Autodesk®
Scaleform®

Getting Started with CLIK

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

Authors: Matthew Doyle
Version: 3.1
Last Edited: May 12, 2011

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.4

How to Contact Autodesk Scaleform:

Document	Getting Started with CLIK
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

Autodesk® Scaleform® Common Lightweight Interface Kit (CLIK™) allows developers to rapidly prototype efficient front-end menu interfaces for games, and then take those prototypes to completion. This document will cover the basics of creating a custom game front end using Scaleform CLIK components and ActionScript™, as well as several best practices for component creation and skinning. The document also offers detailed step-by-step directions for each phase of creation.

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.

1.1 Minimum Software Requirements

- Scaleform 3 and up
- Adobe® Flash® CS3 and up
- Adobe Photoshop® CS3 and up

2. Configuring CLIK and Installing the Scaleform Launcher

Scaleform CLIK comes with a new Adobe Flash panel designed to improve workflow. The first step to using Scaleform CLIK components and the workflow enhancement panel is to ensure the Scaleform CLIK class paths have been properly added to the Flash environment. Then use the Adobe Extension Manager to install the panel extensions.

The installations for Creative Suite® 3 (CS3) and CS4/CS5 are slightly different. Installation instructions are thus provided for CS3 and CS4/CS5.

2.1 Adding Scaleform CLIK Class Paths to Flash CS3

The Flash class path must be set up for the Scaleform CLIK classes to be exposed to the authoring environment. You will need to set up the class paths for both ActionScript 2.0 and ActionScript 3.0.

1. Launch Flash CS3.
2. Click *Edit* in the top Flash menu, then *Preferences* in the dropdown menu.
3. Choose *ActionScript* under *Category*.

ActionScript 2.0

4. Click *ActionScript 2.0 Settings* under *Language*.
5. In the *ActionScript 2.0 Settings* dialogue, add a new path by pressing the *Plus [+]* button.

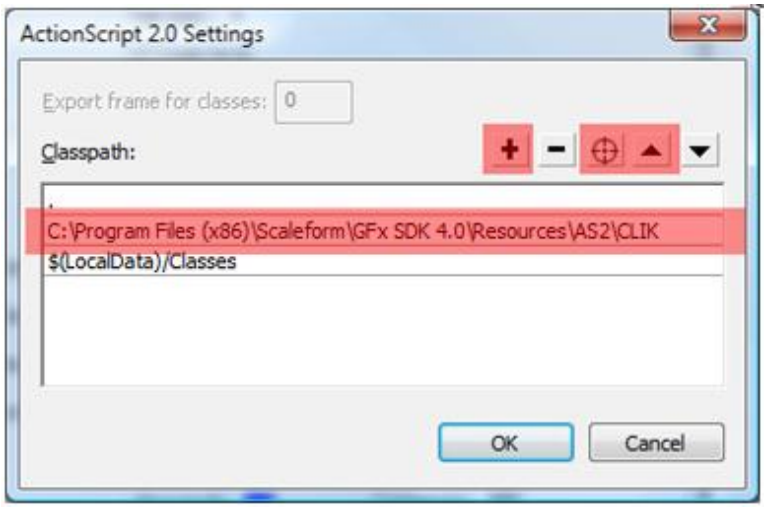

Figure 1: A*ctionScript 2.0* class path window in CS3 on Windows.

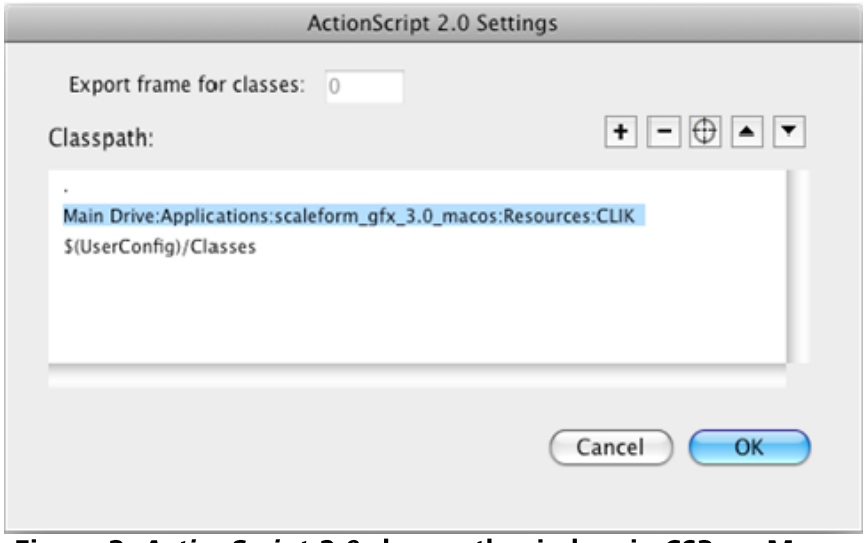

Figure 2: *ActionScript 2.0* class path window in CS3 on Mac.

6. Click the *Browse to Path* icon, which looks like a crosshair in CS3, to select the directory where the Scaleform CLIK components are installed. In a default installation on Windows, this is *C:/Program Files/Scaleform/GFx SDK 4.4/Resources/AS2/CLIK/* and on Mac, the components are located in *scaleform_gfx_4.4_macos/Resources/AS2/CLIK/* in the installed Scaleform 4 SDK directory.

7. After returning to the *ActionScript 2.0 Settings* dialogue, select the added path under *Classpath*. Move it above the *$(LocalData)/Classes* line by using the *Arrow* buttons for Windows and move the path above *$(UserConfig)/Classes* line for Mac. This is to ensure that the Scaleform CLIK classes are referenced before the built-in classes. If this step is not taken, the built-in Adobe components will be used instead of the Scaleform CLIK components.

8. Press *OK* to close the *ActionScript 2.0 Settings* window.

ActionScript 3.0

9. To set up CLIK ActionScript 3.0, click *ActionScript 3.0 Settings* under *Language*.

10. In the *ActionScript 3.0 Advanced Settings* dialogue, add a new *Source* path by pressing the first *Plus [+]* button at the top.

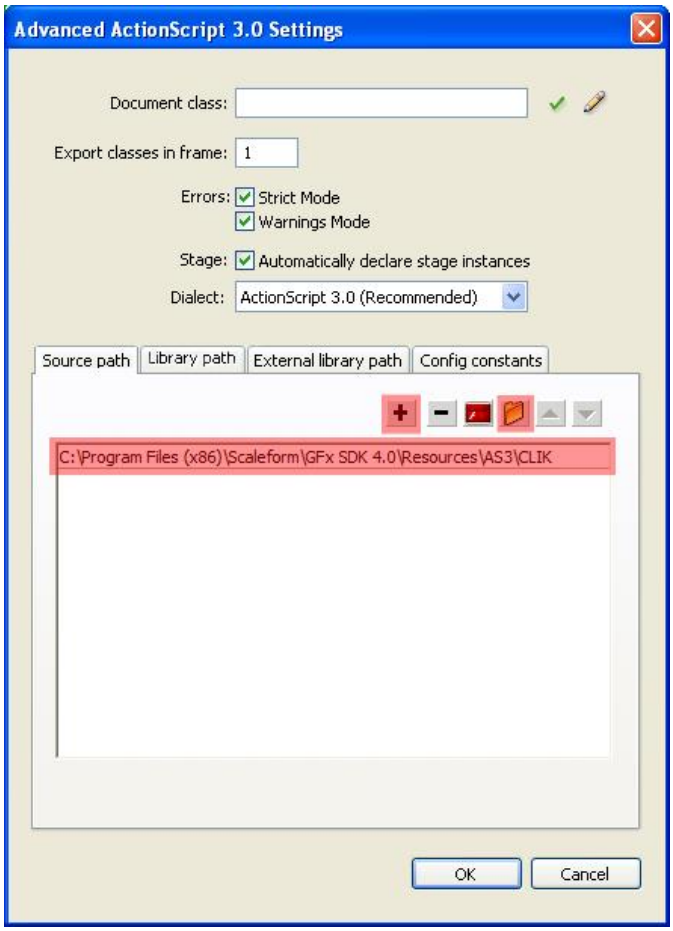

Figure 3: A*ctionScript 3.0* class path window in CS3 on Windows.

11. Click the *Browse to Path* icon, which looks like a crosshair in CS3, to select the directory where the Scaleform CLIK components are installed. In a default installation on Windows, this is *C:/Program Files/Scaleform/GFx SDK 4.4/Resources/AS3/CLIK/* and on Mac, the components are located in *scaleform_gfx_4.4_macos/Resources/AS3/CLIK/* in the installed Scaleform 4.4 SDK directory.

12. Press *OK* to close the *ActionScript 3.0 Advanced Settings* window.

13. Press *OK* to close the *Preferences* window.

2.2 Adding Scaleform CLIK Class Paths to Flash CS4 & CS5

The Flash class path must be set up for the Scaleform CLIK classes to be exposed to the authoring environment. You will need to set up the class paths for both ActionScript 2.0 and ActionScript 3.0.

