Autodesk® Scaleform®

Getting Started with CLIK Buttons

This document explains how to get up and running quickly with Scaleform® Common Lightweight Interface Kit (CLIK™)'s button component.

Authors:	Matthew Doyle
Version:	2.0
Last Edited:	August 19, 2010

Autodesk® GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.4

Disclaimer

How to Contact Autodesk Scaleform:

Document	Getting Started with CLIK Buttons
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. Overview

Scaleform's Common Lightweight Interface Kit (CLIK) allows developers to rapidly prototype efficient front-end menu interfaces for games, and then take those prototypes to completion with a minimum of time and effort. It also allows users to rapidly iterate on their designs. The kit includes over 15 individual widgets or components, including buttons, sliders, progress bars, dropdown menus, text areas, and scrolling lists. This document will focus on the button component, which is core to the rest of the CLIK components. There are four key button types: Button, ToggleButton, AnimatedButton, and AnimatedToggleButton. The latter three are specialized variants of the standard Button component. Learning how and when to use this component and its variants is the first step to understanding the rest of the CLIK components.

It is recommended that the Getting Started with CLIK document be read before reading this document, as many of the concepts discussed in this document are introduced in it.

NOTE that there are different ways of doing the same thing in Flash; developers may be able to find and use alternative workflows that suit them better than the approach adopted in this tutorial.

2. Button

This is the core component or widget that forms the basis of most CLIK components, including sliders, progress bars, and scroll bars. Many of the other components inherit functionality from the base Button component, and most user interface (UI) work will be done using or extending the Button component. The core Button component is a non-animated, bare bones button. It has the core behaviors of responding to mouseover events, and mouse clicks, keyboard presses, or game controller presses.

A few possible use cases for the core Button component might be:

- Menu buttons
    - Ok/Cancel/Apply
    - Screen select
- In-game console control buttons
- Mini-game buttons

This Button component and the other variants described in the next sections use the same ActionScript™ 2 (AS) class found in the Button.as file. It has all of the code necessary to make Button work. While it is not necessary to review this file in order to make use of the Button component, it is recommended.

2.1 Getting Started

1. To begin, open the CLIK_Components.fla file in Adobe® Flash®. On Windows, this is located at *C:/Program Files/Scaleform/GFx SDK 4.4/Resources/AS2/CLIK/components/ CLIK_Components.fla.*
   On Mac, this component file can be found in *scaleform_gfx_4.4_macos/Resources/AS2/CLIK/components* in the installed Scaleform SDK directory.

2. *Hold down the (CTRL) key, then left mo*use click on the following components in the *Library* pane:
    a. *Button*
    b. *ToggleButton*
    c. *AnimatedButton*
    d. *AnimatedToggleButton*

3. Release the (CTRL) key and then right click on *Button*. All four button variants should be highlighted.

4. Select *Copy*.

5. Create a new Adobe® Flash® file (AS2) by pressing (CTRL+N) or selecting *File* then *New* from the top Flash menu.

6. Right click on the blank area inside the *Library* pane of the new Flash file.

7. Select *Paste* to paste the four button variants into the *Library.*

8. Select *Button* in the *Library* pane by left mouse clicking on it.

9. Click and drag the *Button* component from the *Library* to somewhere on the *Stage*.

10. Repeat steps 8–9 for the remaining three button variants: *ToggleButton, AnimatedButton* and *AnimatedToggleButton*.

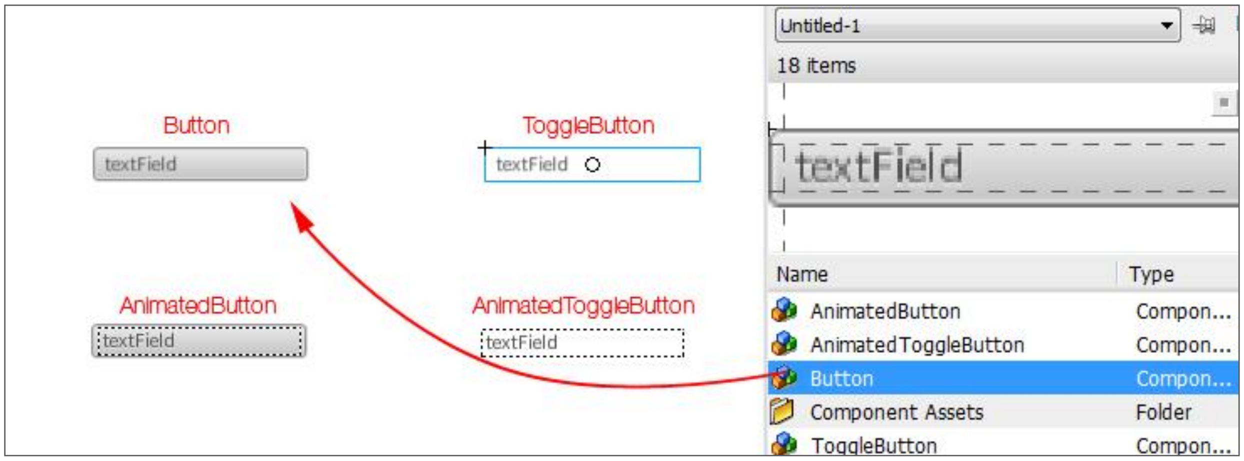


Figure 1: Click and drag the four button variants to the stage.

