잘라낸다.
13. 레이어 1 의 프레임 11 에 있는 새 키프레임을 선택하고 [CTRL+SHIFT+V]를 눌러서 레이어 1 과 정확히 동일한 곳에 비트맵을 위치시킨다.
14. 이 과정을 반복해서 모든 10 개의 비트맵이 동일 레이어(레이어 1)의 정확한 키 프레임에 들어 가도록 한다. 각각의 연속된 비트맵은 마지막 키 프레임의 5 프레임 뒤에 복사되어야 한다.

레이어 2의 비트맵은 키프레임 11, 레이어 3의 비트맵은 프레임 프레임 16, 레이어 4의 비트맵은 프레임 21 등이다. 10 개의 이미지 PSD를 사용한다면, 최종 키프레임은 프레임 51에 있을 것이다.

15. 레이어 2~10을 삭제해서 깨끗이 하자.
16. 마지막 비트맵 키프레임 뒤에도 추가 키프레임이 타임라인에 있다면, 즉 또 다른 5 개의 키프레임이 있다면, 남겨진 프레임을 선택한 후에 마우스 우클릭→ *Remove Frames* 으로 삭제한다. 이 학습서대로 따라 왔다면 마지막 프레임은 55 다.
17. 구 *indicator* 레이어를 선택해서 삭제한다.
18. 구 *track* 레이어를 선택하여 삭제한다. 임포트된 신 *track* 레이어를 삭제하지 않도록 주의한다.
19. 레이어 1을 선택하여 *indicator* 라고 이름을 바꾼다.
20. *indicator* 레이어와 *track* 레이어를 드래그해서 *actions* 레이어 아래로 옮긴다. 이때, *track* 가 *indicator* 레이어 아래에 있어야 한다.

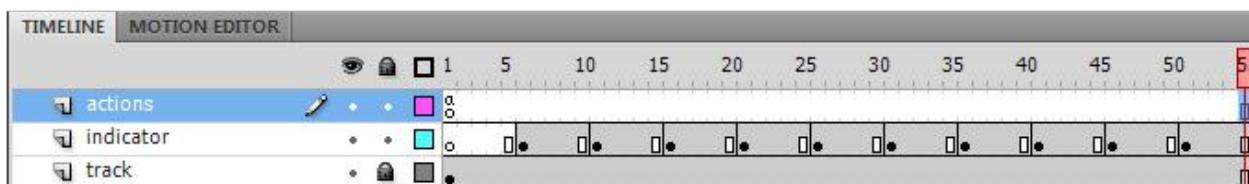


그림 55: The 최종 타임라인 (키프레임 위치와 최종 프레임의 위치를 확인하자).

21. StatusIndicator 의 타임라인을 종료한다.
22. Inspectable parameter 값을 1 과 10 사이의 아무 값이나 정한다.
23. 파일 저장
24. 새로 스킨이 적용된 지시자를 퍼블리싱

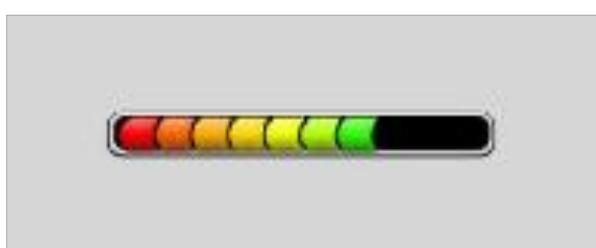


그림 56: 스킨있는 StatusIndicator.

3.4 폰트와 Localization

이번 장은 스케일폼 CLIK 컴포넌트에서 폰트 사용에 관해 다룬다.

3.4.1 개요

스케일폼 Scaleform에서 폰트가 정확히 렌더링 되려면 필요한 글리프(폰트 문자)가 SWF에 내장되거나 Scaleform Localization System을 사용해서 설정되어야 한다. 다음에서 플래시 UI에서 폰트를 관리하는 방법을 다루어보겠다.

3.4.2 폰트 내장

플래시 플레이어와는 달리 스케일폼 Scaleform은 내장되어야 플래시 파일을 제대로 보여줄 수 있다. Scaleform 플레이어는 폰트가 내장되어 있지 않을 경우에는 시스템에 폰트가 있다 하더라도 빈 사각형만 보여준다. 이 방법의 장점은 스케일폼 Scaleform에 폰트가 정확히 내장되어 있다면 손쉽게 볼 수 있다는 것이다. 프리빌트 컴포넌트에는 Slate Mobile이 각 textField 인스턴스에 내장되어 있다. Slate Mobile에는 중국어, 일본어, 한국어(CJK)문자나 글리프가 없으며 프리빌트 컴포넌트는 ASCII 글리프만 포함하고 있다는 것이다. 이 문제는 Slate Mobile을 CJK 글리프가 있는 폰트로 교체하고, 적절한 내장옵션을 설정하면 된다.

textField에 직접 폰트는 내장하는 것은 Scaleform의 Localization System(3.4.4에서 설명)과 호환되지 않는다. 솔직히 스케일폼은 폰트를 세팅하는데 Scaleform Localization System을 사용하기 권하는데 이는 직접 내장하는 방법보다 많은 혜택을 줄 수 있기 때문이다. 하지만 Localization가 필요하지 않은 몇 가지의 특정 케이스에서는 폰트를 내장하는 것이 더 효율적인 방법이 될 수 있다. UI 관리와 비슷하게 폰트 관리 또한 localization과 메모리 관리 같은 여러 가지 고려할 사항들이 많다.

3.4.3 textField에 폰트 내장

폰트는 동적, 혹은 입력 textField에만 내장하면 된다. 정적 텍스트는 컴파일시에 자동적으로 외곽선(순수 벡터 형태)으로 변형된다. 폰트를 동적 textField에 내장하려면 스테이지상의 textField를 선택한 후 속성 지시자(Window → Property Inspector)에서 *Embed* 버튼을 누르면 된다. 내장해야하는 문자나 문자 집합을 선택한 후에 OK를 누른다. 이 과정이 완료되면 FLA에서 폰트를 사용하는 것이 가능하다. 동일한 FLA 내의 다양한 textField에서 동일한 폰트를 사용한다면 이런 식으로 한번만 내장하면 된다. 반대로 말하면, 동일한 폰트를 여러 번 내장하더라도 파일 크기나 메모리 사용이 증가하지 않는다는 것이다. 중국어와 같은 많은 숫자의 글리프가 담긴 문자 세트는 로딩이 되었을 때 많은 메모리 공간을 차지할 수 있음에 주의하자.

3.4.4 Scaleform Localization System

Scaleform Localization System 은 hot-swapping 방식을 사용해서 폰트 라이브러리를 현재 로케일이 변경될 때마다 로드/언로드 한다. 이들 폰트 라이브러리들은 콘텐츠 SWF 가 사용하는 내장된 폰트 글리프를 포함하고 있는 SWF 파일 그 자체이다. Scaleform 는 폰트 라이브러리로부터 글리프를 사용해 요구에 따라 폰트를 동적으로 변경할 수 있는 강력한 방법을 제공한다.

Scaleform Localization System 은 textField 에 글리프를 직접 내장할 경우 폰트를 대체할 수 없기 때문에 textField 는 반드시 import 된 폰트 심볼을 사용하여야 한다. 일반적으로 폰트 심볼은 gxfontlib.fla 라는 파일에 생성되며 콘텐츠 SWF 에 import 된다. 이렇게 import 된 폰트는 폰트 대체를 지원하는 모든 textField 에 세팅된다. Scaleform 는 이제 이 폰트 심볼 링크를 납치하여 현재 위치에 따라 다른 폰트를 로딩한다.

폰트 라이브러리는 어떤 폰트로 export 할 필요가 없다. 단지 필요한 폰트(어떻게 폰트를 내장하는지에 대해서는 바로 전 섹션을 참고하자)만 내장하면 된다. Scaleform Localization System 은 fontconfig.txt 파일을 사용하여 사용할 폰트 라이브러리를 위치마다 지정하고 textField 에서 사용하는 폰트 심볼과 폰트 라이브러리에서 내장한 물리적 폰트 간 폰트 맵핑을 지정한다.

또한, fontconfig.txt 파일은 번역 맵을 포함하고 있다. Scaleform 현지화 시스템은 모든 동적 혹은 스테이지상 입력 textField 에 출력된 텍스트의 빠른 번역을 실행한다. 예를 들어 만약 textField 가 '\$TITLE'이란 텍스트를 포함하고 있고 번역 맵은 '\$TITLE=Scaleform' 처럼 맵핑을 했다고 한다면 textField 는 자동적으로 'Scaleform'을 출력할 것이다.

본 섹션에 포함된 정보는 Scaleform 폰트와 Localization System 에 대한 간단한 개요만 설명하고 있다. Scaleform 의 폰트와 Localization System 에 대하여 조금 더 자세하게 알고 싶다면 [Font Overview](#) 문서를 참고하자.

4 프로그래밍 세부사항

이번 장은 프레임워크의 작동부분을 설명하고, CLIK 컴포넌트 아키텍처의 고수준 이해를 위한 각 서브 시스템 별 설명도 제공할 것이다.