1. Launch Flash CS4 or CS5.
2. Click *Edit* in the top Flash menu, then *Preferences* in the dropdown menu.
3. Choose *ActionScript* under *Category*.

ActionScript 2.0

4. Click *ActionScript 2.0 Settings* under *Language.*
5. In the *ActionScript 2.0 Settings* dialogue, add a new path by pressing the *Plus [+]* button.

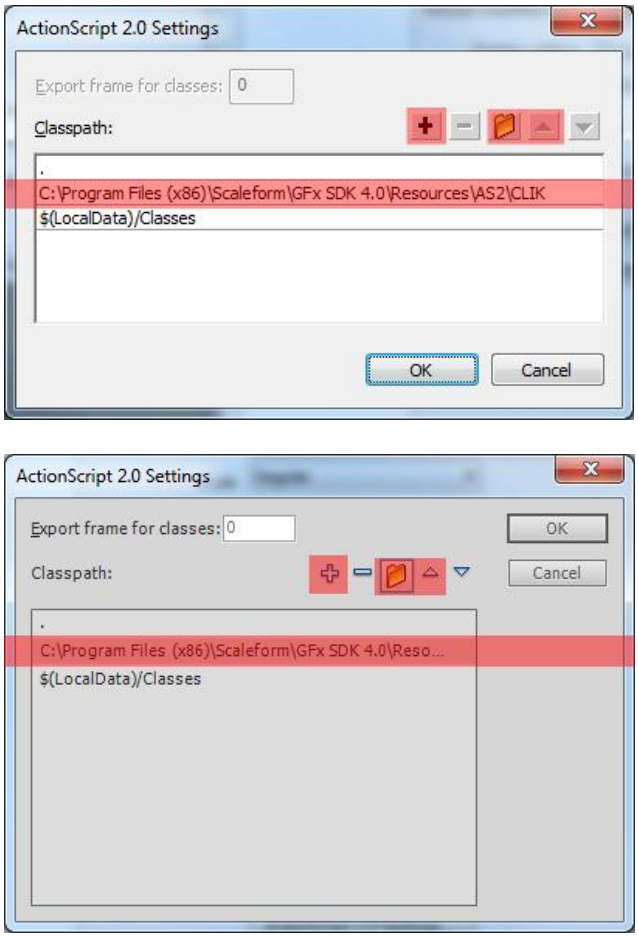

Figure 4: ActionScript *2.0* class path window in CS4 (top) and CS5 (bottom) on Windows.

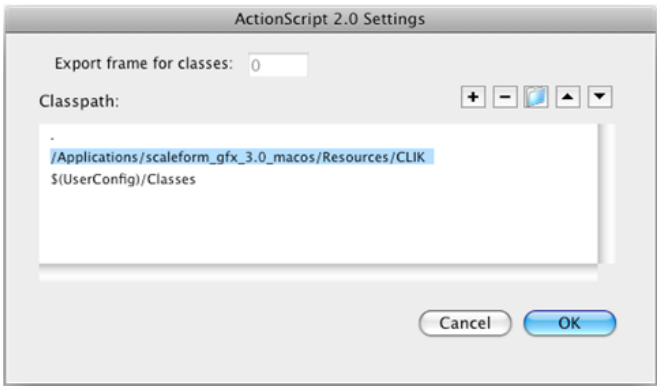

Figure 5: ActionScript *2.0* class path window in CS4 on Mac.

6. Click the *Browse to Path* icon, which looks like a folder icon in CS4, to select the directory where the Scaleform CLIK components are installed. In a default installation on Windows, this is *C:/Program Files/Scaleform/GFx SDK 4.4/Resources/AS2/CLIK/* and on Mac, the components are located in *scaleform_gfx_4.4_macos/Resources/AS2/CLIK.*

7. After returning to the *ActionScript 2.0 Settings* dialogue, select the added path under *Classpath*. Move it above the *$(LocalData)/Classes* line by using the *Arrow* buttons for Windows and move the path above *$(UserConfig)/Classes* line for Mac. This is ensures the Scaleform CLIK classes are referenced before the built-in classes. If this step is not taken, the built-in Adobe components will be used instead of the Scaleform CLIK components.

8. Press *OK* to close the *ActionScript 2.0 Settings* window.

ActionScript 3.0

9. To set up CLIK ActionScript 3.0, click *ActionScript 3.0 Settings* under *Language*.

10. In the *ActionScript 3.0 Advanced Settings* dialogue, add a new *Source* path by pressing the first *Plus [+]* button at the top.

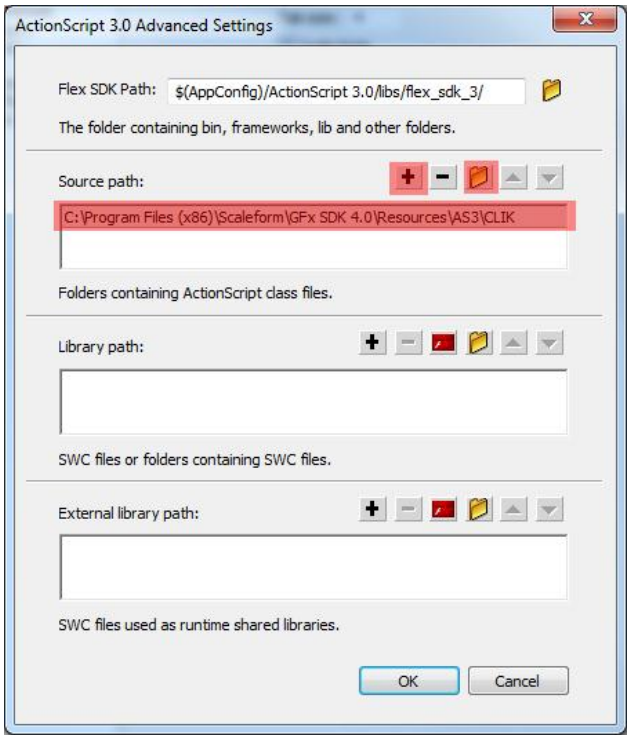

Figure 6: *ActionScript 3.0* class path window in CS4 and CS5 on Windows.

11. Click the *Browse to Path* icon, which looks like a crosshair in CS3, to select the directory where the Scaleform CLIK components are installed. In a default installation on Windows, this is *C:/Program Files/Scaleform/GFx SDK 4.4/Resources/AS3/CLIK/* and on Mac, the components are located in *scaleform_gfx_4.4_macos/Resources/AS3/CLIK/* in the installed Scaleform 4 SDK directory.

12. Press *OK* to close the *ActionScript 3.0 Advanced Settings* window.

13. Press *OK* to close the *Preferences* window.

2.3 Adding the Scaleform Launcher Panel.

The Scaleform Launcher panel provides a quick and accessible method for publishing a SWF file directly to the Scaleform player from the Flash authoring environment. Please review the installation and usage instructions in sections 2.4 through 2.6 in the document Getting Started with Scaleform 4.4 if you haven't already installed the panel.

3. Creating a Simple Front-End Game Menu with Components

This section will detail the steps necessary to create a front-end game menu. The menu will consist of two screens: main menu screen and options screen. The main menu screen will have a single button that takes the user to the options screen. The bulk of the menu work will be on the options screen, which will include a difficulty setting implemented with an option stepper, video options for bloom and anti-aliasing built using checkboxes and radio buttons, and a sound volume slider control.

NOTE: This tutorial was written in ActionScript 2.0.

Assets from this tutorial have been provided in CLIK folder located in the installed Scaleform 4.4 SDK directory:

Resources/AS2/CLIK/tutorial
Resources/AS2/CLIK/tutorial/art_assets

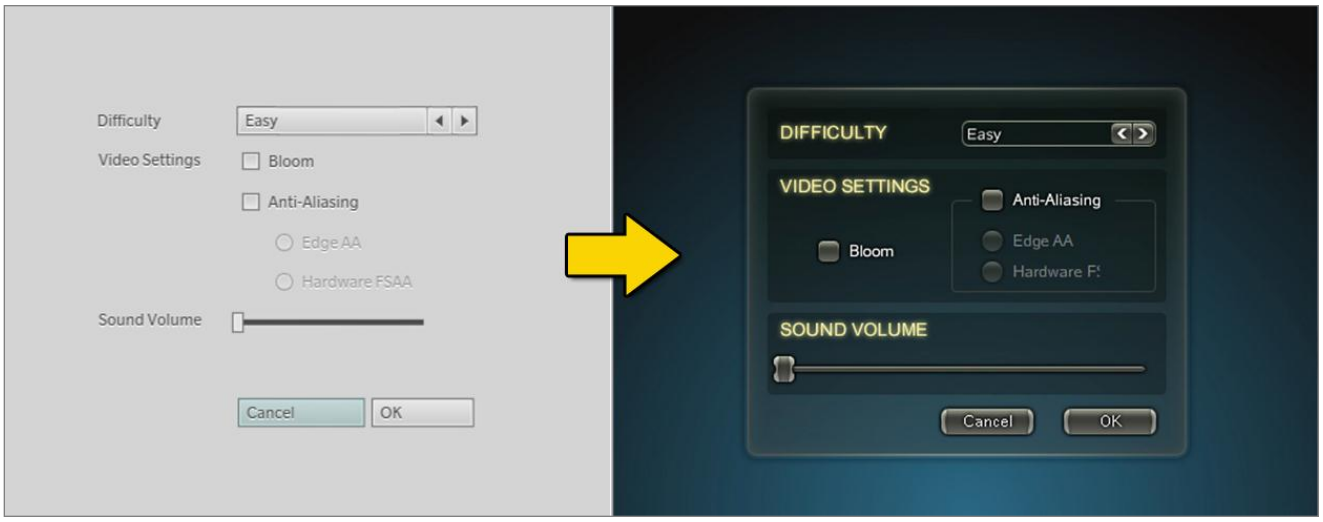

Figure 7: Final Version of Options Screen from Unskinned to Skinned.