2.2 Button Layers

The basic Button component and the other three variants—ToggleButton, AnimatedButton and AnimatedToggleButton—are each made up of five layers. To see the Button component's layers, double click the *button* on the *stage*. Be sure to double click the standard button and not one of the variants to follow the immediate sections.

The layers are:

- *actions:* used to place AS statements
- *labels*: used as a visual cue to indicate each state of a button (described in section 2.3)
- *textField*: used for the text a button displays at runtime
- *button*: used to place the graphical image of the button at various states

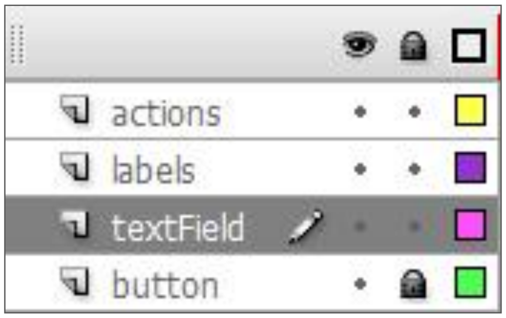


Figure 2: The button layers.

These layers, as they are, are not necessary. They may be repositioned, renamed, merged, or removed completely in favor of other layers. A button may be composed of any layers the UI artist chooses to create; however, we have provided these layers as a good starting point for organization. Keep in mind however, that a button will not function properly if the AS in the *actions* layer and the labeled keyframes of the *labels* layer are not present and in the proper positions. Likewise, graphical representation of each state must be included at the proper positions relative to the *labels* keyframes on the timeline somewhere. Therefore, it is recommended that the UI artist keep the layers as is until a better understanding of how the button component works has been achieved.

2.3 Button States

Each CLIK component is composed of what are referred to as states, or in the case of animated buttons, transitions (transitions will be explained in the animated buttons section). A state can be set via user interaction or code. These states include visual representations, which may or may not be different from other states of the component. The standard button has only four states. Each state's location is indicated by keyframes on the *Labels* layer of the button's timeline, and the corresponding graphics for that state live on the *States* and *Border* layers at the same spot.

The core button states are:

- *up*
- *over*
- *down*
- *disabled*

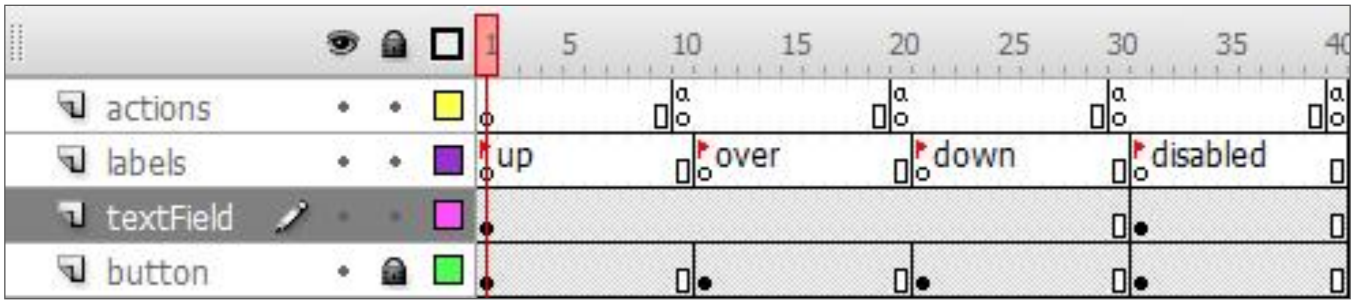


Figure 3: The button timeline with states clearly labeled on the Labels layer.

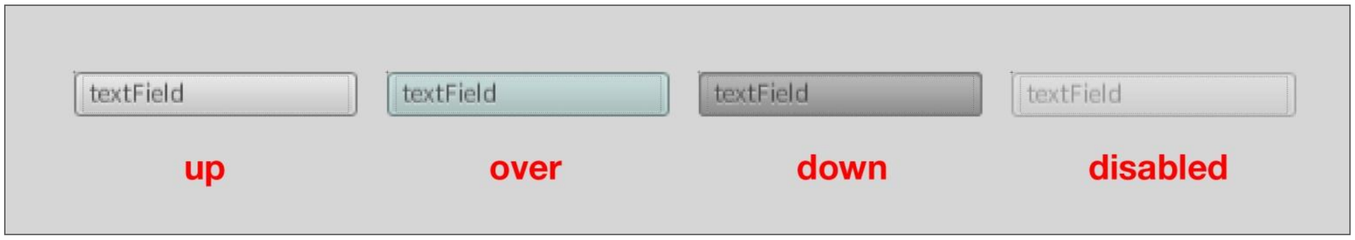


Figure 4: The button states' graphical wireframe representations side by side.

2.3.1 Up

The *up* state is the normal “at rest” state of the button. It is the state in which no user interaction has occurred with the button and it is the state that the button will display by default.

Occurs: Default state of a button; no interaction has occurred.
Usage: Represents a button in its normal state.

2.3.2 Over

The *over* state is the highlighted state of the button. It occurs when the user moves the mouse cursor over the button. Moving the cursor away from the button will return the button to the *up* state.