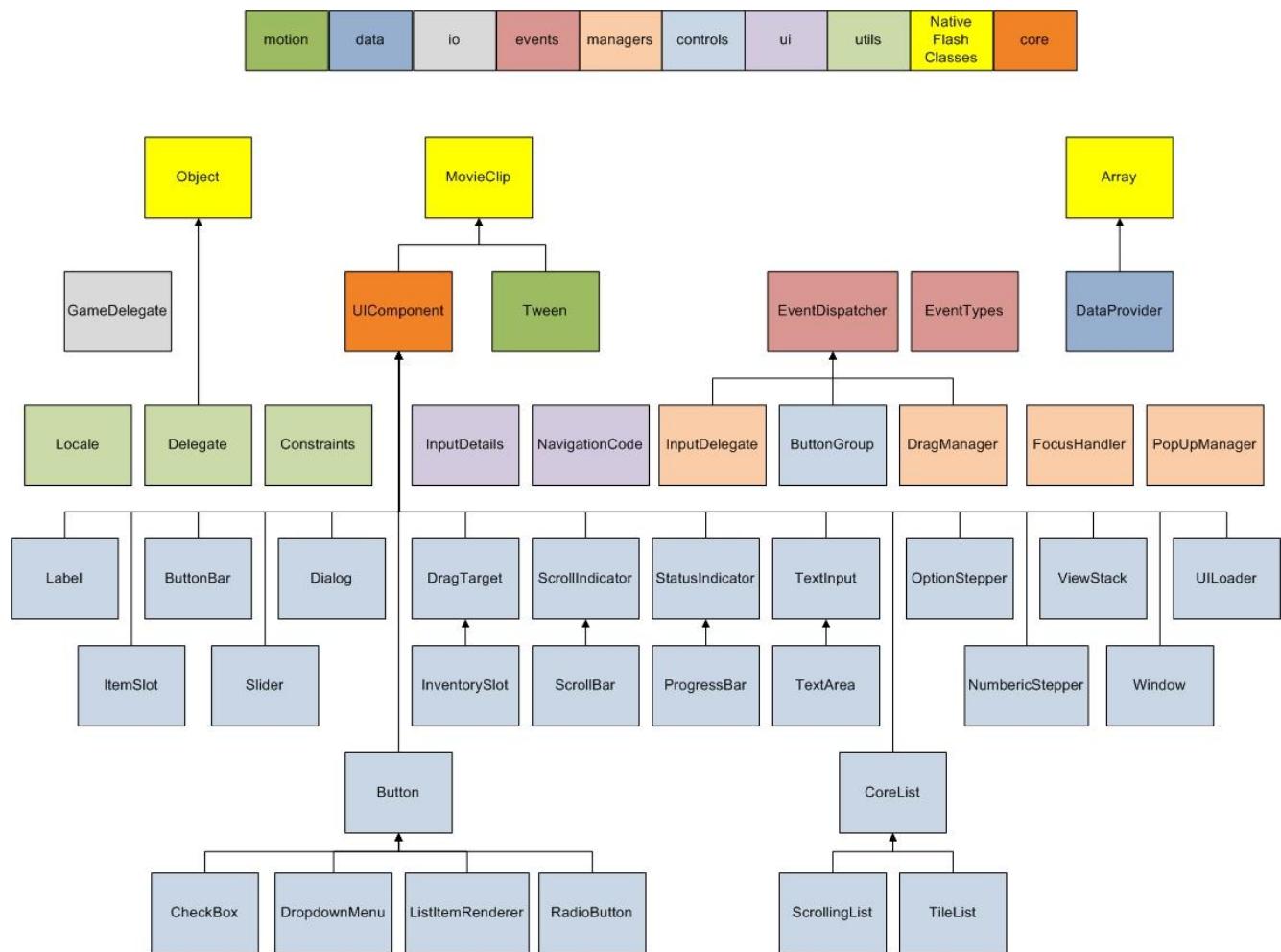


그림 57: 스케일폼 CLIK 클래스 계층구조

4.1 *UIComponent*

프리빌트 컴포넌트 장에서는 스케일폼 CLIK에서 제공하는 구체적인 컴포넌트에서 사용되는 클래스를 설명하였다. 이들 모든 컴포넌트는 핵심 기능을 UIComponent(scaleform.clik.core.UIComponent)

클래스에서 상속받는다. 이 클래스는 모든 CLIK 컴포넌트의 기반이기 때문에, 스케일폼은 커스텀 컴포넌트의 경우에도 UIComponent로부터 상속받을 것을 권한다.

UIComponent 자체는 플래시 8 MovieClip 클래스를 상속받은 것이다. 따라서 표준 플래시 MovieClip의 메소드와 속성을 상속받는다. UIComponent 클래스에서 지원하는 몇 가지 읽고/쓰기 속성이 있다.

- **enabled;**
- **visible;**
- **focused;**
- **focusable;**
- **width;**
- **height;**
- **scaleX;**
- **scaleY;**
- **displayFocus** 컴포넌트가 포커스 상태를 출력하려면 이 속성을 true로 한다. 세부설명은 "포커스 다루기"장을 참고하라

UIComponent는 하위클래스에서 구현해야만 하는 빈 메소드 몇 개가 있다.

- **configUI**. 컴포넌트 설정을 수행할 때 호출된다.
- **draw**. 컴포넌트가 무효화될 때 호출된다. 세부내역은 "무효화"참고
- **changeFocus**. 컴포넌트가 포커스를 얻거나 잃을 때 호출된다
- **scrollWheel**. 커서가 컴포넌트 위에 있을 때 스크롤 휠이 작동되면 호출된다.

UIComponent는 이벤트 신청과 배분을 지원하기 위해서 EventDispatcher 클래스와 혼합된다. 따라서, 모든 하위 클래스는 이벤트 신청과 배분을 지원한다. 자세한 설명은 이벤트 모델 장 참고.

4.1.1 초기화

이 클래스는 다음과 같은 초기화 과정을 onLoad()이벤트 핸들러 내부에서 수행한다.

- 기본 크기를 MovieClip 크기로
- 컴포넌트 설정 수행. 이 과정에서 configUI()메소드 호출
- 컨텐츠를 그린다. 이 과정에서 validateNow() 메소드를 호출하여 즉시 다시 그리기 수행. 즉, draw()호출

4.2 컴포넌트 상태

거의 대부분의 CLIK 컴포넌트는 비주얼 상태를 지원한다. 상태는 컴포넌트 타임라인의 지정된 키프레임으로 이동하거나 상태 정보는 부요소에 전달하면 설정된다. 다음과 같은 공용 상태 설정이 있다.

4.2.1 Button 컴포넌트

(마우스 동작에 반응하는) 버튼처럼 작동하는 모든 컴포넌트들이 이 범주에 해당한다고 볼 수 있다. 이러한 형태를 사용하는 CLIK 컴포넌트는 버튼과 버튼의 변종들이다. 대표적으로 ListItemRenderer, RadioButton, CheckBox 가 이에 해당한다. ScrollBar 같은 복합 컴포넌트의 부요소들도 Button 컴포넌트다. 이 범주에 해당하는 CLIK 컴포넌트는 Button 클래스(scaleform.clik.controls.Button)를 직접 사용하거나 Button 클래스를 상속받은 클래스를 사용한다.

버튼 컴포넌트가 지원하는 기본 상태는 다음과 같다.

- **up** 혹은 기본 상태
- **over** 마우스 커서가 컴포넌트 위에 있을 때, 혹은 포커스를 가졌을 때
- **down** 컴포넌트 위에서 마우스 버튼이 눌렸을 때
- **disabled** 컴포넌트가 disable 되었을 때

Button 컴포넌트는 프리픽스 상태(prefixed state)도 지원한다. 이 상태는 다른 속성값에 따라서 설정된다. 기본적으로 코어 CLIK Button 컴포넌트는 "selected_"라는 프리픽스만 지원하며, 컴포넌트가 선택된 상태일 때 프레임 레이블에 추가된다.

선택된(selected)상태를 포함해서 Button 컴포넌트는 기본적으로 다음 상태를 지원한다.

- **up** 혹은 기본 상태
- **over** 마우스 커서가 컴포넌트 위에 있을 때, 혹은 포커스를 가졌을 때
- **down** 컴포넌트 위에서 마우스 버튼이 눌렸을 때
- **disabled** 컴포넌트가 disable 되었을 때
- **selected_up** 혹은 기본 상태
- **selected_over** 마우스 커서가 컴포넌트 위에 있을 때, 혹은 포커스를 가졌을 때
- **selected_down** 마우스 버튼이 눌렸을 때
- **selected_disabled** 상태

주의: 여기서 언급되는 상태들은 CLIK Button 컴포넌트에서 지원하는 상태들의 일부분일 뿐이다. [Getting Started with CLIK Buttons](#) 문서를 보면 완벽한 지원 상태들에 대하여 알 수 있다.

CLIK 버튼 클래스는 getStatePrefixes()메소드를 제공하는데, 컴포넌트의 속성에 따른 프리픽스를 변경할 수 있다. 이 메소드 정의는 다음과 같다.

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

앞서 언급한 것처럼 CLIK Button은 기본적으로 "selected_" 프리픽스만 지원한다. getStatePrefixes()메소드는 선택된 상태에 따라서 다른 프리픽스 배열을 반환한다. 이 프리픽스 배열은 재생하려는 프레임을 결정하기 위해서 적절한 상태 레이블과 연계되어 내부적으로 사용된다.

상태가 내부적으로 설정되면, (즉 마우스가 위로 지나가면) 프레임 레이블 리스트 참조테이블이 검색된다. Button 클래스의 stateMap 속성은 프레임 레이블 맵핑 상태를 정의한다. 다음은 CLIK 버튼 클래스 내의 상태 맵핑 정의값이다.

```
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"]
}
```

각각의 상태는 하나 이상의 타겟 레이블을 가질 수 있다. 상태 맵에서 반환된 값은 getStatePrefixes()에서 반환된 프리픽스 값과 결합되어 재생할 프레임의 타겟 리스트를 생성한다. 다음 그림은 올바른 재생 프레임을 결정하기 위한 전체 프로세스를 나타낸 것이다.

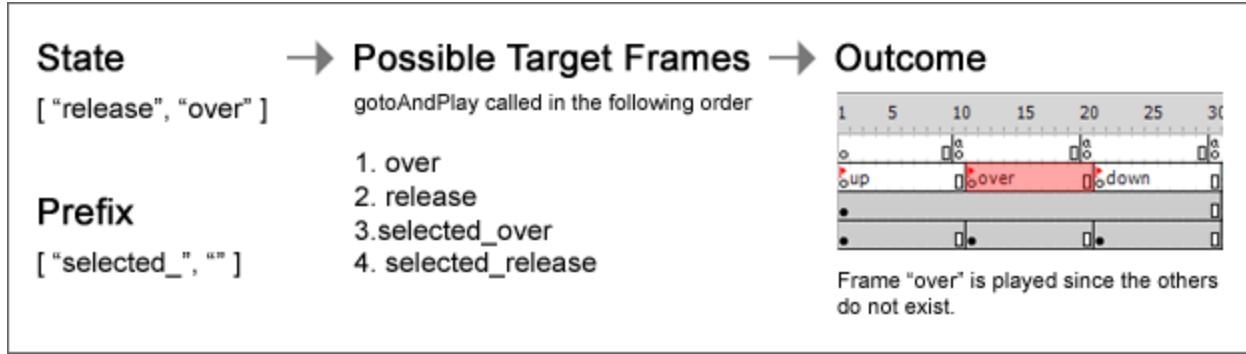


Figure 58: Process used to determine the correct keyframe to play.