The custom user interface that will be created in this tutorial will have the following workflow:

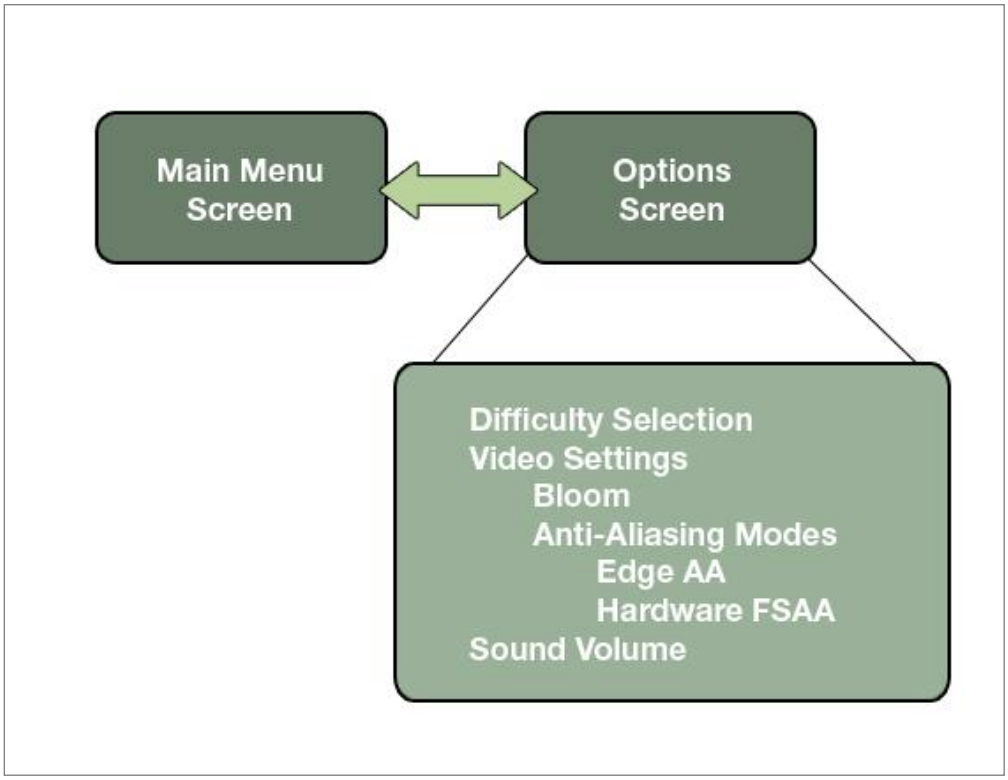

Figure 8: The front end menu flow.

3.1 Creating the Menu Shell

The first step is to set up the various layers and keyframes for the menu. It is here that the rough skeletal setup of the front end will be created.

1. Create a new Flash (ActionScript 2.0) document.

2. Ensure the document's publish settings are set to Player 8 by selecting *File* from the top menu, choosing 'Publish Settings' in the dropdown, then clicking on the *Flash* tab.

3. Set the *Version* dropdown to 'Flash Player 8' or better (Scaleform 4 supports up to Flash Player 10.1).

4. Set the *ActionScript version* dropdown to 'ActionScript 2.0'.

5. Press *OK* to close the *Publish Settings* window.

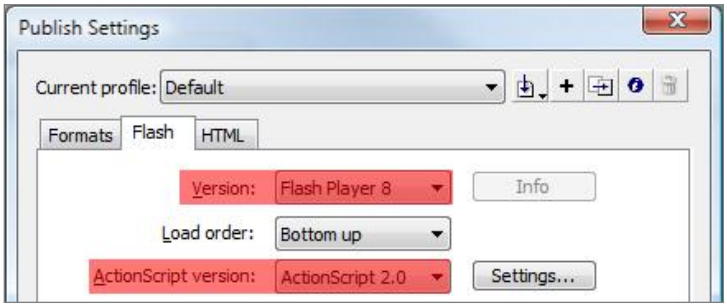

Figure 9: Publish settings window.

6. Select the current layer on the main document timeline, currently labeled *Layer 1*, and rename it to 'scene' by double clicking on the word *Layer 1*. While the name is not important, naming layers in this manner is considered a best practice.

7. Create a new layer by clicking the *Insert Layer* button, and name this layer 'actions'. Again, the name of the layer is not important, but useful for remembering the layer's purpose.

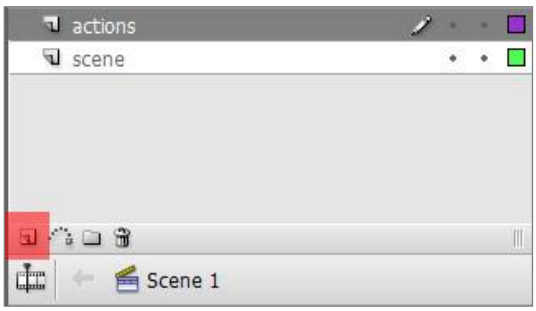

Figure 10: The insert layer button.

8. Add a second keyframe on frame 10 for both the *actions* and *scene* layers. Select the *scene* layer, and then right click on frame 10 of the timeline. Select 'Insert Keyframe' from the popup menu. This keyframe will be used to create the Options screen portion of the menu. Repeat this step for the *actions* layer.

Figure 11: The main menu timeline with added keyframes on frame 10.

9. Label the keyframe on frame 1 of the *scene* layer 'mainMenu'. This can be accomplished by selecting the keyframe and entering the name on the *Properties* tab. This label will be used by the code to tell Scaleform to "goto" the *mainMenu* frame of the timeline. Frame 1 will be used to create the Main screen of the menu.

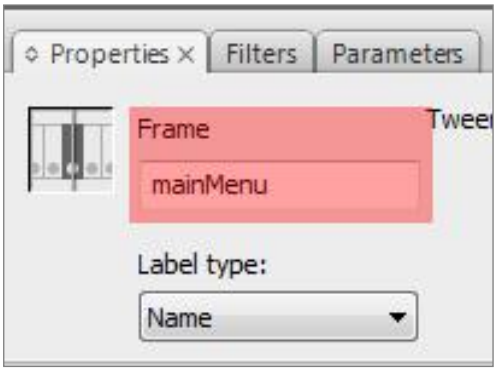

Figure 12: Keyframe 1 is labeled 'mainMenu'.

10. Label the keyframe on frame 10 of the *scene* layer 'optionsScreen'. This label will be used by the code to tell Scaleform to "goto" the *optionsScreen* frame of the timeline.

11. Select frame 1 on the *actions* layer. Open the *Actions* window (F9) and enter the code "stop();" (without quotes) on the first line. This code tells the player to stop playback on this frame. If the code were not there, the player would continue to play until it did reach a stop command, or until it reached the end of the timeline, in this case frame 10. Select frame 10 of the *actions* layer, and add the same code.

12. At this time, it would be a good idea to save the Flash file.

3.2 Adding the Main Menu Navigation Controls

This step explains how to add an *Options* button to the *Main Menu*, which will allow the user to enter the *Options Screen*.

1. Open the file *CLIK_Components.fla* located in the *Resources/AS2/CLIK/components* of the installed Scaleform 4 SDK directory.

2. First add the menu title. Do this by right clicking the *Label* component in the *Library* panel of *CLIK_Components.fla* and selecting 'Copy' from the popup menu.

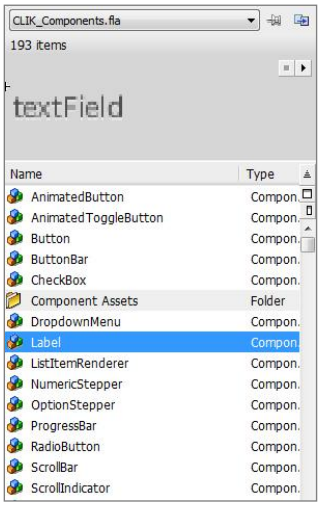

Figure 13: The library panel and label component.

3. Do not close the *CLIK_Components.fla* file, as it will be needed again. Return to the original Flash document being used to create the tutorial menu.