Occurs: When the user moves the mouse cursor over the button or when the button is focused via mouse press, keyboard arrow keys or game controller directional pads/analog sticks. Focus is described in section 2.3.3.1.
Usage: Represents a button that the mouse is currently over or has focus.

2.3.3 Down

The *down* state occurs when the user presses the button via a left mouse click, the (Enter) key on the keyboard or the appropriate button on a game controller. The button component will remain in the *down* state for as long as the mouse button, (Enter) key or game controller button is pressed. The button will return to the *over* state once released.

By default, the button will receive focus when the mouse button is pressed over it. When focused, the button will maintain its *over* state instead of the normal *up* state while “at rest,” even if the mouse is no longer over the button. The *over* state will be the button’s default state until focus is transferred elsewhere by selecting another component via mouse press, keyboard arrows or game controller. If the user moves the mouse cursor away from the button while the mouse button is still pressed, the button will revert to its *over* state and maintain focus; focus is transferred on mouse press.

Occurs: When the user presses the left mouse button while over the button component, presses the (Enter) key or presses the appropriate game controller button.
Usage: Represents a button that is being clicked.

2.3.3.1 Focus

A focused component will be the target for all keyboard and controller events. This means when a button is focused, pressing the (Enter) key will affect only that button and no other component. Focus can be transferred by pressing the left mouse button or using the keyboard arrow keys to move the focus to a neighboring component. If a game controller's directional pad and/or analog stick are mapped to the keyboard arrow keys, then they will also move the focus. Focus can also be transferred via AS code.

2.3.4 Disabled

The *disabled* state represents a button that has been disabled. A disabled button cannot be pressed or focused, and does not react to mouseover events. A button may be disabled via the *Component Inspector Parameters* tab, or it may be disabled via ActionScript using the following code:

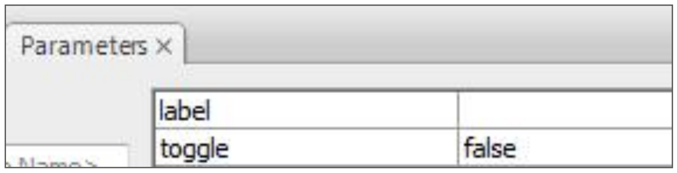
```
buttonInstanceName.disabled = true;
```

Occurs: Set via ActionScript or the *Parameters* tab.
Usage: Used to represent a button that is currently disabled and thus unusable.

2.4 Button Parameters

The Button component has a list of parameters, found in the *Parameters* tab and the *Component Inspector*, that are designed to allow a UI artist the ability to set some basic button properties without modifying the code. These parameters are:

- *disabled*: set to true to disable a button
- *disableFocus*: set to true to disallow a button to receive focus
- *disableConstraints*: set to true to disable counter scaling of the label
- *label*: enter the text label here that the button will display at runtime
- *toggle*: set to true to allow a button to toggle on or off
- *visible*: set to false to hide a button at runtime



label	
toggle	false

Figure 5: The Button component parameters in CS3 (Parameters Tab).

2.4.1 Inspectables

Parameters can be added to the list by modifying the Button.as file. This is done by adding an inspectable statement above a member variable or getter/setter function that has a modifiable property such as a Boolean (e.g., true or false). For instance, the following inspectable statement was added directly above the block of code responsible for setting a button's label, thus allowing the UI artist to set the button label via the *Parameters* panel within Flash, rather than via code:

```
[Inspectable(name="label", defaultValue="")]
```

Directly above the toggle variable in Button.as, is the following inspectable statement, allowing the UI artist to set toggle to true or false via the *Parameters* tab:

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

3. Toggle Button

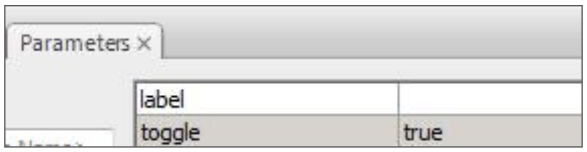
Toggle Buttons add checkbox style functionality to the standard button. This allows for the creation of buttons which can have a selected state and an unselected state. A Toggle Button "toggles" from selected to unselected— on to off (and vice versa , off to on)—each time a user clicks on it. It may also be toggled via AS.

A few possible use cases for the Toggle Button component might be:

- Menu buttons
    - Enable/disable bloom visual effect
    - Mute/unmute audio
    - High quality lighting on/off
    - Auto save on/off
- In-game console controls that require an on/off state
- Mini-game buttons that require an on/off state

The Toggle Button uses the same AS class as the standard button. The only difference is that the *toggle* property has been set to "true." Since the AS class inherently supports Toggle Buttons, it is up to the user to provide the additional state frames on the timeline if the user chose to use the standard button component over the provided Toggle Button component. However, it is recommended that the user makes use of the prebuilt Toggle Button to save time. The same is true for Animated Button and Animated Toggle Button described in sections 4 and 5.

In order for the toggle button to function properly, make sure the *toggle* field in the *Component Inspector Parameters* tab in CS4 (*Parameters* tab in CS3) is set to "true." By setting the toggle property of any CLIK Button component to you force it to behave like a Toggle Button.



label	
toggle	true

Figure 6: The toggle parameter set to 'true'.