재생시작점은 항상 최종 사용 프레임으로 점프한다. 따라서, 만약 어떤 프리픽스 프레임을 사용할 수 없다면 컴포넌트는 기본적으로 직전에 요청된 프레임으로 이동한다. 개발자는 이를 상태 맵을 오버라이드해서 커스텀 한 작동을 시킬 수 있다.

4.2.2 버튼이 아닌 상호작용 컴포넌트

이들은 상호작용하고, 포커스를 받지만 마우스 이벤트에 응답하지는 않는 모든 컴포넌트를 말한다. 이러한 형태를 사용하는 CLIK 컴포넌트는 ScrollingList, OptionStepper, Slider, TextArea 가 있다. 이들 컴포넌트는 자식 요소를 통해서 마우스 이벤트에 반응할 수 있다. 비버튼 상호작용 컴포넌트에서 지원되는 상태는 다음과 같다.

- **default** 상태
- **focused** 상태
- **disabled** 상태

4.2.3 비 상호작용 컴포넌트

이들은 사용작용하지 않고, disable 될 수 있는 모든 컴포넌트를 말한다. 레이블 컴포넌트가 기본 컴포넌트 중에서 유일하게 상태를 지원하는 비상호작용 컴포넌트다.

- **default** 상태
- **disabled** 상태

4.2.4 특수한 경우

앞서 설명한 규칙에 어긋나는 몇 가지 컴포넌트가 있다. StatusIndicator 와 이 클래스를 상속받은 ProgressBar 는 컴포넌트의 값을 출력할 때 타임라인을 사용한다. 재생시작점은 컴포넌트의 퍼센트 값(%)을 나타내는 프레임으로 설정될 것이다. 예를 들어서 최소가 0, 최대값 10 인 50 프레임 타임라인의 StatusIndicator 에서, 현재 값을 5(50%)로 설정하면 gotoAndStop()이 프레임

25(50 프레임의 50%)가 될 것이다. 이를 컴포넌트의 출력을 관리하기 위해서 프로그램적으로 확장하는 것은 엄청나게 쉽다. `updateValue()`메소드는 수정가능하며, 오버라이드를 통해서 작동방식을 변경할 수도 있다.

경우에 따라서, 컴퍼넌트들은 그것들의 추가적인 컴퍼넌트 상태들에서 제공하는 기본적인 행동들 외에도 특수한 모드가 있을 수 있다. `TextInput`과 `TextArea` 컴퍼넌트들은 포커스가 없고 `asAsButton` 속성이 마우스 커서가 roll over 와 roll out 이벤트를 지원하도록 설정되었을 때 over 와 out 상태가 사용 가능하다.

4.3 이벤트 모델

CLIK 컴포넌트 프레임워크는 이벤트 모델이라고 알려진 통신 패러다임을 사용한다. 컴포넌트는 변경이나 상호작용이 발생하면 이벤트를 "발송(dispatch)"한다. 그리고, 컨테이너 컴포넌트는 다른 이벤트를 신청할 수 있다. 이를 통해 여러 오브젝트가 변경되는 것을 통보 받을 수 있습니다. ActionScript 3 에는 Clik에서 활용하는 강력한 이벤트 시스템이 실제로 포함됩니다.

ActionScript 3 이벤트 시스템에 대한 자세한 정보는 다음에서 찾을 수 있습니다.

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

4.3.1 사용예와 가장 좋은 연습(Usage and Best Practices)

4.3.1.1 이벤트에 신청하기(Subscribing to an Event)

이벤트에 신청을 하려면 `addEventListener()`메소드를 사용하라. 이때 대기할 상호작용 타입을 매개변수로 넘긴다. 이와 반대로 `removeEventListener()`는 이벤트에대한 신청을 취소한다. 동일한 매개변수로 다중 리스너가 등록되면 하나의 이벤트만 발생한다. 이 각각의 메소드에는 또한 사전에 정의된 기능 또는 기능 변수와 연결된 형식 기능의 리스너 파라미터가 필요합니다.

CLIK는 이벤트에 대한 좀 더 자세한 정보를 제공하기 위해 각각의 고유한 프로퍼티와 함께 다양한 사용자 정의 이벤트를 사용합니다. 이벤트 클래스에서 얻는 모든 표준 Clik 이벤트 클래스는 `scaleform.clik.events`에서 찾을 수 있습니다.

`addEventListener()`는 또한 기본값이 제공되는 파라미터 3 개를 더 받습니다. **useCapture**, **priority** 및 **useWeakReference**입니다. 일반적으로 사용자는 모든 이벤트 리스너를 약한 참조로 사용하여, 그렇지 않으면 조각모음이 될 오브젝트에 대해 강한 참조를 유지하는 것을 피하고자 합니다. 리스너가 약한 참조를 사용하게 하려면 `addEventListener()` 문법이 다음과 같아야 합니다.

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

`addEventListener()` 파라미터에 대한 자세한 정보는 다음에서 찾을 수 있습니다.

[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);
}
```

위 예제에서 이 클래스는 `buttonInstance`에서 보낸 `ButtonEvent.CLICK` 이벤트에 대한 리스너를 생성합니다. 이 이벤트가 수신될 때마다 `onButtonClick` 기능이 실행됩니다. 리스너 기능이 이벤트 리스너를 제거합니다.

리스너에 대한 이벤트 파라미터의 프로퍼티는 이벤트의 기원 및 형식에 따라 다릅니다.

CLIK 컴포넌트에 의해서 생성되는 이벤트 객체(와 그 속성)에 대한 전체 리스트는 [프리빌트 컴포넌트]장을 참고하기 바란다.

4.3.1.2 이벤트 발송(Dispatching an Event)

CLIK 컴포넌트가 변경이나 사용작용에 대한 통지를 신청한 리스너들에게 이벤트를 알려줄때는 `dispatchEvent()` 메소드를 사용한다. 이 메소드는 전달인자가 하나만 필요하다. 전달인자는 관련 데이터를 포함하고 있는 객체로서, 발송될 이벤트 타입이 지정된 위임형 속성값을 함께 가지고 있다. 컴포넌트 프레임워크는 자동적으로 타겟 속성을 추가하는데, 이 타겟 속성은 이벤트를 발송하는 객체의 참조형이다. 물론 커스텀 타겟으로 오버라이드하여 수동으로 설정할 수도 있다.

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

4.4 실행 시 구성 요소 생성

ActionScript 3에서 실행 시 구성 요소를 동적으로 생성하려면 다음 문법을 사용합니다.

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

public var myButton:Button;

// Retrieve a reference to the Class definition.
var classRef:Class = getDefinitionByName("ButtonLinkageID") as Class;
// Instantiate a new instance of the class and store it.
if (classRef != null) { myButton = new classRef () as Button; }
myButton.addEventListener(ButtonEvent.CLICK, onButtonClick, false, 0, true);
```

이 예제에서 "ButtonLinkageID"와 같은 클래스 세트 및 scaleform.clik.controls.Button과 같은 기본 클래스 세트와 함께 .FLA 라이브러리에 심볼이 존재합니다.

클래스가 있고 BaseClass가 없이 .FLA 라이브러리에 심볼이 존재하는 경우, 심볼 및 클래스가 함께 연결되는 것과 같이 해당 심볼의 인스턴스는 클래스의 생성자를 사용하여 생성될 수 있습니다.

4.5 포커스 다루기

스케일폼 CLIK 컴포넌트는 커스텀 포커스 처리 프레임워크를 사용하는데 (이는 대부분의 컴포넌트에 구현되어져 있다), 프레임워크가 없는 컴포넌트와 심볼들과도 잘 작동해야 한다.

FocusHandler (scaleform.clik.managers.FocusHandler) 매니저는 컴포넌트 클래스가 하나 생성되면 인스턴스화 된다. 따라서 직접 FocusHandler를 인스턴스화 할 필요는 없다.

모든 focus 변경은 Scaleform 플레이어 레벨에서 마우스나 키보드(게임패드)의 focus 변경 또는 ActionScript를 통해 발생합니다. focus 설정을 위한 ActionScript 메소드에는 다음이 포함됩니다.

- myUIComponent.focused = 1; // 1은 인덱스 0에 있는 컨트롤러에 대한 비트임.
- stage.focus = myMovieClip; // MovieClip이 이 로직과 함께 스테이지에 있는 동안.
- FocusHandler.getInstance().setFocus(myMovieClip); // FocusHandler는 정적 클래스임.

4.5.1 사용예와 가장 좋은 연습(Usage and Best Practices)

focus 는 InteractiveDisplayObject 에 설정되는 데 필요하며, 그렇지 않으면 focus 는 플레이어에 대한 기본값이 됩니다. 포커스가 인스턴스에 설정되지 않으면 포커스 관리 시스템이 제대로 작동하지 않는다(즉, 텁키로 포커스를 바꿀 수 없다). 포커스는 아무 컴포넌트의 포커스 속성을 true 로 하면된다. 컴포넌트가 아닌 요소에도 작동하는 메소드는 다음과 같다.

false 로 설정된 **tabEnabled** 및 **mouseEnabled** 가 있는 MovieClip 은 Scaleform 또는 Flash 플레이어에 의해 focus 될 수 없고 focus 변경 이벤트를 생성하지 않습니다. 대부분의 CLIK 구성 요소는 기본적으로 해당 구성 요소 및 자식에 대한 tabEnabled 및 mouseEnabled 프로퍼티를 설정합니다.

특정한 경우 디자이너가 특정 구성 요소가 tabEnabled 및 mouseEnabled 둘 다 되도록 하고 싶으나 focus 를 수신하는 것이 불가능할 수 있습니다. AS3 에는 MovieClip.focusEnabled 가 없으므로 UIComponent.**focusable** 프로퍼티가 이 문제를 해결합니다. 이 프로퍼티가 false 로 설정되면 UIComponent 는 CLIK focus 프레임워크에 의해 focus 를 받을 수 없습니다. **focusable** 프로퍼티를 변경하면 이전에 구성 요소에 적용된 tabEnabled 및 mouseEnabled 구성을 유지하기 위해 구성 요소의 tabEnabled 및 mouseEnabled 프로퍼티 내부에 영향을 줍니다.

포커스를 가질 수 있는 (ScrollBar 같은) 부요소를 가진 컴포넌트는 focusTarget 속성을 사용해서 소유주 컴포넌트로 포커스를 넘긴다. 포커스가 컴포넌트로 변경되었을 때 FocusHandler 는 재귀적으로 focusTarget 체인을 검색해서 focusTarget 을 반환하지 않은 마지막 컴포넌트에 포커스를 넘긴다.