4. Select the first keyframe of the *scene* layer.

5. In the *Library* pane, right click on a blank area and select *Paste* to copy the *Label* component into the document's Library.

6. Click and drag the *Label* component from the *Library* pane onto the stage somewhere, preferably in the center, near the top.

7. Open the Parameters tab for the newly created label, and enter 'Main Menu' in the *text* field. This field is what will be displayed at runtime in the player. **Note:** In CS3, the tab is located next to the *Properties* and *Filters* tabs. It is located in the *Component Inspector* in CS4, and is turned off by default. Press [Shift + F7] to show it, or select Window – Component Inspector from the top menu.

Note: These parameters will only be displayed when the SWF is published and played.

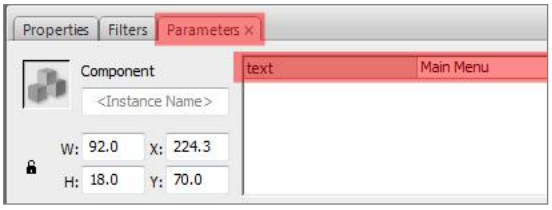

Figure 15: The parameters tab is located next to the properties tab in CS3.

8. Select keyframe 10 and repeat steps 6 and 7 to create a title for the Options screen – enter 'Options' in the text field this time.

9. Open the *CLIK_Components.fla* file, and then right click the *Button* component in the *Library* pane and select 'Copy' again.

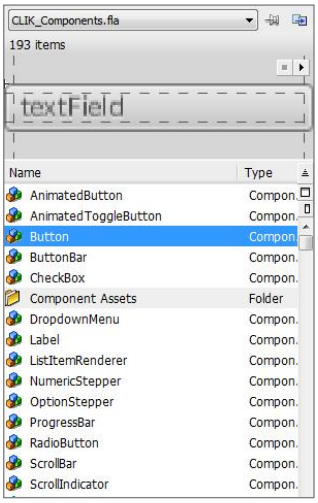

Figure 14: Button component in the library pane.

10. Select the original Flash document.

11. In the *Library* pane, right click on a blank area and select *Paste* to copy the *Button* component into the document's Library.
12. Select the first keyframe of the *scene* layer.

13. Click and drag the *Button* component from the Library pane onto the Stage of the document, below the *Main Menu* label.

14. Select the new button on the Stage, and change its instance name—found in the *Properties* tab—to 'optionsBtn'. This instance name will be referenced by the code which defines the button's behavior.

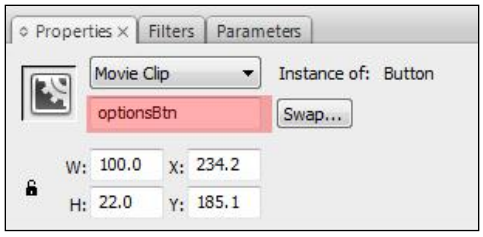

Figure 16: Name the options button instance 'optionsBtn'.

15. Click the *Parameter* tab, and enter 'Options' in the *label* field. This field contains what will be displayed on the button at runtime.

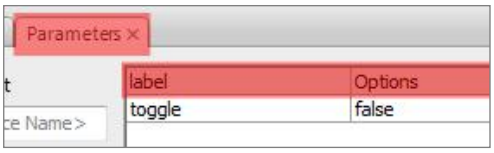

Figure 17: Label the button 'Options'.

16. Congratulations! Test the first screen by using the *Scaleform Launcher* panel. Press the *Test with [Executable Name]* button in the *Scaleform Launcher* panel to run the movie in the Scaleform player.

Figure 18: Main menu with options screen button in Scaleform player.

3.3 Listening for the Options Screen Button

The first interactivity needed is to add functionality to the main menu's options button, which will allow the user to press it and go to the *Options Screen*. In this sample the Scaleform CLIK event model will be used to perform this function. While other methods could be used, the Scaleform CLIK event model will provide greater flexibility and efficiency when using Scaleform CLIK components. In this model, the *Options* button is told to invoke a callback function when a specific event is fired. The callback is a function that tells the main timeline to advance to the *Options Screen* keyframe, and the event is a mouse click, fired when the user presses the left mouse button.

1. Select the first keyframe of the *actions* layer.

2. Insert the following code above the "`stop();`" command. This is the function that will take the user to the options menu, which is located on frame 10:

```
function OpenOptionsScreen()
{
      gotoAndPlay("optionsScreen");
}
```

3. Now insert the code that tells the *optionsBtn* instance to invoke the function above—openOptionsScreen—on a "click" event.

```
optionsBtn.addEventListener("click", this, "OpenOptionsScreen");
```

4. Finally, set initial focus to the Options button with the final line of code:

```
optionsBtn.focused = true;
```

5. Test the movie. The *Options* button should be selected or focused by default. Pressing the *Options* button should take the user to the *Options* screen, which is currently empty.

3.4 Adding the Difficulty Options Stepper

A game difficulty setting control will be added to the *Options Menu* using the *Options Stepper*. This component is a dynamically filled data driven element which displays a text field filled from a list of choices—in this case, the choices displayed are: Easy, Medium, Hard, and Insane. It uses two button components to step through each option in the list.

1. Go back to the CLIK_Components.fla file.

2. Right click the component *OptionsStepper* in the *Library* pane and select *Copy*.

3. Return to the original Flash document.

4. Select the keyframe on frame 10 of the *scene* layer (the *Options Menu* keyframe).

5. Paste the *OptionsStepper* into the *Library* pane.

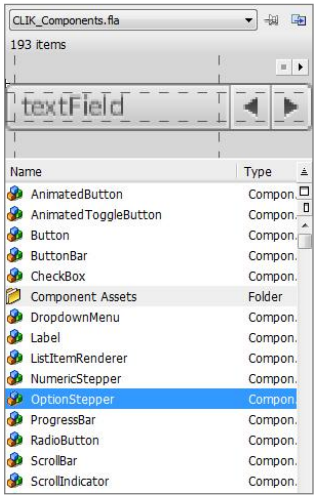

Figure 19: OptionStepper component in the library pane.

6. If the OptionsStepper has already been pasted onto the stage, skip this step. Otherwise, click and drag the OptionsStepper component from the *Library* pane onto the Stage.

7. Select the OptionsStepper, and change its instance name to 'difficultyOption'.

8. Click and drag a new *Label* component from the library to the Stage and position it to the left of the *Difficulty* control. This will represent the text that described the difficulty setting control.

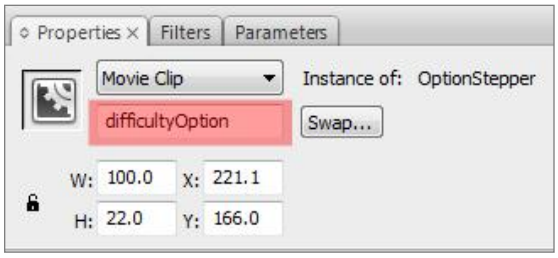

Figure 20: Name the Difficulty Control instance 'difficultyOption'.

9. Change the *text* field in the *Parameters* tab for the *Label* to 'Difficulty'.

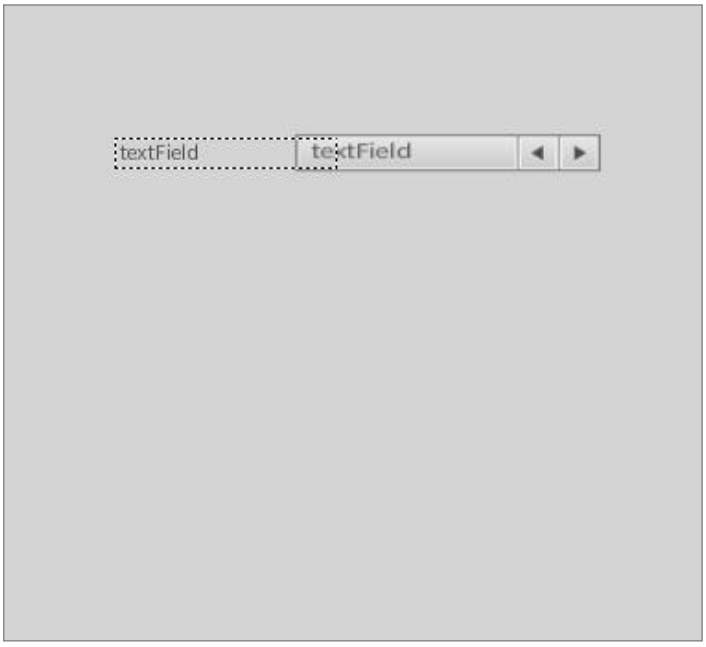

Figure 21: The final unskinned difficulty option stepper.

3.5 Adding the Video Settings Checkboxes and Radio Buttons

The video settings controls will make use of checkboxes to turn one of two settings on or off—in this case: *Bloom* and *Anti-Aliasing*—as well as a group of two radio buttons for changing how anti-aliasing works in the game. A checkbox is used when the desired functionality of a button is to toggle it as enabled or disabled; a radio button is used when grouping a set of options into a list in which only one of the grouped options may be enabled.