3.1 Toggle Button States

In addition to the standard Button states—*up, over, down,* and *disabled*—a Toggle Button also includes the following specialized states:

- *selected_up*
- *selected_over*
- *selected_down*
- *selected_disabled*

These four specialized states are used to represent a Toggle Button which has been toggled "on" or selected, whereas the original Button states of *up, over, down,* and *disabled* are used in a Toggle Button to represent a button which has been toggled "off" or unselected.

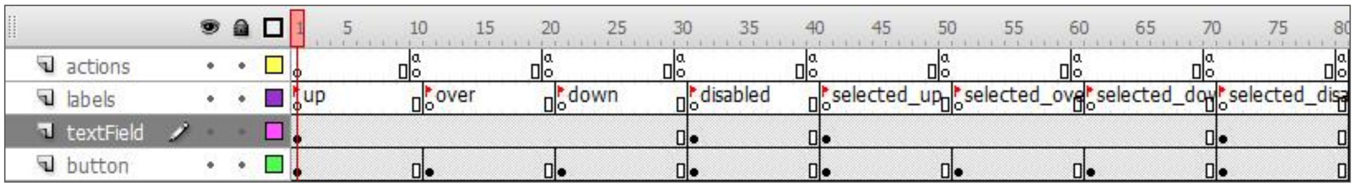


Figure 7: The Toggle Button timeline with states clearly labeled.

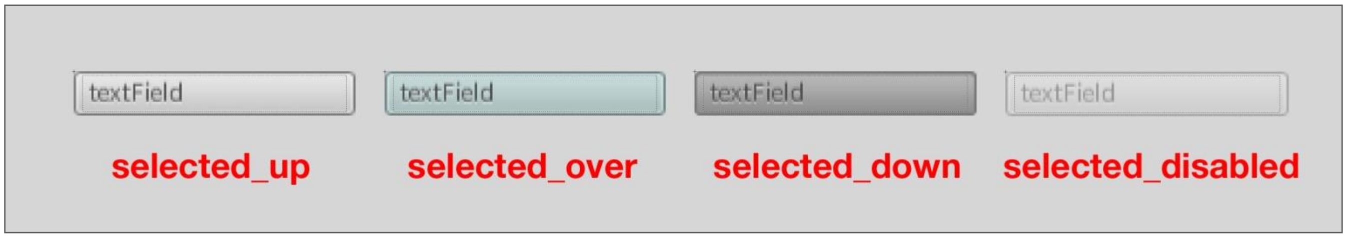


Figure 8: The new Toggle Button states' graphical wireframe representations side by side.

3.1.1 Selected_Up

Selected_up is the normal "at rest" state of an unfocused Toggle Button that has been toggled "on" or "selected" by the user. This state typically includes a graphical indicator of some kind to indicate the button is "on" such as a thick border or color change.

Occurs: When a user clicks on a Toggle Button that is currently "off" or "unselected," the button will become "selected" or "on" and go to the *selected_over* state. Also, once the user moves focus away from this button by clicking on another component, or using the arrow keys or game controller, it will change to the *selected_up* state.
Usage: Represents an unfocused Toggle Button set to "selected" or "on."

3.1.2 Selected_Over

Selected_over occurs when the user moves the mouse cursor over a toggled button or uses the arrow keys or game controller to move to an unfocused Toggle Button that is “on” or “selected.” Moving the mouse away or using the arrow keys or game controller to move away from the button will return the button to the *selected_up* state.

Occurs: When the user moves the mouse cursor over a Toggle Button set to “on” or “selected” or when the user focuses on the button using the arrow keys or game controller.
Usage: Represents a focused Toggle Button set to “on” or “selected.”

3.1.3 Selected_Down

Selected_down occurs when the user presses on a Toggle Button set to “on” or “selected” via a left mouse click, the (Enter) key, or the appropriate button on a game controller. The button will remain in the *selected_down* state for as long as the mouse button, (Enter) key, or game controller button is pressed. The act of pressing a toggle button switches its selected state, and therefore the button will return to the *over* state, which represents “unselected” or “off,” once released. The converse is true for the *down* state, where the button moves to the *selected_over* state.

Similar to the standard Button, pressing on a selected Toggle Button transfers focus to it. Its default state will be set to the *selected_over* state if the Toggle Button is “on,” or the *over* state if the button is “off.” If the user moves the mouse cursor away from the button component while the mouse button is still depressed and then releases the mouse button, then the button component will revert to the *selected_over* state if “on,” or the *over* state if “off.”

Occurs: When the user presses the left mouse button while the mouse cursor is over the Toggle Button, or presses the (Enter) key or game controller button while the Toggle Button is focused.
Usage: Represents a focused Toggle Button set to “on” that is being clicked.

3.1.4 Selected_Disabled

The *selected_disabled* state, much like the standard Button *disabled* state, represents the state of a Toggle Button set to “on” that has been disabled. A disabled Toggle Button cannot be pressed or focused, and does not react to mouse over events, keyboard events, or game controller events. A Toggle Button may be disabled via the *Parameters* tab, or via AS using the following code:

```
buttonInstanceName.disabled = true;
```

Occurs: Set via ActionScript or *Parameters* tab.
Usage: Represents a Toggle Button that is selected or set to “on” that has been disabled and is unusable.