가끔은 플레이어나 어플리케이션에 실제 포커스가 없음에도 포커스가 있는 것처럼 보여야 할때가 있다. displayFocus 속성을 true 로 하면 컴포넌트가 마치 포커스가 있는 것처럼 행동한다. 예를 들어서 Slider 가 포커스를 가졌을 때 Slider 의 트랙도 포커스가 있는 것처럼 보여야 한다. 포커스 속성에 변화가 생기면 UIComponent.changeFocus() 메소드가 호출된다는 점을 잊지말자.

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

반대로, 때로는 구성 요소를 클릭할 수 있어야 하지만 드래그 가능한 패널 또는 마우스 전용 제어

효과를 얻을 수 있는 다른 구성 요소와 같은 탭을 통해 focus 를 받을 수 없습니다. 이 경우에 사용자는 tabEnabled 프로퍼티를 false 로 설정할 수 있습니다.

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

4.5.2 복합 컴포넌트에서 포커스 캡처

복합 컴포넌트란 다른 컴포넌트의 조합으로 이루어진 컴포넌트를 말하는데, ScrollingList, OptionStepper, ButtonBar 같은 것들이 있다. 이런 컴포넌트는 마우스 핸들러가 서브 컴포넌트에는 있지만, 자체에는 없는 것처럼 보인다. 즉, 플래시의 선택 엔진과 Scaleform 는 이들 아이템에 포커스를 줄수없으며, 빌트인 네비게이션 기능을 사용해도 컴포넌트 판독에 실패할 것이며, 결국 자식 컴포넌트만 찾게 될 것이다.

복합컴포넌트를 마치 하나의 단일 구성물처럼 보이게 하려면 다음과 같이 한다.

1. 마우스 핸들러가 있는 모든 부컴포넌트의 tabEnabled = mouseEnabled = focusable = false 를 한다(즉, OptionStepper 의 화살표 버튼)
2. 마우스 핸들러를 가진 모든 부컴포넌트의 focusTarget 속성을 컨테이너 컴포넌트로 설정한다. 컨테이너 컴포넌트는 focusable = true 로 되어있어야 한다.
3. 컨테이너 컴포넌트의 changeFocus 메소드에서 부컴포넌트의 displayFocus 속성을 true 로 한다.

이제, 부컴포넌트에 포커스가 가면, 컨테이너 컴포넌트로 포커스가 이동할 것이다.

4.6 입력 다루기

스케일폼 CLIK 컴포넌트가 플래시 8 MovieClip 클래스로부터 상속받았기 때문에 사용자 입력을 받게 되면 대부분 MovieClip 인스턴스와 동일하게 행동한다. 마우스 핸들러가 설치되어있다면 마우스 이벤트가 캡처될 것이다. 하지만, CLIK 과 관련된 가장 중요한 컨셉상의 변화점은 바로 키보드와 동급 컨트롤러 이벤트 어떻게 처리될 것인가 하는 것이다.

4.6.1 사용예와 가장 좋은 연습(Usage and Best Practices)

모든 비 마우스 입력은 InputDelegate(scaleform.clik.managers.InputDelegate) 매니저 클래스에서 가로챈다. InputDelegate 는 입력명령을 InputDetails(scaleform.clik.ui.InputDetails) 객체로 변환하는데,

이때 게임엔진에게 입력 커맨드의 값을 요청한다. 이러한 처리는 InputDelegate.readInput()메소드를 수정하면 된다. 일단 InputDetails 가 생성되면 입력 이벤트가 InputDelegate 에서 발송된다.

InputDetails 는 다음과 같은 속성들로 이루어져 있다.

- “key”같은 타입
- 눌려진 키 코드 같은 코드
- 버튼 벡터나 키눌림 타입 같은 입력관련 추가정보 값. Key 이벤트는 key up 과 key down 동작을 위해 발생되고, InputDetails 의 value 파라미터에 상응하는 것은 각각 inputValue.KEY_UP (“keyUp”) 또는 inputValue.KEY_DOWN (“keyDown”) “KeyDown” 둘 중 하나일 것이다.
- 가능한 경우라면 NavigationCode.UP (“up”), NavigationCode.DOWN (“down”), NavigationCode.LEFT (“left”), or NavigationCode.RIGHT (“right”),처럼 인간이 판독가능한 방향으로 맵핑하여 navEquivalent (navigation equivalent)값으로 정의.
NavigationCode(scaleform.clik.ui.NavigationCode)클래스에서 일반적인 방향관련 열거값 제공

FocusHandler 는 InputDelegate 에서 보낸 InputEvent 를 받은 다음, 현재 focus 된 구성 요소에서 같은 InputEvent 를 다시 받아 이벤트의 버블 연쇄에서 해당 구성 요소 또는 다른 구성 요소에 의해 처리되도록 합니다. 포커스 패쓰란 디스플레이 리스트 계층구조로 된 컴포넌트 리트스로 handleInput()메소드를 구현한다.

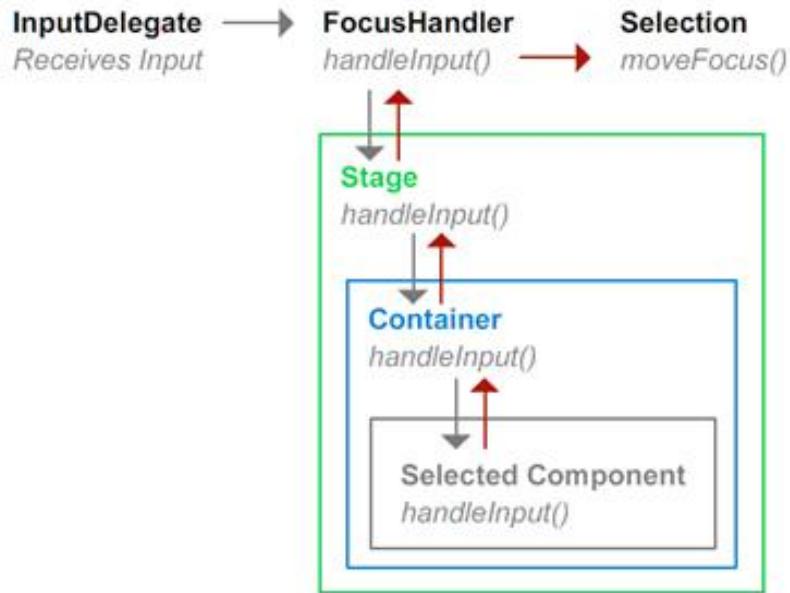


그림 59: handleInput() 체인

InputEvent 가 처리되면 처리된 프로퍼티는 true 로 설정되어야 합니다. 이벤트를 받은 다른 구성 요소는 event.isDefaultPrevented()를 확인하여 이벤트가 이미 처리되었는지 파악할 수 있습니다.

```

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 는 해당 .bubble 프로퍼티가 수정되지 않았다고 가정하여 ActionScript 3 이벤트 시스템에 의해 자동으로 촉발됩니다. ScrollingList 와 같은 일부 경우, 구성 요소는 이벤트 자체를 처리하려고 시도하기 전에 InputEvent 를 자식으로 전달하려고 할 수 있습니다. 인스턴스에 대해 Enter 키가

눌리면 ListItemRenderer 는 ButtonEvent.CLICK 이벤트 및 이벤트를 받는 목록을 생성하고 작업을 수행하지 않습니다.

```
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; }
    }
    ...
}
```

InputDelegate.instance 에 이벤트 리스너를 추가해서 입력 이벤트를 수동으로 대기하는 것거나 처리하는 것도 가능하다. 입력은 FocusHandler 에 의해서 캡처되고, 포커스 계층구조 아래쪽으로 전달된다.

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

InputDelegate.instance 에 이벤트 리스너를 추가해서 입력 이벤트를 수동으로 대기하는 것거나 처리하는 것도 가능하다. 입력은 FocusHandler 에 의해서 캡처되고, 포커스 계층구조 아래쪽으로 전달된다.

4.6.2 다중 마우스 커서

(Wii™같은) 몇몇 시스템은 다중 커서 디바이스를 지원한다. CLIK 컴포넌트 프레임워크도 다중 커서를 지원하지만 단일 SWF 파일에 다중 컴포넌트 포커스를 지원하지는 않는다. 두명의 사용자가 독립적으로 버튼을 누른다면, 마지막으로 클릭된 아이템이 포커스된 아이템이 된다.

모든 마우스 이벤트는 프레임워크에 포함된 controllerIdx 속성에 의해서 발송되는데, mouseIndex 란 이벤트를 생성한 입력디바이스의 인덱스를 말한다.

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