1. Create a new instance of the *Label* component below the *Difficulty* control, and set its text parameter to 'Video Settings'.

2. Select the CLIK_Components.fla file.

3. Right click the component *CheckBox* in the library and select *Copy*.

4. Go back to the working Flash document and paste the CheckBox into the *Library* pane.

5. Click and drag the CheckBox component from the *Library* pane onto the Stage.

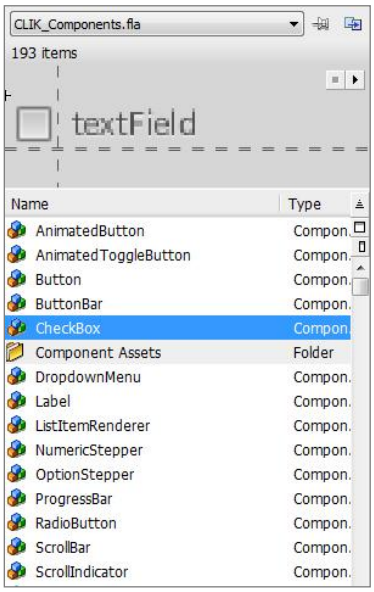

Figure 22: CheckBox component in the library pane.

6. Select the new instance, and change its instance name to 'bloomBtn'.

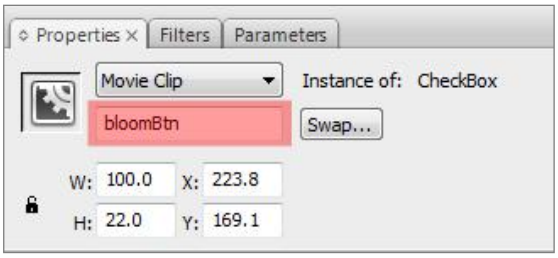

Figure 23: Name the bloom radio button instance 'bloomBtn'.

7. Change the *label* field under *Parameters* to 'Bloom'

8. Copy the instance, placing the copy directly under the Bloom checkbox, and change the new instance's name to 'aaBtn'.

9. Change the *label* field under *Parameters* to 'Anti-Aliasing'.

10. Open the file *RadioButton.fla*.

11. Copy and paste the RadioButton component into the working library.

12. Drag a new instance of the RadioButton onto the Stage under the Anti-Aliasing checkbox.

13. Change the instance name of the *RadioButton* to 'edgeaaBtn'.

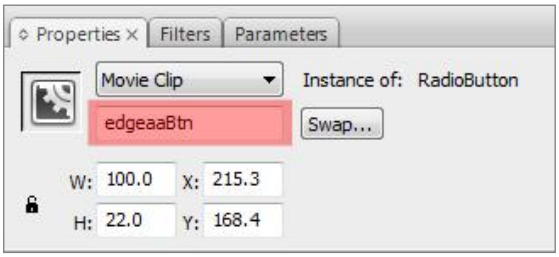

Figure 24: Name the Edge AA radiobutton instance 'edgeaaBtn'.

14. Change the *label* field under *Parameters* to 'Edge AA'.

15. Change the *group* field to 'aaGroup'.

16. Copy the instance, placing the copy directly under the Edge AA radio button, and change its instance name to 'hwaaBtn'.

17. Change the *label* field under *Parameters* to 'Hardware FSAA'.

18. Change the *group* field to 'aaGroup'.

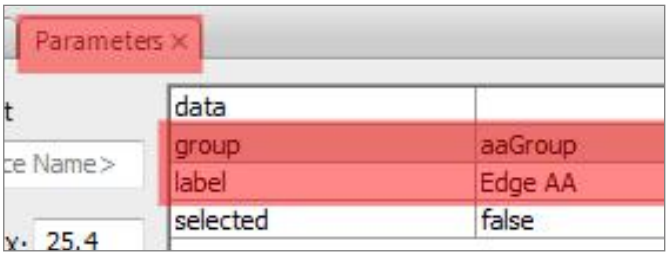

Figure 25: Set the radio button group to 'aaGroup'.

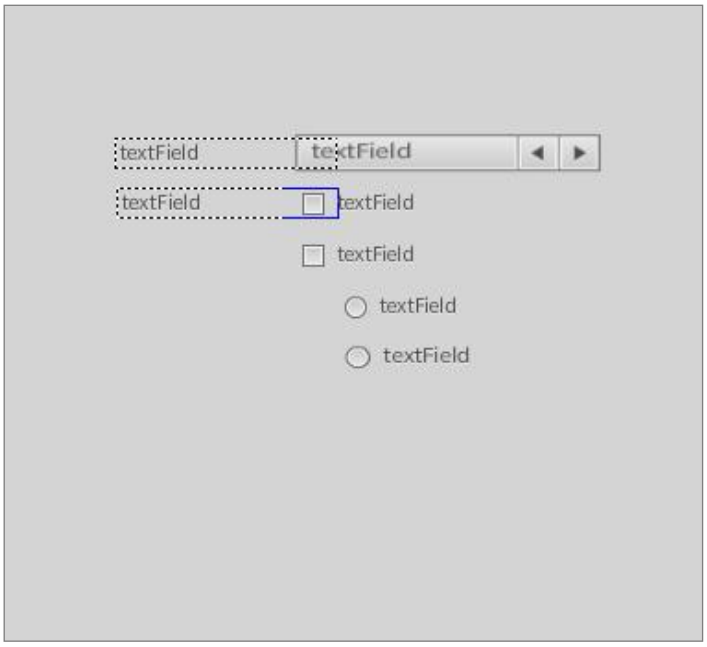

Figure 26: The final unskinned Video Settings added.

3.6 Adding the Sound Volume Slider

The sound volume will be added using a slider component. The slider makes use of a Track and a Thumb control. Both of these are simple button components. The Thumb is constrained to the length of the slider, so the user cannot move it off the slider to the left or right. It is also constrained vertically, so that it cannot move up or down.

1. Copy or create a new instance of the *Label* component below the *Video Settings Controls* on the *Stage*, and set its *text* parameter to 'Sound Volume'.

2. Select the CLIK_Components.fla file and copy and paste the *Slider* component into the working FLA document.

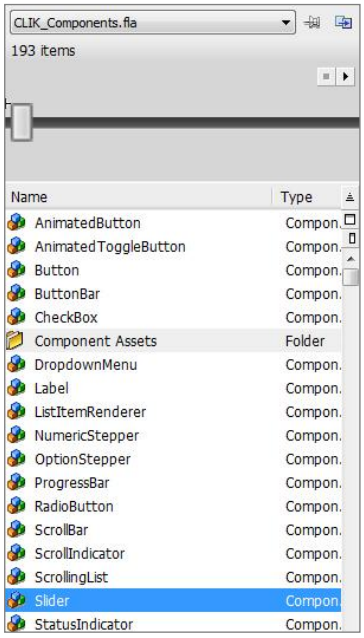

Figure 27: Slider component in the library pane.

3. Drag an instance of *Slider* onto the *Stage* next to the *Sound Volume* label and change its instance name to 'soundSlider'.

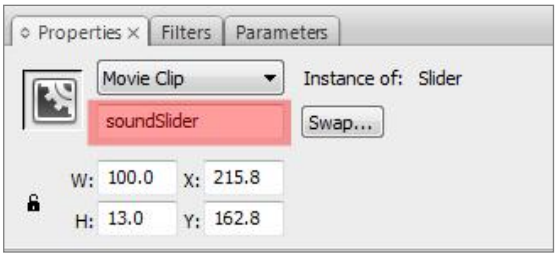

Figure 28: Name the volume slider instance 'soundSlider'.

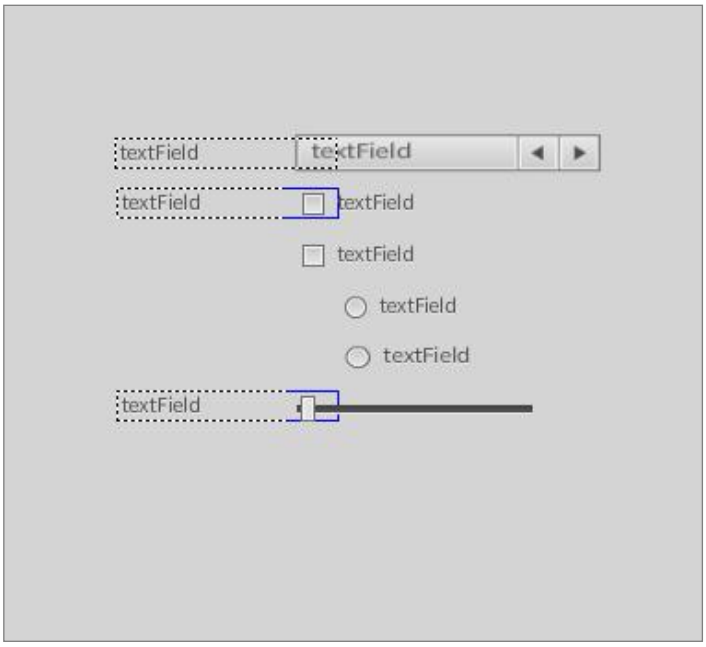

Figure 29: The final unskinned volume slider added.