4. Animated Button

Animated Buttons are essentially the same as the standard Button component; however, they may include animated transitions from state to state (e.g., up to over, over to down). Use an Animated Button whenever it is desirable to have a timeline based animation as a transition between the various states of a button, as opposed to a standard Button, which simply snaps to each button state without animated effects.

A few possible use cases for the Animated Button component might be:

- Menu buttons with animated transitions
  - OK/Cancel/Apply
  - Screen select
- In-game console control buttons with animated transitions
- Mini-game buttons with animated transitions

The only real difference between the core and animated button is visual. If a design calls for animated effects on the buttons as they move from state to state, then the Animated Button is the proper choice; otherwise it is best to use the core Button, as it contains less data in the form of transition keyframes and animation tweens.

No extra code is required to create an Animated Button. The CLIK Button class provides all of the functionality for the different variants described in this document.

4.1 Animated Button Transitions

Animated Buttons use a different terminology to describe the keyframes associated with their states. Rather than referencing the states themselves, each keyframe represents an animated transition between two states. *Up*, *down*, and *over* still have meaning as states, but for the purposes of the animated button we have recycled the names for the analogous transitions.

Think of *up*, *down*, *over*, *selected_up*, *selected_down* and *selected_over* as the states that the transitions move the button to and from. This helps understand the complexity of the many different transitions possible in a robust animated button component.

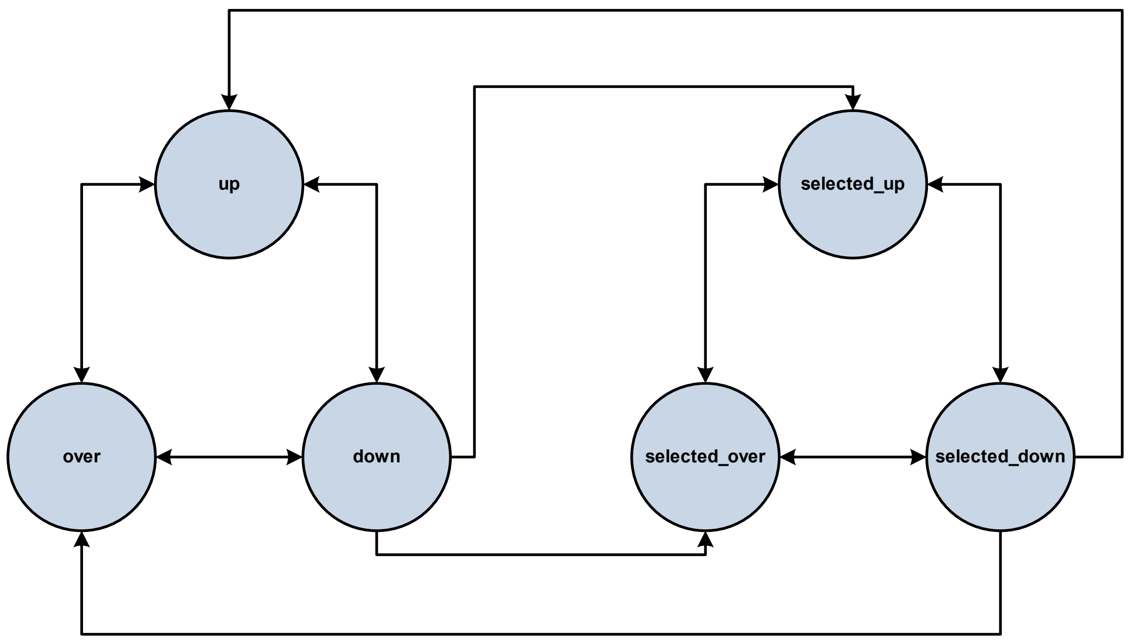


Figure 9: Animated Button state transitions.

To have a classic animation tween between states, set the last keyframe to the end or final state's appearance. This is the final resting point of the animation and should represent the state the button will be in when the animation finishes. Then create a tween that blends from the first frame to the last frame. For example: in the case of the *over* transition, the first keyframe of the animation could be a graphic similar to the *up* state of the button, and the final keyframe would be a graphic representing the actual *over* state of the button.

In addition to the core Button transitions of *up*, *over*, *down*, and *disabled*, an Animated Button includes the following specialized transitions:

- *release*
- *out*
- *kd_down*
- *kb_release*

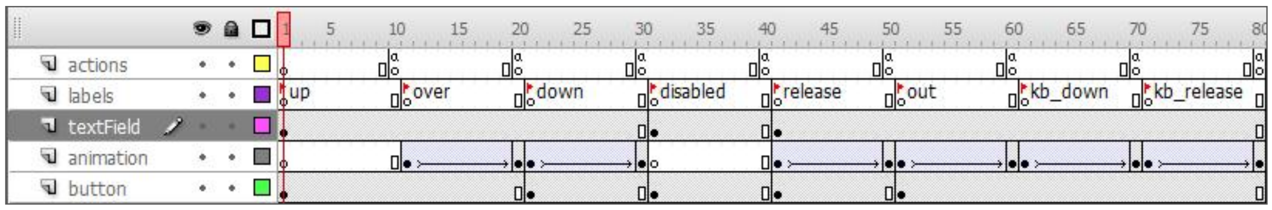


Figure 10: The animated button timeline with transitions clearly labeled.