```
    trace(event.controllerIdx);  
}
```

4.7 무효화

무효화는 컴포넌트가 여러가지 속성이 변경되었을 때 다시그리기 회수를 제한하는 메커니즘이다. 컴포넌트가 무효화되면 다음 프레임에서 자기 자신을 다시 그리게 된다. 이렇게 함으로써 개발자는 컴포넌트에 동시에 여러가지 변경을 하면서도 업데이트는 단 한번만 할 수 있는 것이다. 가끔은 컴포넌트의 다시 그리기가 즉시 이루어져야 하는 경우가 있지만, 대부분의 경우는 무효화만으로 충분하다.

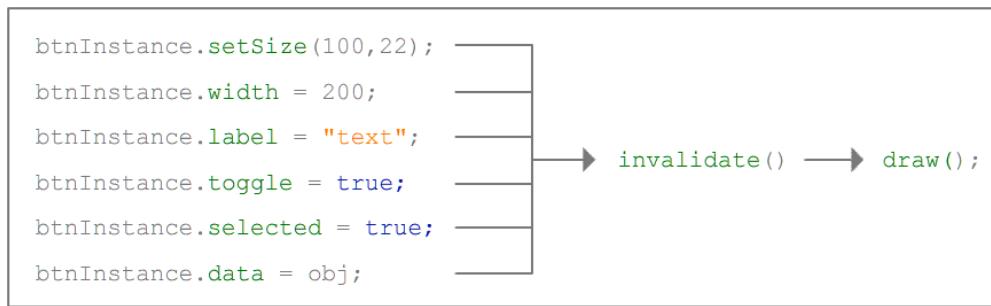


그림 60: CLIK 컴포넌트는 특정 속성이 변경되면 자동적으로 무효화한다.

4.7.1 사용예와 가장 좋은 연습(Usage and Best Practices)

컴포넌트 내부적인 변경이 발생하면 주로 `set` 함수에 의해서 발생한다), `UIComponent.invalidate()`가 호출되어야 하고, 마지막에는 컴포넌트 내의 `UIComponent.draw()`가 호출되어야 한다.

최상의 성능을 위해, `draw()` 내의 로직은 구성 요소(예: 데이터, 크기, 상태 등)의 어떤 측면이 현재 변경되지 않은 구성 요소의 부분이 업데이트되는 것을 피하는 데 무효한지 파악하는 `isValid()` 검사로 포괄되는 하위 섹션으로 나뉩니다. `invalidate()`가 호출되면 메소드가 `draw()` 내에 전체 수정을 하는 구성 요소의 모든 측면을 무효화합니다. 따라서 최상의 성능을 위해 하위 클래스는 `isValid()` 검사 내에서 `draw()` 로직을 포괄하여 동일한 패러다임을 따르고, `invalidate()` 호출 대신에 `invalidateData()`, `invalidateSize()`와 같은 특수 `invalidate()` 메소드 중 하나를 사용하여 불필요한 수정을 피합니다.

`invalidate()` 메소드는 불필요한 업데이트를 피하기 위해서 `draw()` 호출을 타이머에 기반해서 생성한다. 이미 존재하는 컴포넌트들을 사용하는 개발자는 무효화에 대한 처리를 해줄 필요는 없다. `invalidate()` 메소드는 구성 요소에 대한 다음 스테이지 무효화(Event.RENDER) 또는 다음

프레임(Event.ENTER_FRAME)에서 draw() 호출을 생성하여 단일 draw() 호출로 여러 변경을 일괄 처리합니다.

구성 요소에서 기존에 무효로 표시된 부분이 없는 경우 draw()가 호출되지 않습니다.

즉시 다시 그리기가 필요할 경우에는 UIComponent.validateNow()메소드를 사용할 수 있다.

4.8 컴포넌트 크기조절

CLIK 컴포넌트 크기 조절은 2 가지 방법이 있다.

1. reflowing 레이아웃을 사용하면 컴포넌트의 크기가 리셋되고 구성요소도 원래 크기로 재조정된다. 배경없이 부요소만 이는 컴포넌트는 이 방법을 사용한다.
2. 역 크기조절(counterscaling)로 정횡비를 유지시킨다. 이렇게 하면 컴포넌트는 크기가 조절되지만 구성요소는 역크기조절을 통해서 크기조절이 안된 것처럼 보인다. 이 방식은 배경 그림이 있는 컴포넌트, scale9grid, 왜곡되면 안되는 내용이있는 경우에 사용된다.

일반적으로 컴퍼넌트들은 Flash scale9Grid 의 reflow 방법을 사용한다. 이것은 개발자들이 Flash/Flex 컴퍼넌트와 비슷한 "skin"심볼들을 만들 수 있도록 도와준다. Reflowing 은 만약 컴퍼넌트들이 비교 할 수 있는 하위요소를 가질 때 가장 효과적이므로 이것은 container 와 비슷한 실체들을 위한 것이다.

Scaleform 에서 사용 가능한 확장된 scale9Grid 기능 때문에 카운터 스케일링 메서드는 Clik 을 위해 특별히 만들어졌다. 이것은 다른 컴퍼넌트 셋들에 의해 사용되는 좀 더 심층적인 layered 방식 대신에 프래임 형태를 사용하는 단일 에셋 컴퍼넌트들의 생성을 가능하게 했다. 기본적인 Clik 컴퍼넌트들은 최소한의 요소로 최대한의 효과를 내는데, 일반적으로 배경, 라벨과 임의의 아이콘 또는 하위 버튼을 포함한다. 이는 외곽 스케일링 방식을 위한 이상적인 방법이다. 그러나 외곽 스케일링은 setups(페널 레이아웃 등등...)같은 container 를 위한 것이 아니다. 각각의 경우, 나머지 하부 요소에 압박 받고 있는 백그라운드를 가진 Reflow 메서드 방식을 추천한다.

컴포넌트는 플래시 IDE 스테이지 상에서 크기가 조절가능하며, setSize()메소드나, width, height 속성을 사용해도 된다. 크기조절된 컴포넌트가 플래시 IDE 에서는 정확하게 보이지 않을 수도 있다. 이는 컴파일 안된 MovieClips 을 LivePreview 없이 컴포넌트로 사용할 때 발생하는 제약사항이다. 무비를 Scaleform Player 에서 테스트하는 것 만이 크기조절된 컴포넌트가 게임내에서 어떻게 보일지 정확한 형태를 가늠해볼 수 있는 유일한 방법이다. Clik 확장은 이러한 작업과정을 향상시키기 위해서 런처패널을 제공한다.

4.8.1 Scale9Grid

대부분의 컴포넌트는 두번째 크기조절 방법을 사용한다. Scaleform Player에서 Scale9Grid 내부의 MovieClip 애셋은 대부분 scale9grid에 부착된다. 그러나, 플래시는 그리드가 MovieClip을 포함하고 있으면 이를 버리게 된다. 이 말은 플래시 플레이어에서 scale9grid가 작동하지 않는다 하더라고 Scaleform에는 완벽하게 작동할 것이라는 뜻이다.

MovieClip이 scale9grid를 가지고 있을 때 주의해야 할 것은 동일한 규칙으로 부요소도 크기조절 될 것이라는 것이다. scale9grid를 부요소에 추가하면 부모의 그리드를 무시하게 함으로써 정상적으로 그려지지게 된다는 것이다.

4.8.2 Constraints

Constraints(scaleform.clik.utils.Constraints)유틸리티 클래스는 크기 조절되는 컴포넌트 내부의 애셋들을 크기조절하거나 위치시키는데 도움을 준다. 이를 사용하면 개발자가 스테이지 상에 애셋을 위치시키고, 부모 컴포넌트의 끝부분에서 얼마만큼의 거리를 유지할지를 결정할 수 있다. 예를 들어서, ScrollBar 컴포넌트는 트랙과, 아래쪽 화살표 버튼을 스테이지 상의 위치에 따라서 크기조절과 위치지정을 할 것이다. Constraints는 컴포넌트에서 사용된 두가지 크기조절 방법에서 모두 작동한다.

다음 코드는 ScrollBar 애셋을 initialize() 메소드 내의 constraints 객체에 추가하고, 아래쪽 화살표 버튼을 바닥에 정렬하고, 부모크기에 따라 트랙의 크기를 조절한다. draw()메소드는 constraints를 업데이트 하는 코드를 포함하고 있으며, 등록된 요소를 업데이트 한다. 이러한 업데이트는 draw()내에서 이루어지는데, 이는 컴포넌트 무효화 다음에 호출되기 때문이다(일반적으로 컴포넌트 크기가 바뀌면 무효화가 직후에 호출된다).

```
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 컴포넌트와 데이터 셋(*Components and Data Sets*)

데이터 리스트가 필요한 컴포넌트는 dataProvider 접근법을 사용한다. dataProvider 란 데이터 저장소며, IDataProvider(scaleform.clik.interfaces.IDataProvider) 클래스내에 정의된 API를 노출하는 객체다.

dataProvider 접근법은 직접 속성에 접근하는 것이 아닌 콜백이라는 요청 모델을 사용한다. 이렇게 하면 dataProvider 가 필요한 게임엔진 내의 데이터에 접근할 수 있기 때문이다. 이렇게하면 메모리를 절약할 수 있으며, 커다란 데이터 셋을 관리하기 좋은 작은 청크로 만들수 있다.

dataProvider 를 사용하는 컴포넌트 및 CoreList (ScrollingList, TileList), OptionStepper, DropdownMenu 를 확장하는 모든 것이 포함된다.

사용예와 가장 좋은 연습(Usage and Best Practices)

프레임워크에 포함된 DataProvider (scaleform.clik.data.DataProvider) 클래스는 정적 initialize()메소드를 사용해서 액션스크립트 배열에 dataProvider 메소드를 추가한다. dataProvider 를 사용하는 컴포넌트는 자동적으로 배열을 초기화하기때문에 dataProvider 를 사용한 접근할 수 있도록 해준다. 즉, 다음 구문은 IDataProvider 에서 설명한 방법으로 완벽하게 작동하는 dataProvider 를 정적 배열로 초기화하는 것이다.

```

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

```

dataProvider 가 구현해야하는 메소드는 다음과 같다.

- *length*: 데이터 집합의 길이를 반환하는 get 함수나 속성
- *requestItemAt*: dataProvider 로부터 지정된 아이템을 요청한다. 일반적으로 OptionStepper 처럼 항상 단일 아이템을 출력하는 리스트 컴포넌트에서 사용된다.

- *requestItemRange*: 시작과 끝 인덱스로 dataProvider로부터 아이템의 범위를 요청한다. 주로 ScrollingList 처럼 하나 이상의 아티엠을 출력하는 리스트 컴포넌트에서 사용된다.
- *indexOf*: 아이템의 인덱스를 반환한다.
- *invalidate*: dataProvider가 변경되었거나 새로운 길이의 데이터가 제공되었음을 나타내는 플래그. 이 메소드는 데이터가 업데이트 되었다는 것을 컴포넌트에 공지하는 "dataChange" 이벤트를 발송해야 한다. dataProvider는 데이터 크기를 반영하는 length 속성을 제공해서 공개적으로 접근할 수 있게 해야 한다.

짧은 리스트 데이터를 필요로하는 인스턴스라면 dataProvider로 배열을 사용할 수 있다. 개발자는 데이터를 액션스크립트 2에 보관하는 것이 어플리케이션 내부적으로 저장하는 것보다 더 많은 메모리와 수행시간이 필요하다는 것을 알아야 한다. 커다란 데이터는 필요성을 기준으로 게임 엔진의 데이터를 전송 할 수 있도록 dataProvider가 ExternalInterface.call와 함께 연결될 것을 권한다.

4.10 동적 애니메이션

Scaleform CLIK는 다른 비슷한 Tween 클래스와 유사하게 동작하지만 Scaleform과 완전히 호환되는 사용자 정의 Tween 클래스(scaleform.clik.motion.Tween)를 제공합니다. CLIK Tween은 fl.transitions.easing.* 패키지에 있는 기능과 같은 표준 완화 기능을 지원합니다.

tween을 시작하려면 새 Tween()을 생성하고 지속 시간(1000분의 1초 단위), 대상 Sprite, tween에 대한 프로퍼티, tween해야 할 값 및 오브젝트를 추가 Tween 파라미터로 제공되는 프로퍼티와 함께 입력합니다. 다음의 프로퍼티/파라미터가 지원됩니다.