3.7 Adding the OK and Cancel Buttons

The OK and Cancel buttons use the base button component.

1. Drag a button component to the stage from the library.
2. Select the button.
3. Click the *Parameter* tab of the button and set the *label* to 'OK'.
4. Change the instance name to 'okBtn'.
5. Copy the button on the Stage and change the instance name of the new button to 'cancelBtn'.
6. Click the *Parameters* tab for the new button copy, and set the *label* field to 'Cancel'.

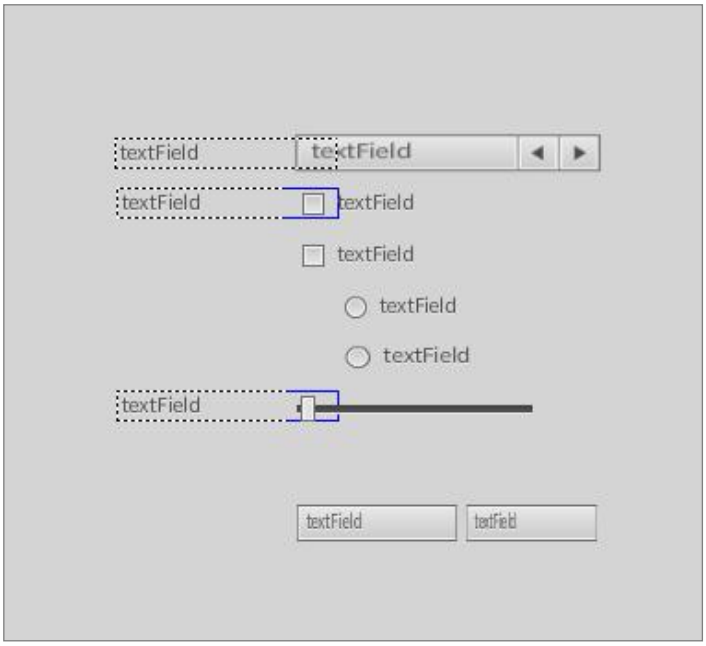

Figure 30: The final unskinned Options screen.

4. Adding Functionality with ActionScript

At this point, all the individual default skinned components have been created, but the menu has no real functionality. It's time to add some ActionScript. Bring up the *Actions* panel by pressing (F9), if it isn't already open.

If the tutorial code does not work, open the tutorial file and look over the code there to see what may have been entered incorrectly.

4.1 Adding Difficulty Levels to the Difficulty Options Stepper

In order for the difficulty stepper to display what difficulty it is set to and allow the user to change that setting, data must be assigned to it. This is done using the 'dataProvider' property of the difficultyOption component, and passing the desired data. In this case, the data to pass is a list of difficulties: Easy, Medium, Hard, and Insane.

1. Select the keyframe on frame 10 of the *actions* layer.
2. Click somewhere inside the *Actions* panel to begin entering the code.
3. Insert the following code above the "`stop();`" command:

```
difficultyOption.dataProvider = ["Easy", "Medium", "Hard", "Insane"];
```

4. Test the movie. The game difficulty setting option stepper control should now be functional — showing the four difficulties as you press the forward and reverse buttons.

4.2 Setting up the aaGroup Radio Button Group

The Anti-Aliasing radio buttons need to be grouped in code. The Anti-Aliasing radio buttons should be disabled (unchangeable) unless the user places a checkmark in the *Anti-Aliasing* checkbox.

1. Select the keyframe on frame 10 of the *actions* layer.

2. First, at the very top of the Actions panel enter the following line of code to provide access to the CLIK ButtonGroup class:

```
import gfx.controls.ButtonGroup;
```

3. Next, move down below the `difficultyOption.dataProvider` line of code, and create a new button group named 'aaGroup'. Assign edgeaaBtn and hwaaBtn to it:

```
var aaGroup:ButtonGroup = new ButtonGroup();
edgeaaBtn.group = hwaaBtn.group = aaGroup;
```

4. Create a function that will enable or disable the radio buttons:

```
function EnableAARadioButtons()
{
      edgeaaBtn.disabled = hwaaBtn.disabled = !aaBtn.selected;
}
```

5. Finally, add an event listener to the Anti-Aliasing checkbox that invokes the function above:

```
aaBtn.addEventListener("click", this, "EnableAARadioButtons");
```

6. Test the movie. By default, the video settings anti-aliasing radio buttons should now be disabled but be able to be enabled by clicking on the anti-aliasing checkbox.

4.3 Setting the Initial States of the Options

The options should retain their states when the user presses the *OK* button to return to the *Main Menu,* and they should ignore any changes if *Cancel* is pressed.

1. Create a new layer at the top of the timeline (above the *actions* layer) and rename it 'vars'.

2. Select frame 1 of the *var* layer.

3. Create a new object named 'options' which will hold the various options' states. Put the new object inside a conditional statement which tests to see if the object already exists:

```
if (!options) {
      var options:Object = {};
}
```

4. Select frame 10 of the *actions* layer.

5. Set the initial state of the difficultyOption. If the property options.selectedDifficulty has not yet been defined, set the difficultyOption's initial value to index 0 (which corresponds to 'Easy'), otherwise, set it to whatever options.selectedDifficulty is:

```
difficultyOption.selectedIndex = (options.selectedDifficulty) ?
                                  options.selectedDifficulty : 0;
```

6. Set the initial state of the bloomBtn and the aaBtn:

```
bloomBtn.selected = options.bloom;
aaBtn.selected = options.aa;
```

7. Set the edgeaaBtn and hwaaBtn to be disabled or enabled based upon the selected state of the aaBtn checkbox.

```
edgeaaBtn.disabled = hwaaBtn.disabled = !aaBtn.selected;
```

8. Set the initial state of the edgeaaBtn. If the property options.edgeaa has not yet been defined, set edgeaaBtn's selected state to true, otherwise set it to whatever options.edgeaa is.

```
edgeaaBtn.selected = (options.edgeaa) ? options.edgeaa : true;
```

9. Do the same for the initial state of the hwaaBtn, but set it to false if it is undefined.

```
hwaaBtn.selected = (options.hwaa) ? options.hwaa : false;
```

10. Set the initial state of the soundSlider.

```
soundSlider.value = options.soundVolume;
```

4.4 Exiting the Options Screen

The final bit of code will allow the user to exit the *Options Screen* and return to the *Main Menu* when either the *Cancel* button or the *OK* button is pressed.

1. Select the keyframe on frame 10 of the *actions* layer.

2. Insert the function that will take the user back to frame 1 (the *Main Menu*) when the OK button is pressed, after the code from section 4.3 and before the "`stop();`" command. The function sets (and saves) the properties of the options object so that when the user returns to the *Options Screen*, they will be the same.

```
function ReturnToMainMenu()
{
	options.selectedDifficulty = difficultyOption.selectedIndex;
	options.bloom = bloomBtn.selected;
	options.aa = aaBtn.selected;
	options.edgeaa = edgaaaBtn.selected;
	options.hwaa = hwaaBtn.selected;
	options.soundVolume = soundSlider.value;
	gotoAndPlay("mainMenu");
}
```

3. Insert the function that will take the user back to frame 1 (the *Main Menu*) when the Cancel button is pressed. This function does not save any changes made.

```
function CancelToMainMenu()
{
	gotoAndPlay("mainMenu");
}
```

4. Next listen for the user to click on the *OK* or *Cancel* buttons:

```
okBtn.addEventListener("click", this, "ReturnToMainMenu");
cancelBtn.addEventListener("click", this, "CancelToMainMenu");
```

5. Last of all, give default focus to the *Cancel* button.

```
cancelBtn.focused = true;
```

6. Test the movie. All *Options Screen* controls should work now.

5. Skinning the Menu

The final step in the process is to apply custom, artist-created skins to the menu and each UI component. Rather than describe how to skin every component, to avoid duplication, this document will describe how to skin a sample of the components used. Once the process is understood, skinning the rest of the components should follow the same workflow.