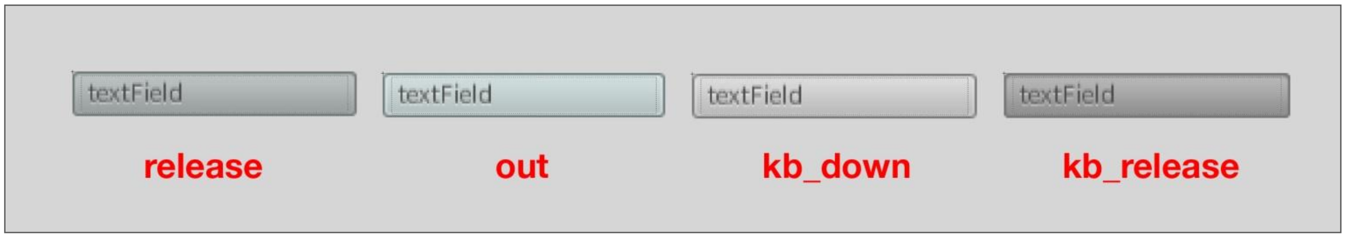


Figure 11: The new Animated Button states' graphical wireframe representations side by side.

The *over* and *down* states of an Animated Button have new functionality over the core Button versions. Instead of being static image states, they use animated transitions that end in the static state appearance.

4.1.1 Over

Over functions slightly differently in an Animated Button. It is used to display the animated transition from the *up* state of a button to the *over* state of the button.

Occurs: When the user moves the mouse cursor over the button or uses the arrow keys or game controller to focus on the button.
Usage: Represents a button that the mouse cursor is currently over or that has focus.

4.1.2 Down

Down is used to display the animated transition from the *over* state of a button to the *down* state of the button.

Occurs: When the user presses the left mouse button while over a button, presses the (Enter) key or presses the appropriate button on a game controller while on a focused button.
Usage: Represents a button which has been clicked.

4.1.3 Release

Release is a state unique to Animated Buttons and Animated Toggle Buttons. It is used to display the animated transition from the *down*—or pressed—state of a button to the *over* state of the button. The last frame of this transition (the *over* state) is used to show the focused state.

Occurs: When the user clicks and releases the button with the left mouse button, the (Enter) key or a game controller button. Also occurs when the user presses down on the left mouse button while over the button and drags the mouse cursor away from the button.
Usage: Represents a focused Animated Button where the left mouse button, enter key or equivalent controller button was released.

4.1.4 Out

Out is used to display the animated transition from the *over* state of a button to the *up* state of the button.

Occurs: When the user moves the mouse cursor away from the button.
Usage: Represents an unfocused button where the mouse cursor has moved away from the button boundary.

4. 2 Keyboard Transitions

Keyboard transitions are used to display special animated transitions from one state to another when the keyboard or game controller, rather than the mouse, is used, and a separate focus indicator movie clip is used (see section 4.2.1 for more details on the focus indicator). There are two additional transitions that are required for keyboard events if a separate focus indicator is used: *kb_down* and *kb_release,* described in sections 4.2.2 and 4.2.3 respectively.

4. 2.1 Focus Indicator

By default, the Button component uses the last frame of the *release* transition to display the focused state. However, it is sometimes best to have a separate graphic as an indicator for the focused state, particularly when keyboard and/or game controller input is used. The CLIK Button inherently supports this functionality if the button movie clip contains a focus indicator movie clip internally. This focus indicator movie clip instance must be named 'focusIndicator', and it must be composed of two keyframes labeled: 'hide' and 'show' in that order. The *show* frame should have a graphic used to represent focus, such as a glowing border. The *hide* frame should be used to represent an unfocused button, and will typically have no graphic at all. To create this focus indicator movie clip, follow the steps below:

1. Create a new layer inside the button's timeline.
2. Name the layer 'FocusIndicator' for organizational purposes.
3. Ensure the layer's total number of frames extend to the end of the button's timeline.

4. Create a new movie clip on the *FocusIndicator* layer that will represent keyboard focus of the button.
   a. For instance, draw a rectangle with an empty fill and a red border that surrounds the button graphic.
   b. Right click the rectangle on the *stage* and select *Convert to Symbol.*
   c. Name the movie clip, and hit Ok.

5. Give the movie clip an instance name of 'focusIndicator' in the *Properties* panel.

6. Double click the *focusIndicator* movie clip to enter its timeline.

7. Inside the timeline, add a second keyframe at *frame 2* on the *Layer1* timeline*.*

8. Select and label *keyframe 1*: 'hide' on the *Properties* panel.

9. Open the Actions panel (F9) and add the following code on frame 1: `stop();`

10. Select and label *keyframe 2:* 'show'.

11. In the Actions panel, add the following code on frame 2: `stop();`

12. Ensure the red rectangle is visible on the *show* frame, and deleted on the *hide* frame.

4. 2.2 KB_Down

Keyboard events are passed to the button only when it has focus. If a separate focus indicator is not used then the button's focused state will fall back to its *over* state (the last frame of the *release* transition). If a separate focus indicator is used then an *up* to *down* transition is required since the button will be in its *up* state even when focused.

Kb_down is used to display the animated transition from the *up* state of the button to the *down* state of the button.