- *ease*: 기능 형식입니다. 완화 기능입니다.*easeParam*: 오브젝트 형식입니다. 완화 기능으로 보낼 수 있는 추가 파라미터입니다.
- *onComplete*: 기능 형식입니다. Tween이 완료되면 이 종료가 호출됩니다.
- *onChange*: 기능 형식입니다. Tween이 그 값을 업데이트하면 이 종료가 호출됩니다(모든 tick/frame).
- *data*: 오브젝트 형식입니다. Tween에 붙이는 모든 사용자 정의 데이터입니다(Tween 동작에는 어떤 영향도 주지 않음).*nextTween*: Tween 형식입니다. 다른 Tween에 연결하려면 사용합니다. 현재 Tween이 완료되면 연결된 Tween이 pause=false로 설정됩니다.
- *frameBased*: Boolean 형식입니다. 실제 시간 대신 프레임 기반 시간을 사용합니다.
- *delay*: 숫자 형식입니다. 1000분의 1초 단위로 숫자를 지정하여 tween을 연기합니다.

- *fastTransform*: Boolean입니다. 오브젝트 프로퍼티를 표시하기 위해 매트릭스 계산을 사용합니다.
- *paused*: Boolean 형식입니다. Tween을 일시 중지할지 여부입니다.

하지만 이 tween 메소드는 MovieClip마다 다중 속성을 지원하기 때문에 동시에 동일한 객체의 다른 속성들에 대한 tweening을 할 수 있다. 다음의 예는 동시에 다중 속성에 영향을 주는 tween을 어떻게 생성하는지를 보여준다.

4.10.1 사용예와 가장 좋은 연습(Usage and Best Practices)

Tween 인스턴스는 대상 MovieClip을 MovieClip의 현재 프로퍼티 값에서 기능 파라미터 목록의 다른 지정 항목으로 tween 합니다.