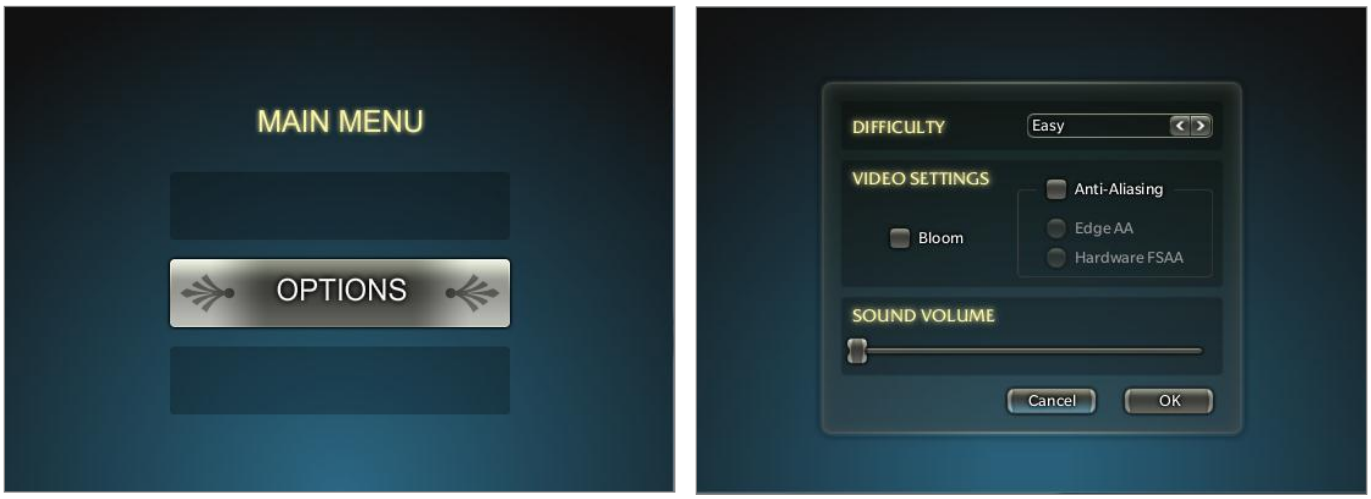

Figure 31: The final skinned menu.

5.1 Skinning the Options Button

The workflow presented here for Photoshop®, Illustrator® and Flash CS3, should work equally well in Photoshop, Illustrator and Flash CS4 or CS5.

Every object in a UI can have multiple states. In the case of a button, those states are Up (default), Over (when selected or moused over) and Down (when pressed). Each of these states should have a different piece of art to provide visual feedback to users. These states can be built in Photoshop or Illustrator. As a best practice, all of the art for each of the states of one component should be included in a single Photoshop or Illustrator file. Each of these different states should be a separate layer in the art file. Doing this will allow Flash to merge the layers in as keyframes on the timeline. Before importing the artwork, merge all related layers into a single layer for each state. As a best practice, keep separate working and importing art files where the working file contains the original layers, which may be needed for future editing and adjustments, and the importing file should have all layers properly merged for import into Flash. In the case of a simple button, the final importing art file should only have 3 layers — Up, Over, and Down.

1. Build each graphical state of the *Options* button in the same Photoshop Document (PSD) file.
    a. See the documentation for each component to find a list of the states that component needs, or refer to the unskinned component's *labels* layer on the timeline.
    b. **IMPORTANT:** Avoid creating duplicate layers for states that have identical content. In this case, all that is required are layers for up, over, and down.

2. Once completed, save the PSD. If multiple layers were used to create a state, then merge all the layers that are part of the same state into one layer. Merging can be accomplished first holding down the (CTRL) key, then pressing each layer to be merged in the *Layers* panel of Photoshop, and then right clicking on a selected layer and choosing *Merge Layers* from the popup menu.
    a. For instance, merge every layer that is part of the *over* state into an *over* layer.

3. Ensure that the layers are ordered so that the first state on the component's timeline in Flash — usually *up*—is at the bottom of the Photoshop *Layers* panel, and the last state on the timeline is at the top. This order is critical to how Flash will import each layer.

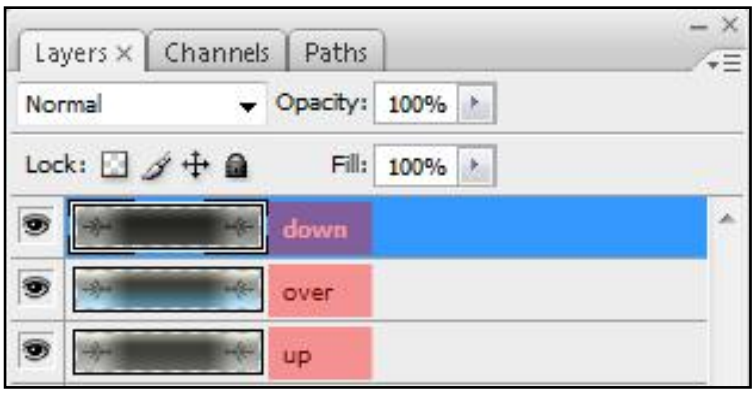

Figure 32: The options button Photoshop layers.

4. (Optional) Rename each layer so that it matches the state name in Flash: *up*, *over*, *down*, etc. This is purely for organizational purposes.

5. Save the file as a new PSD, appending '_mergedLayers' to the end of the file name. Do not overwrite the original PSD saved in step 2 unless the unmerged layers are no longer required.

6. In Flash, open the unskinned *Options* button so that its timeline is visible by double clicking on it on the stage.

7. Select the *button* layer on the timeline and delete all the frames by pressing on the layer name to select all frames, right clicking on a frame in the timeline, and choosing *Remove Frames* from the popup menu.

8. Add one new blank keyframe to the layer and select it. This is necessary to do step 9.

9. Click *File* in the top menu, and select 'Import', then 'Import to Stage' from the popup menus—or press (CTRL + R).
    a. If 'Import to Stage' is not available, be sure the layer you are attempting to import to is not locked. If the layer has a lock icon to the right of the layer name, just click the lock to toggle the layer unlocked.

10. Browse for and select the PSD or Adobe Illustrator (AI) file of the skinned UI element — in the case of this tutorial, a PSD file was used.

11. The Import dialogue will open. Ensure each layer to import is checked.
    a. **IMPORTANT:** The import dialogue does not show up when importing PSDs in Flash 8. However, it does in CS3 and CS4. If using Flash 8 and importing PSDs, it is necessary to modify the workflow beyond these instructions.

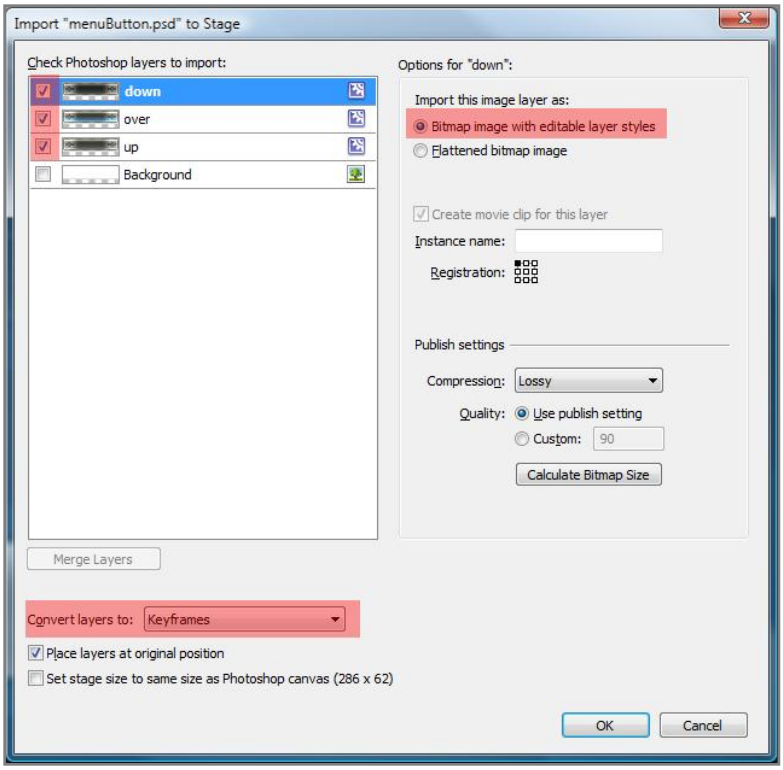

Figure 33: The import to stage dialogue.

12. Ensure the dropdown *Convert layers to:* is set to 'Keyframes'. This dropdown should be at the bottom left of the dialogue.

13. Click on each layer in the list and select *Bitmap image with editable layer styles* under the *Import this image layer as* heading.
    a. **IMPORTANT:** If this step is not performed on each layer, any layers with layer styles will not show up on the *Stage*—although they will be added to the library—and each layer beyond the first layer will be a flattened image containing all the other layers below it.

14. The layers will be imported as individual keyframes on the timeline. They will be in order of first state to last state.
    a. **IMPORTANT:** When importing a PSD file, the keyframes will be added to a new layer at the top of the timeline. Simply delete the old *button* layer, and rename the newly created layer *button*. Then click and drag the new *button* layer to the bottom of the list of layers.
    b. When importing an AI file, the keyframes will be added to the *button* layer, as long as it is selected before importing.

15. Click and drag each keyframe so that it lines up with the appropriate label; these are displayed on the *labels* layer of the timeline. Move the first keyframe into the first frame (*up*). Move the second keyframe into position beneath the *over* keyframe. Move the third and final keyframe into position beneath the *down* keyframe. Disabled has been left out for the purposes of this tutorial.