Occurs: When the user presses the (Enter) key or the appropriate button on a game controller on a focused button that has a separate focus indicator movie clip.
Usage: Represents a pressed button with a separate focus indicator.

4. 2.3 KB_Release

Kb_release is used to display the animated transition from the *down* state of a button to the *up* state.

Occurs: When the user releases the (Enter) key or the appropriate button on a game controller on a focused button that has a separate focus indicator movie clip. May also occur if a separate focus indicator is used and the user drags the mouse away, or uses the arrow keys or game controller to move away from the button to another button before releasing the (Enter) key, game controller button or left mouse button.
Usage: Represents a focused button with a separate focus indicator that was released.

5. Animated Toggle Button

Animated Toggle Buttons are a hybrid component that combines Toggle Buttons with Animated Buttons. Use an Animated Toggle Button whenever it is desirable to have a timeline based animation as a transition between the various states of a Toggle Button, as opposed to a standard Toggle Button, which simply snaps to each button state without animated effects.

This button type is the most comprehensive of the four types, as it includes the functionality of all of the previous three types, as well as specialized transitions for animated toggle buttons. Like the animated button from the previous section, we have recycled *selected_up*, *selected_down* and *selected_over* as transitions. They still have meaning as states.

A few possible use cases for the Animated Toggle Button component might be:

- Menu buttons which can be toggled "on" or "off," with animated transitions
    - Enable/disable bloom visual effect
    - Mute/unmute audio
    - High quality lighting on/off
    - Auto save on/off
- In-game animated console controls that require an on/off state
- Mini-game animated buttons that require an on/off state

5.1 Animated Toggle Button Transitions

Animated Toggle Buttons use the same terminology as Animated Buttons to describe the keyframes associated with them. Rather than referencing states, each keyframe represents an animated transition between two states.

An animated toggle button includes all the previous states and transitions from Button, Toggle Button, and Animated Button. It also includes the following eight specialized transitions:

- *selected_selecting*
- *selected_release*
- *selected_out*
- *selecting*
- *selected_kb_down*
- *selected_kb_release*
- *selected_kb_selecting*

- *kb_selecting*

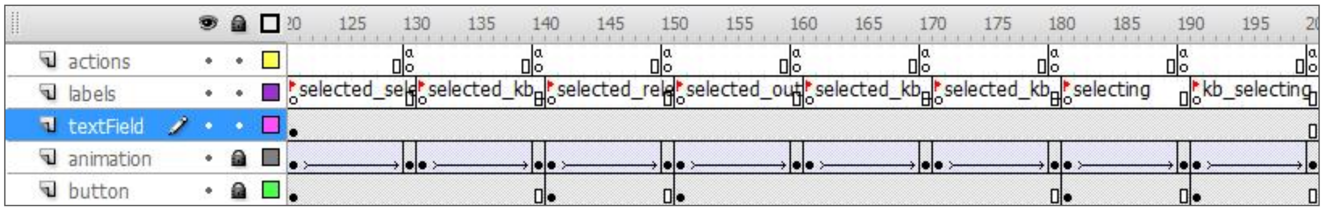


Figure 12: The animated toggle button timeline with new transitions clearly labeled.