tween이 완료되었을 때 통보를 받으려면, Tween 생성자로 네 번째 파라미터를 보내는 것처럼 onComplete 프로퍼티에서 오브젝트로 기능 참조를 생성하고 추가합니다. 이 콜백은 호출한 Tween이 완료되고 파라미터 하나를 보내면 호출됩니다.

```
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 레이아웃 프레임워크

CLIK 레이아웃 프레임워크는 여러 요소를 포함하고 여러 해상도로 쉽게 조정되는 동적 레이아웃을 생성하도록 개발자를 지원하기 위해 설계되었습니다. 다소 단순하지만, Clik 레이아웃 프레임워크는 여러 해상도 및 플랫폼을 위한 UI를 생성할 뿐만 아니라 Clik 내에서 사용자 정의 레이아웃 시스템 개발을 위한 기초를 제공하여 개발자를 지원합니다.

참고: Scaleform v4.0.14 는 CLIK AS3 레이아웃 프레임워크의 초기 릴리스이며 클래스와 API 는 변경될 수 있습니다. CLIK AS3 레이아웃 프레임워크와 관련된 질문, 문제, 제안 또는 의견이 있는 경우, Scaleform 개발자 센터를 통해 티켓을 여시기 바랍니다(Scaleform 지원은 개발자 센터에 위치한 온라인 티켓팅 시스템을 통해 제공됩니다. 이 시스템에서는 라이센스 취득 업체를 위한 전체 설명서, 데모, 비디오 및 포럼도 제공합니다.).

다음의 두 클래스가 CLIK 레이아웃 프레임워크의 핵심을 이룹니다.

- **scaleform.clik.layout.Layout:** 같은 부모 내에서 .layoutData 프로퍼티를 올바르게 구현하는 Sprite 의 레이아웃을 관리하는 레이아웃입니다.
- **scaleform.clik.layout.LayoutData:** 특정 Sprite 이 배치되는 방법에 대한 정보를 포함하는 데이터 구조입니다. 이 데이터는 관련 레이아웃 인스턴스에 의해 읽히고 레이아웃을 생성하는 데 사용됩니다.

CLIK 내에서 새 레이아웃을 생성하는 방법은 여러 가지입니다. 새 레이아웃을 생성하는 가장 간단한 방법은 다음과 같습니다.

1. Sprite/MovieClip 을 생성하고 스테이지의 DisplayObjectContainer 에 추가합니다.
2. LayoutData 클래스의 새 인스턴스와 함께 이 Sprite/MovieClip 에 대한 .layoutData 프로퍼티를 정의합니다.
3. 원하는 대로 각 Sprite/MovieClip 에 대한 LayoutData 오브젝트를 채웁니다.
4. 레이아웃 클래스의 새 인스턴스를 생성합니다. 1 단계에서 이전에 Sprite/MovieClip 에 의해 사용된 동일한 DisplayObjectContainer 에 추가합니다.

이 시나리오에서 레이아웃 인스턴스의 프로퍼티가 수정되지 않은 경우, 위치를 잡기 위해 사각형(x, y, 넓이 및 높이)으로 부모의 넓이 및 높이를 사용하여 .layoutData 프로퍼티를 정의한 부모 내에서 레이아웃이 모든 Sprite 과 MovieClip 을 관리합니다.

SDK 의 *Scaleform\#Resources\#AS3\#CLIK\#demos\#* 디렉터리에 레이아웃 프레임워크의 데모가 두 개 포함되어 있습니다. 첫 번째 데모인 Layout_Main 은 Scaleform SDK Browser 를 통해 열릴 수 있는 CLIK Component Browser 에서 사용할 수 있습니다. Layout_Main 은 크기 조절 가능한 MovieClip 내에서의 간단한 레이아웃 생성을 보여줍니다. 다른 데모인 Layout_FullScreen_Demo 는 FxPlayer 의 단일 실행형 인스턴스에서 실행되도록 설계되어 스테이지의 전체 크기로 사용할 수 있습니다.

Layout_FullScreen_Demo 는 사용자가 여러 해상도에 대해 전체 화면 콘텐츠를 배치할 수 있는 방법과 레이아웃 시스템이 다양한 파라미터를 처리하는 방법을 보여줍니다.

4.11.1 레이아웃

레이아웃은 크기 및/또는 비율이 변경될 수 있는 컨테스트 내에서 레이아웃 요소로 사용될 수 있는 구성 요소입니다. 레이아웃은 같은 부모 내에 존재하는 Sprite(및 Sprite 의 하위 클래스)만 관리할 수 있습니다. 즉 레이아웃 인스턴스 자체의 자식으로 요소를 추가할 수 없지만 해당 레이아웃 인스턴스의 동일한 레벨에서는 추가할 수 있습니다. 예를 들어 동일한 레벨(myWindow.myMovieClip 및 myWindow.layoutInstance)에서 자식 두 개를 가지고 있다면 myMovieClip 관리를 위한 layoutInstance 가 허용됩니다.

레이아웃은 LayoutData 클래스의 인스턴스와 함께 .layoutData 프로퍼티를 정의한 Sprite 만 관리합니다. 각각 고유한 .identifier 프로퍼티 세트를 가지고 또한 동일한 레벨의 Sprite 내에 모든 LayoutData 이 자체 .layoutIdentifier 프로퍼티를 정의하도록 제공하는 동일한 부모 내에 여러 레이아웃이 존재할 수 있습니다.

스테이지에 추가될 때 레이아웃은 부모 내에서 모든 자식을 찾고 .layoutData 를 정의했는지 확인합니다. 그때부터 레이아웃은 부모에 추가된 새 자식을 추적하고, 스테이지에 추가하기 전에 .layoutData 을 정의한 경우 관리되는 Sprite/MovieClip 목록에 자식을 추가합니다. 관리되는 Sprite/MovieClip 오프셋 및 배치되는 순서는 스테이지에 처음 추가될 때만 업데이트됩니다. 사용자가 오프셋을 업데이트된 위치에서 재계산하거나 목록을 재정렬하려는 이벤트에서 다음 기능을 사용할 수 있습니다.

public function reset():void

관리되는 Sprite 목록 제거, LayoutData 로 새 Sprite 검색 및 맨 처음부터 오프셋 재계산/새 Sprite 재이동을 통해 레이아웃을 재설정합니다.

public function resortManagedSprites():void

레이아웃이 적용되는 순서를 정의하는 `layoutData.layoutIndex` 프로퍼티를 통해 관리되는 Sprite 목록을 정렬합니다. 레이아웃의 초기 설정 후에 `layoutIndex` 가 변경되면 호출됩니다.

사용자는 레이아웃 클래스를 Flash 파일 라이브러리의 심볼에 추가하여 디자이너가 디자인을 할 때 스테이지에 레이아웃을 배치하도록 허용할 수 있습니다. 레이아웃이 심볼에 연결되지 않은 경우(대신 넓이와 높이가 0 인 코드를 통해 인스턴스화), 레이아웃의 크기는 부모의 크기로 기본 설정됩니다. 레이아웃이 관리되는 요소를 배치하기 위해 사용하는 영역을 정의하는 레이아웃의 `.rect` 프로퍼티 설정에 의해 변경될 수 있습니다.

이상적으로 다른 `Sprite/MovieClip` 에 대한 모든 `LayoutData` 가 이미 올바르게 정의되었는지 확인하려면 레이아웃은 부모에게 마지막 요소로 추가되어야 합니다.

4.11.1.1 퍼블릭 프로퍼티

rect	레이아웃의 "크기"입니다. 이 프로퍼티의 x, y, 넓이 및 높이에 따라 요소가 배치됩니다. 레이아웃의 넓이가 <code>width != 0</code> 인 경우(심볼 연결 및 스테이지 배치), <code>MovieClip</code> 의 x, y, 넓이 및 높이를 사용합니다. 레이아웃의 넓이가 <code>width == 0</code> 인 경우(심볼 지원 없는 <code>addChild()</code>), 부모의 x, y, 넓이 및 높이를 사용합니다. 또한 <code>.rect</code> 프로퍼티를 사용하여 사용자 정의 사각형을 할당할 수 있습니다.
tiedToStageSize	이 레이아웃의 크기가 항상 스테이지 크기와 일치하도록 업데이트되어야 하면 <code>true</code> 이고, 그렇지 않으면 <code>false</code> 입니다.
tiedToParent	이 레이아웃의 크기가 항상 부모의 크기와 일치하도록 업데이트되어야 하면 <code>true</code> 이고, 그렇지 않으면 <code>false</code> 입니다.
hidden	이 레이아웃을 실행 시 숨겨야 하면 <code>true</code> 이고, 그렇지 않으면 <code>false</code> 입니다. 보이는 배경 또는 실행 시 즉시 <code>visible = false</code> 로 설정될 플레이스홀더 이미지를 레이아웃이 가지도록 허용합니다.

4.11.2 LayoutData

특정 Sprite 에 대한 레이아웃을 정의하는 데이터입니다. 유효한 scaleform.clik.layout.Layout 인스턴스와 함께 사용되어야 합니다.

4.11.2. 1 퍼블릭 프로퍼티

alignH	이 Sprite 에 대한 수평 정렬입니다. 유효한 값: LayoutMode.ALIGN_NONE, LayoutMode.ALIGN_LEFT, LayoutMode.ALIGN_RIGHT.
alignV	이 Sprite 에 대한 수직 정렬입니다. 유효한 값: TextFieldAutoSize.NONE 에 대한 LayoutMode.ALIGN_NONE, LayoutMode.TOP, LayoutMode.BOTTOM 프로퍼티는 크기를 변경하지 않고 그대로 놔둡니다.
offsetH	레이아웃 또는 relativeToH Sprite 의 모서리에서 수평 오프셋입니다. 변경되지 않은 경우(-1), 레이아웃은 디자이너가 디자인 중에 정의한 것처럼 스테이지의 Layout(Sprite)에서 원본 수평 오프셋으로 자동 설정합니다.
offsetV	레이아웃 또는 relativeToV Sprite 의 모서리에서 수직 오프셋입니다. 변경되지 않은 경우(-1), 레이아웃은 디자이너가 디자인 중에 정의한 것처럼 스테이지의 Layout(Sprite)에서 원본 수직 오프셋으로 자동 설정합니다.
offsetHashH	다양한 영상비에 대한 사용자 정의 수평 오프셋의 해시 표입니다. 레이아웃이 스테이지 크기에 연결되면 이 값이 질의됩니다(Layout.tiedToStageSize == true). 예를 들어 레이아웃이 스테이지 크기에 연결되고 영상비가 현재 4:3 인 경우 "LayoutData.offsetHashH[LayoutData.ASPECT_RATIO_4_3] = 70;"은 이 Sprite 이 70px 의 수평 오프셋을 사용하도록 합니다.
offsetHashV	다양한 영상비에 대한 사용자 정의 수직 오프셋의 해시 표입니다. 레이아웃이 스테이지 크기에 연결되면 이 값이

	<p>질의됩니다(Layout.tiedToStageSize == true). 예를 들어 레이아웃이 스테이지 크기에 연결되고 영상비가 현재 4:3 인 경우 "LayoutData.offsetHashV[LayoutData.ASPECT_RATIO_4_3] = 70;"은 이 Sprite 이 70px 의 수직 오프셋을 사용하도록 합니다.</p>
relativeToH	이 Sprite 이 수평으로 상대적이어야 하는 Sprite 의 인스턴스 이름입니다. null 로 남는 경우 Sprite 은 레이아웃에 상대적으로 정렬됩니다.
relativeToV	이 Sprite 이 수직으로 상대적이어야 하는 Sprite 의 인스턴스 이름입니다. null 로 남는 경우 Sprite 은 레이아웃에 상대적으로 정렬됩니다.
layoutIndex	이 Sprite 이 같은 레이아웃에서 다른 Sprite 에 상대적으로 배치되어야 하는 순서를 관리하는 인덱스입니다. Sprite 의 레이아웃이 이 Sprite 전에 업데이트되도록 relativeToH 또는 relativeToV 를 사용하는 경우에 설정해야 합니다. layoutIndex 가 변경되지 않은 경우(-1), 목록의 끝에 임의로 추가됩니다.
layoutIdentifier	여러 레이아웃이 단일 DisplayObjectContainer 내에 존재하는 경우 이 LayoutData 오브젝트가 연결되어야 하는 레이아웃을 정의하는 문자열입니다. 변경되지 않은 경우(null), 같은 부모 내에서 모든 레이아웃에 의해 자동으로 관리됩니다. 설정하는 경우 이 프로퍼티는 레이아웃 인스턴스의 식별자 프로퍼티와 일치해야 합니다.

4.12 팝업 지원

스케일폼 CLIK 은 PopUpManager (scaleform.clik.managers.PopUpManager) 클래스를 통해서 대화상자나 툴팁 같은 팝업지원을 제공한다.

4.12.1 사용예와 가장 좋은 연습(Usage and Best Practices)

PopUpManager 는 팝업생성과 관리를 지원하는 몇가지 정정 메소드를 제공한다. 첫 번째 파라미터는 고유한 연동 ID 에 대한 문자열입니다. 두 번째 파라미터인 initProperties 는 생성할 때 새 팝업으로 복사될 프로퍼티를 가진 오브젝트를 제공하도록 허용합니다. 세 번째와 네 번째 파라미터는 팝업의

부모와 관련된 새 팝업에 대해 각각의 x 및 y 좌표를 정의합니다. relativeTo 파라미터인 Sprite 형식은 다른 Sprite 과 관련된 팝업을 위치시키는 데 사용될 수 있습니다.

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

Scaleform 확장자인 topmostLevel 은 팝업을 스테이지 상 모든 컨텐츠 위에 출력할 수 있도록 해준다. 이러한 계획은 root 레벨에 있는 팝업을 생성하는 것 대신으로 사용되었는데 이는 라이브러리 스코프에 관련 된 문제때문이다. 만약 부모 SWF 내에 로드된 자식 SWF 가 부모 context 에 있는 경로에 자체 라이브러리에서 지정된 심볼의 인스턴스를 생성하려고 하면 심볼 검색 에러가 발생하는데 이는 부모가 자식의 라이브러리에 액세스할 수 없기 때문이다. 불행이도 이 문제에 대하여 깔끔한 차선책은 없으며 topmostLevel 안 만이 최고의 대안이다.

topmostLevel 은 팝업이 아닌 것에도 응용 가능하며 스테이지 안에 있는 이러한 요소의 z-ordering 도 유지된다. 따라서, 팝업 위에 커스텀 커서를 그리기 위해서는 최고의 깊이나 레벨에 커서를 생성하고 topmostLevel 속성을 true 로 설정하면 된다.

4.13 끌어 놓기

DragManager (scaleform.clik.managers.DragManager)클래스는 드래그 작용에 대한 초기화와 관리 지원을 제공한다. 이 클래스는 단일체(싱글톤 singleton)와 비슷하게 작동하며 단일 객체 접근에 대한 인스턴스 속성을 제공한다. 이 객체를 사용하면 개발자가 드래그 기능을 시작하고 종료시킬 수 있다.

드래그를 시작하려면 DragEvent.DRAG_STATE 를 새 드래그와 관련된 프로퍼티와 함께 보냅니다. DragManager.handleStartDragEvent() 는 이벤트를 수신하고 이벤트의 프로퍼티에 기반한 드래그를 시작합니다. 그러면 드래그는 DragManager.handleEndDragEvent() 가 호출되도록 하는 MouseEvent.MOUSE_UP 이벤트가 수신되기 전까지 DragManager 에 의해 관리됩니다.

4.13.1 사용예와 가장 좋은 연습(Usage and Best Practices)



그림 64: 작동중인 드래그 데모 DragDemo

DragManager 는 UI 를 통해 사용할 수 있는 IDragSlot 과 효율적으로 통신하기 위한 기능을 정의하는 IDragSlot 인터페이스를 사용합니다. DragManager 는 드래그 관리만 수행한다. 진정한 드래그 & 드롭을 제공하기 위해서 DragManager 를 사용한 컴포넌트는 만드는 것은 개발자에게 달려있다. Scaleform CLIK 에는 DragSlot 이라는 샘플 IDragSlot 구현이 포함됩니다(scaleform.clik.controls.DragSlot). DragSlot 은 UIComponent 을 확장했기 때문에 CLIK 컴포넌트로 분류된다. 그리고, DragManager 와 비교해서 독특한 기능을 몇 가지 지원한다. DragSlot 은 드래그될 MovieClip/Sprite/Bitmap 을 포함하도록 설계되었습니다. 하지만, DragSlot 은 또한 버튼으로도 사용될 수 있는 많은 버튼 동작을 모방합니다.

DragSlot 은 드래그 타입 개념을 가지고 있다. 드래그 타입은 DragSlot 마다 드롭기능을 허가할지 불허할지 설정가능하다. 이러한 드래그 타입과 다르게 DragSlot 은 DragManager 에 dragBegin 과 dragEnd 이벤트 리스너를 설치한다. 이렇게 해서 DragTarget 이 드롭기능을 허가하건 말건 비쥬얼하게 보여질 수 있는 것이다.

결국 DragSlot 은 단지 IDragSlot 의 샘플 구현이며, 데이터 백엔드와 보다 밀접하게 통신하는 서브 클래스에 의해 대체되도록 많은 기능이 종료되었습니다. 다양한 기능을 가진 DragSlot 하위 클래스의 샘플에 대해서는, UI 를 통해 드래그 가능한 콘텐츠에 대한 기본 클래스로 DragSlot 을 사용하는 MMO UI 키트을 확인하십시오.

5 사용예

이번 장에서는 스케일폼 CLIK 컴포넌트를 사용한 사용예를 보여줄 것이다.

5.1 기초

다음 예제는 간단하지만 스케일폼 Clik 프레임워크를 사용해서 일반적인 업무를 수행하는 것이 얼마나 쉬운지를 보여준다.

5.1.1 2 개의 textField 를 가진 ListItem 렌더러



그림 65: 2 개의 레이블을 포함한 리스트 아이템을 보여주는 ScrollingList

ScrollingList 컴포넌트는 열(row)의 내용을 출력할 때 기본적으로 ListItemRenderer 를 사용한다.

하지만 ListItemRenderer 는 단일 textField 만 지원한다. 하나 이상의 textField 를 가지는 리스트 아이템이 필요한 경우가 많다. 혹은 아이콘처럼 textField 가 아닌 것을 출력해야 할 경우도 있다. 이 예제는 2 개의 textField 를 리스트 아이템에 추가하는지를 보여줄 것이다.

먼저 요구사항을 정의하도록 하자. 목적은 2 개의 textField 를 지원하는 커스텀 ListItemRenderer 를 만드는 것이다. 커스텀 ListItemRenderer 는 ScrollingList 와 호환되어야 한다. 현재 목적은 아이템마다 2 개의 textField 를 갖도록 하는 것이기 때문에 데이터도 하나의 문자열 이상이 될 것이다. 다음과 같은 dataProvider 를 사용하도록 하자.