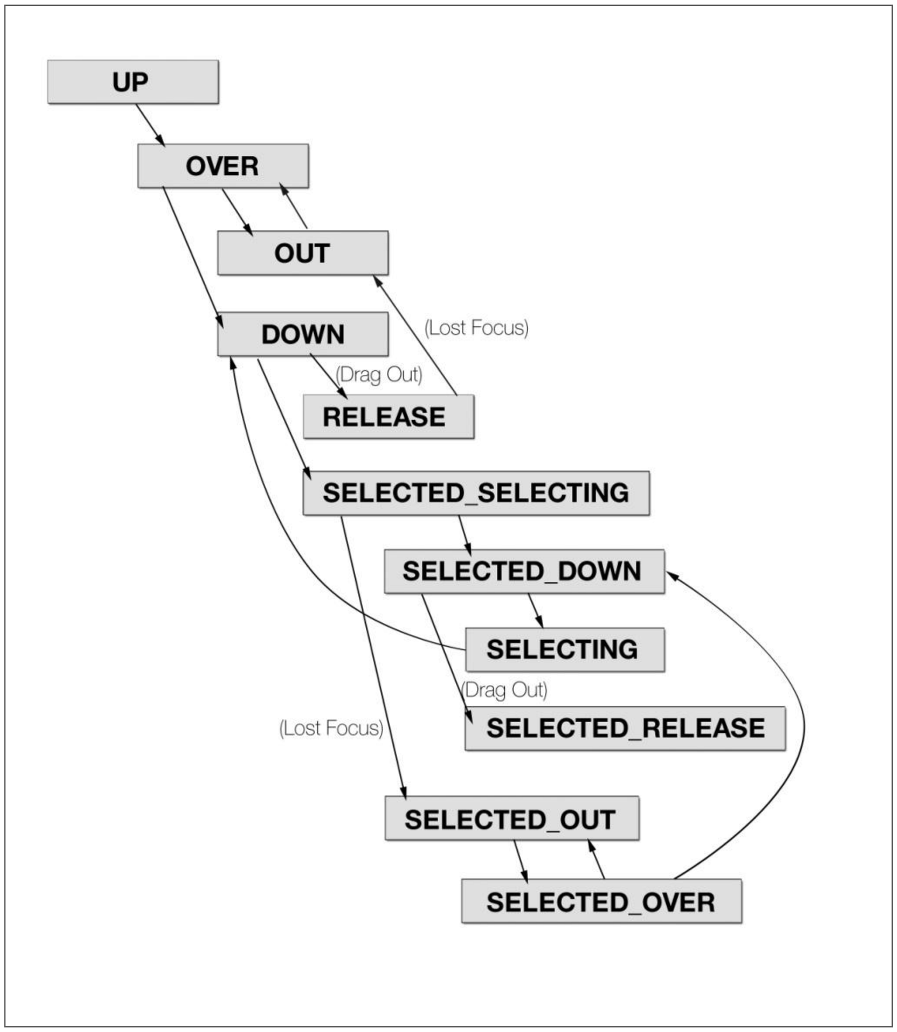


Figure 13: The animated toggle button transitions (not including keyboard transitions).

5.1.1 Selected_Selecting

Selected_selecting is used to display the animated transition from the *down* state of the button to the "selected" or "on" and focused state—*selected_over*—of the button. This is a toggle transition from the "unselected" state to the "selected" state.

Occurs: When the user clicks and releases an Animated Toggle Button that is currently not toggled "on" or "selected." The button then toggles "on" or "selected."
Usage: Used to indicate a button that is toggled "selected" and has focus but no focus indicator.

5.1.2 Selected_Release

Selected_release is used to display the animated transition from the *selected_down* state of a button—a button that has been pressed and is currently in the "selected" state—to the "selected" and focused state of the button: *selected_over*.

Occurs: When the user clicks on a selected button and drags the mouse cursor away from the button. This effectively cancels the toggle transition from "selected" to "unselected."
Usage: Used to indicate a button that is toggled "selected" and has focus, whose click action has been cancelled.

5.1.3 Selected_Out

Selected_out is used to display the animated transition from the *selected_down* state of a button (a button which is "selected" , focused and pressed) to the "selected" and unfocused state of the button — *selected_up*.

Occurs: When the user clicks another component with the left mouse button or uses the arrow keys or game controller to select another component, causing the focused "selected" button to lose focus.
Usage: Used to represent an unfocused, though still "selected" button that just lost focus.

5.1.4 Selecting

Selecting is used to display the animated transition from the selected*_down* state of the button to the *over* state of the button.This is a toggle transition from the "selected" state to the "unselected" state.

Occurs: When the user clicks and releases on an Animated Toggle Button that is currently “selected.” The button then becomes “unselected.”
Usage: Used to represent an “unselected,” though still focused button that was just toggled.

5.2 Keyboard Transitions

Animated Toggle Buttons require the same setup instructions detailed in the Animated Button keyboard transitions section of the document.

5. 2.1 Selected_KB_Down

The *selected_kb_down* transition is used to display the animated transition from the *selected_up* state of the button to the *selected_down* state of the button.

Occurs: When a “selected,” focused Animated Toggle Button, with a separate focus indicator movie clip, has been pressed via the (Enter) key on the keyboard or the appropriate button on a game controller.
Usage: Represents a “selected” Animated Toggle Button that has been pressed via the (Enter) key or game controller button.

5.2.2 Selected_KB_Release

Selected_kb_release is used to display the animated transition from the *selected_down* state of a selected, focused button to the *selected_up* state of the button.

Occurs: When a “selected,” focused Animated Toggle Button, with a separate focus indicator movie clip, has been pressed via the (Enter) key or game controller button, or clicked on with the left mouse button, and then the user drags the mouse cursor away from the button, or uses the arrow keys or game controller to select another button without letting go of the (Enter) key, game controller button, or left mouse button.
Usage: Represents a “selected,” focused Animated Toggle Button that was released.

5.2.3 Selected_KB_Selecting

Selected_kb_selecting is used to display the animated transition from the *down* state of a button to the *selected_up* state of the button. This is a toggle transition from the “unselected” state to the “selected” state.

Occurs: When the user toggles an "unselected," focused Animated Toggle Button by pressing and releasing the (Enter) key or game controller button.
Usage: Used to represent a "selected," focused Toggle Button that has been toggled via keyboard or game controller, and has a separate focus indicator movie clip.

5.2.4 KB_Selecting

The *kb_selecting* transition is used to display the animated transition from the *selected_down* state to the *up* state of the button. This is a toggle transition from the "selected" state to the "unselected" state.

Occurs: When the user toggles a "selected," focused Animated Toggle Button by pressing and releasing the (Enter) key or game controller button.
Usage: Used to represent an "unselected," focused Animated Toggle Button that has been toggled via keyboard or game controller, and has a separate focus indicator movie clip.

6. Conclusion

The Scaleform CLIK button variants have been designed to provide a UI artist with greater flexibility. Learning the various transitions and states of each button variant are critical to successful use of CLIK. Knowing when to use each variant will help save time and effort.

- *Button*: the base button variant
- *Toggle Button*: Button with added toggle on/off functionality
- *Animated Button*: Button with animated transitions
- *Animated Toggle Button*: a combination of Toggle Button and Animated Button functionality