```
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"}];
```

데이터 제공자는 2 개의 속성(fname 과 lname)을 가진 객체를 포함하고 있다. 이 2 개의 리스트 아이템 textField 에 2 개의 속성을 출력될 것이다.

기본 ListItemRenderer 가 하나의 textField 만 지원하므로, 2 개의 textField 를 지원하는 기능을 추가해야 한다. 가장 쉬운 방법은 ListItemRenderer 클래스의 서브클래스를 만드는 것이다. 다음 소스코드는 MyItemRenderer 라는 클래스인데, 이 클래스는 ListItemRenderer 를 상속받아서 2 개의 textField 지원에 대한 기본 기능을 추가하고 있다. (이 코드를 작동시키려는 FLA 와 동일한 디렉토리에 MyItemRenderer.as 라는 이름의 파일로 저장한다)

```
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 의 setData 와 updateAfterStateChange 메소드 서브클래스에서 재지정(override)되었다. setData 는 리스트 아이템이 아이템 데이터를 컨테이너 리스트 컴포넌트(ScrollingList, TileList, 등)로부터 받았을 때 항상 호출된다. ListItemRenderer 에서 이 메소드는 하나의 textField 값을 설정한다. 그러나, MyItemRenderer 에서는 두개 모두의 textField 값을 설정하고, 또한 아이템 객체의 참조값을 내부적으로 저장한다. 이 아이템 객체는 updateAfterStateChange 에서

재사용 되는데, 이 메소드는 ListItemRenderer 의 상태가 변경될때마다 호출된다. 이 상태변화에 따라서 textField 값이 자신들의 값으로 갱신 될 필요가 있을 수 있다.

이렇게해서 복합 리스트 아이템용 dataProvider 와 이들 리스트 아이템의 렌더링을 지원하는 ListItemRenderer 클래스가 정의 되었다. 이제, 이 새로운 ListItemRenderer 클래스를 지원하는 심볼이 생성되어야 한다. 이를 처리하는 가장 빠른 방법은 ListItemRenderer 심볼을 수정해서 'textField1' 과 'textField2'라는 2 개의 textField 를 갖도록 하는 것이다. 다음으로, 이 심볼의 ID 와 클래스가 MyItemRenderer 로 변경되어야 한다. MyItemRenderer 를 사용하려면 리스트 인스턴스의 itemRenderer 점검가능 속성을 'ListItemRenderer' 에서 'MyItemRenderer'로 변경한다.

FLA 를 실행하자. 2 개의 레이블을 출력하는 리스트 요소가 포함된 형태로 보일 것이다. fname 과 lname 속성은 dataProvider 에서 설정한 값일 것이다.

5.1.2 픽셀별 스크롤 뷰

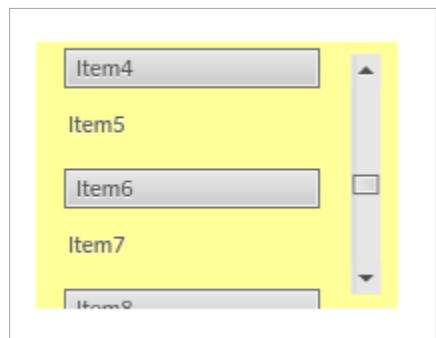


그림 66: CLIK 요소를 컨텐츠로 포함하고 있는 픽셀별 스크롤 뷰

픽셀별 스크롤은 복잡한 사용자 인터페이스에서 흔하게 사용된다. 이번 예제는 CLIK ScrollBar 컴포넌트를 사용해서 어떻게 손쉽게 이 기능을 구현하는지에 대하여 보여줄 것이다.

일단, 스크롤 뷰용 새로운 심볼을 만든다. 이렇게 하면 컨텐츠 옵셋을 계산할 때 필요한 수학연산을 단순화 시키는 유용한 컨테이너를 제공해 준다. 이 스크롤 뷰 컨테이너 내에 위에서부터 아래쪽 순서로 레이어를 설정한다. 이들 레이어가 불필요하기는 하지만 깔끔한 처리를 위해서는 사용할 것을 추천한다.

- **actions**: 스크롤 뷰가 작동하도록 하는 액션스크립트 코드를 포함하고 있다.
- **scrollbar**: CLIK ScrollBar 의 인스턴스를 포함하고 있다. 이 인스턴스를 'sb'라고 부르자.

- **mask**: 사각형 형태나 MovieClip으로 정의된 마스크 레이어. 레이어 속성도 마스크(mask)로 설정한다.
- **content**: 스크롤될 컨텐츠를 포함하고 있다.
- **background**: 마스크 되지 않은 영역을 하이라이트 처리되게 할 배경을 포함하고 있는 옵션 레이어

컨텐츠 레이어에 적절한 요소를 추가하고, 이들 모두를 포함하는 심볼을 생성한다. 모든 컨텐츠를 포함하는 심볼을 생성함으로써 스크롤을 쉽게 처리할 수 있다. 이 컨텐츠 인스턴스를 'content'라 하고, y 값(y-value)를 0으로 하자. 이렇게 하면 스크롤 로직이 다른 옵셋에 따라서 달라지지 않도록 해준다. 하지만 이들 옵셋들이 스크롤 뷰의 복잡도에 따라서 필요할 수도 있다.

이제 간단한 scrollview 를 생성하는데 필요한 구조와 서로 가로채는데 필요한 명명된 요소가 정의되었다. 다음 액션스크립트 코드를 *code* 레이어의 첫번째 프레임에 넣는다.

```
// 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;
}
```

Scrollbar 가 다른 컴포넌트에 연결되어 있지 않기 때문에 수동으로 설정해줘야 한다. setScrollProperties 메소드를 사용해서 포지션에 따른 스크롤을 만들고 컨텐츠를 완전히 출력하기 위해 최대, 최소값 설정 등을 한다. 이 경우의 scrollbar 포지션은 픽셀이다. 파일을 실행해보자. Scrollview 가 scrollbar 와 상호작용되어 변경될 것이다.

6 자주 묻는 질문

1. Flash Studio에서 게시할 때 다음과 같은 오류 메시지를 수신했습니다.

Quote:

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

클래스의 요소를 두 번 정의하면 이 오류를 수신합니다. 게시할 때 ActionScript 정의 및 .FLA에 의해 자동으로 생성된 정의 사이에서 충돌을 일으키는 "Automatically Declare Stage Instances"를 비활성화하도록 .FLA가 설정되지 않은 경우 이 오류 메시지가 가장 자주 발생합니다.

ActionScript 클래스에서 강하게 입력된 자식을 참조하기 위해 Stage 요소의 명시적인 정의가 일반적으로 필요하기 때문에(권장 사항) CLIK 아키텍처에서 올바른 기능 작동을 위해 모든 CLIK 구성 요소에 대해 이 설정을 비활성화하는 것이 필요합니다.

File -> Publish Settings -> "Flash" tab -> Actionsript 3.0 Settings을 열어서 .FLA에 대한 "Automatically Declare Stage Instances"를 비활성화할 수 있으며 "Automatically declare stage instances" 체크를 해제해야 합니다. 이 설정은 전역 Flash Studio 설정이 아니며 특정 .FLA에만 저장됩니다.

2. 다른 .FLA 라이브러리에서 CLIK 구성 요소를 가져오는 중입니다. 스테이지에 구성 요소의 인스턴스를 두고 조사 가능한 프로퍼티를 변경하자 다음과 같은 오류 메시지를 수신했습니다.

Code:

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

이 오류는 구성 요소의 부적절한 선언 때문에 가장 일반적으로 발생합니다. 일반적으로 다음 단계를 사용하여 오류를 해결할 수 있습니다.

- 구성 요소의 인스턴스 이름이 유효한지 확인합니다.

- 구성 요소가 ActionScript에서 정의되었는지 확인합니다.
- ActionScript 정의가 정확한 클래스를 참조하는지 확인합니다. 일부 경우, 구성 요소의 보내기 이름이 아닐 수 있습니다. 대신 ActionScript 클래스 이름이 사용되어야 합니다. 예를 들어, MyButtonSymbol이 가져온 심볼의 보내기 이름이고 myButton이 구성 요소의 인스턴스 이름인 myButton:MyButtonSymbol 대신 myButton:Button (scaleform.clik.controls.)입니다.

3. 라이브러리에 같은 기본 클래스에서 얻은 두 개의 구성 요소가 있습니다. .SWF 파일을 게시할 때 다음과 같은 오류를 수신했습니다.

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.

이 오류는 일반적으로 기본 클래스 이름과 동일한 클래스 이름을 가진 심볼 때문에 발생합니다. 예를 들어 "Button" 클래스와 "scaleform.clik.controls.Button" 기본 클래스를 가진 심볼입니다. 이로 인해 Symbol의 타임라인 정의가 기본 클래스와 병합됩니다. 따라서 같은 기본 클래스를 공유하는 다른 심볼은 의도치 않은 동작을 초래하여 이 타임라인 정의를 상속합니다.

이 문제를 해결하려면 여러 심볼과 연관된 기본 클래스가 있는지 확인한 다음 심볼의 클래스 이름이 기본 클래스의 이름과 일치하지 않도록 합니다. 같은 이름을 공유하는 심볼의 클래스를 변경하면 문제가 해결됩니다.

7 잠재적인 위험

1. 다른 심볼이 해당 기본 클래스를 확장하지 않는 한 심볼 클래스의 이름을 기본 클래스와 동일하게 지정하지 않습니다. 심볼이 기본 클래스와 동일한 클래스 이름을 가지면 타임라인 정의가 기본 클래스와 병합되고 해당 기본 클래스의 다른 하위 클래스에 문제가 발생할 수 있습니다.

클래스의 다른 서브 클래스가 존재하지 않으면 기본 클래스를 클래스로 사용하고 심볼에 대한 기본 클래스를 제공하지 않을 것을 권장합니다. 이렇게 하면 기본 클래스의 새 인스턴스를 인스턴스화할 때 기본 클래스와 심볼을 단단하게 결합하고, 심볼은 `getDefinitionByName()`과 같은 검색을 사용하지 않고 자동으로 결합됩니다.