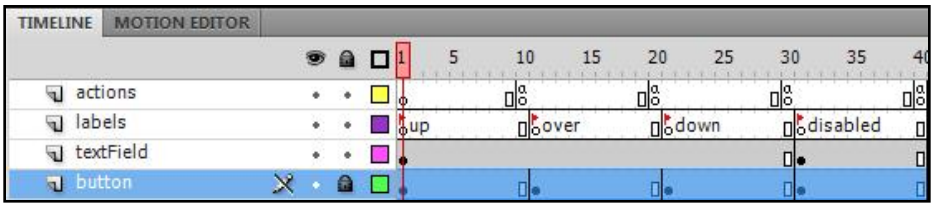

Figure 34: Adding the keyframes to the button layer.

16. Edit the font settings of the *textField* instance for each keyframe of the *Text* layer — up, over, down — if the button label should appear differently when the user interacts with the button. Otherwise, leave the textField the same for all states.

17. Test the movie. The *Options* button should now be fully skinned.

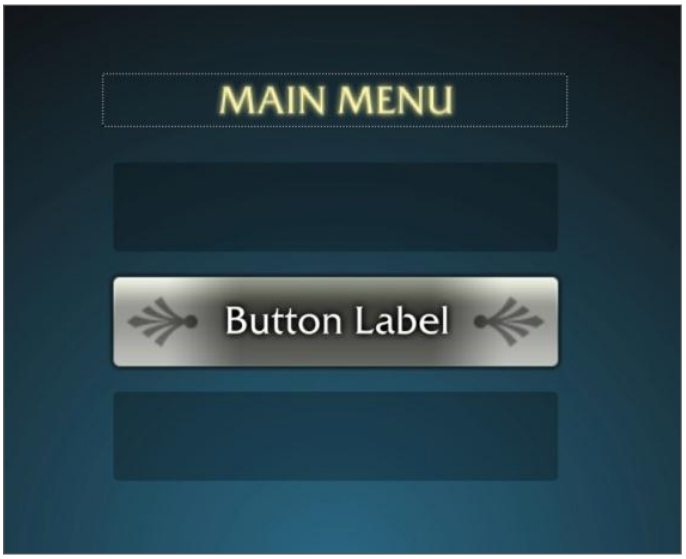

Figure 35: The final skinned options button and main menu.

5.2 Skinning the Volume Slider

The Volume slider is essentially made up of two buttons. One button represents the Thumb of the slider (the piece that slides from left to right). The other button represents the Track on which the Thumb moves and is constrained to. As this is the case, create each button—the Thumb and the Track—in much the same way as creating a normal button as described above in section 5.1, with a few caveats.

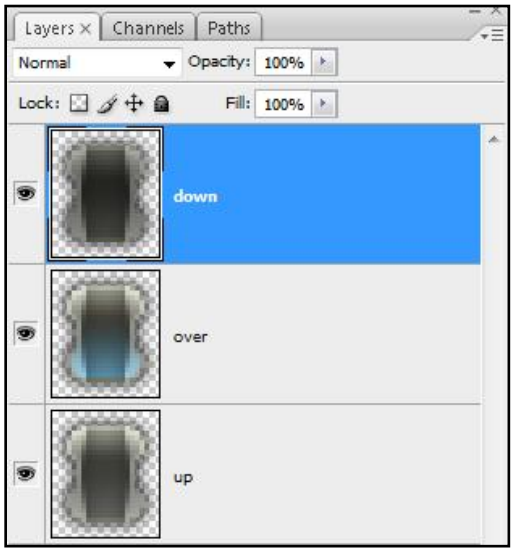

Figure 36: The volume slider thumb skin Photoshop layers.

1. Double click the *Volume* slider on the *Stage* to enter its timeline.

2. Double click the *Track* component inside the *slider* to enter its timeline.

3. Import the PSD or AI file created for the Track, and align the keyframes. The registration point for the Track should be at the far left of the graphic, and it should match the registration point of the Slider component exactly.

4. Return to the Slider timeline (one level up).

5. Double click the Thumb component to enter its timeline.

6. Import the Thumb PSD or AI file.
    a. **IMPORTANT:** When importing the Thumb skin, be sure to change the registration point for each layer to middle.

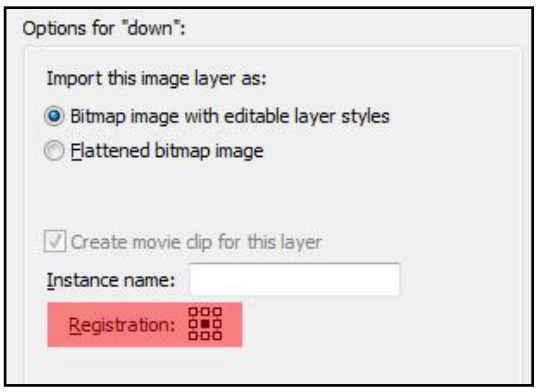

Figure 37: Set the registration point to middle.

7. Select the Thumb graphic on each keyframe, set its *X* property in the *Property* tab to '0.0'.
8. Return to the Slider timeline (one level up) and select the Thumb component and move it so that the top left corner of its bounding box matches up with the Slider component's registration point.

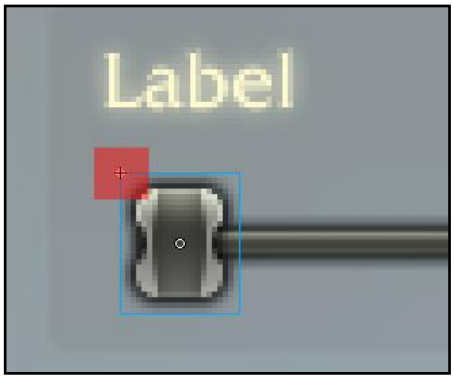

Figure 38: Align the Thumb's top left corner with the Slider's registration point.

9. Edit the font properties of the *textField* label.
10. Test the movie.

***Figure* 39: The final skinned volume slider.**

5.3 Skinning the Radio Buttons and Checkboxes

Radio Buttons and checkboxes are essentially button components with the following additional states: *selected_up*, *selected_over*, *selected_down*, *selected_out*, and *selected_disabled*. As such, it is not necessary to explain how to skin this component. Simply follow the instructions for a regular button in section 5.1; however, be sure to include the new states as part of the skinning process. Typically, a radio button includes a small dot inside the button to indicate it has been selected, and a checkbox includes a check mark to indicate the same. However, the final design does not have to follow this paradigm.

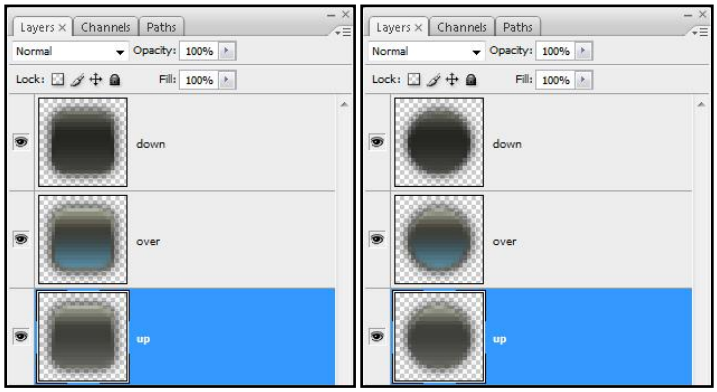

Figure 40: The radio button and checkbox skin Photoshop layers.

Finally, be sure to edit the font properties of the *textField* label instance.

Figure 41: The final skinned edge AA radio button.

5.4 Skinning the Difficulty Option Stepper

The Difficulty stepper is, once again, made up primarily of button components. Two buttons are required for this element: a left and a right button. A third graphic may also be used to represent the background/border of this component, but it is not necessary.

1. Create a button for left and right buttons using the methods described above in section 5.1.
    a. Alternatively, if using an identical image for both buttons, instead create only one button, and then mirror it in Flash.

2. On the Stage, double click the *Difficulty* option stepper to enter its timeline.

3. Next, if it fits the design, replace the bottom background/border image, seen below as a gray border around the component.

4. Select the *left* button by double clicking on it to enter its timeline.

5. Now simply import the new skinned PSD file into the *States* layer as described in section 5.1, and align the keyframes.

6. Repeat steps 4–5 for the right button.

7. Return to the parent timeline of the two buttons—the timeline of the Difficulty option stepper—and edit the font properties for the label named *textField*.

8. Test the movie.

Figure 42: The final skinned difficulty stepper.

6. Conclusion

Figure 43: The final skinned options menu.

Congratulations! This document presented one possible way to use Scaleform CLIK. There are many permutations, and most projects will present unique challenges. However, understanding the basic workflow of Scaleform CLIK in the way presented in this document should at the very least ensure a properly equipped and informed use of the Scaleform CLIK framework. For further education, it is recommended to read the more technically oriented documentation, to try out the demos that come with Scaleform CLIK, to browse over the source code for each component and read the comments therein. Great steps have been taken to ensure the end user has been provided with everything needed to be successful with Scaleform CLIK.

Demos and Source code can be found in the Scaleform SDK 4 directory at:
On Windows:

- Demos – *C:/Program Files/Scaleform/GFx SDK 4.4/Resources/AS2/CLIK/demos*
- Source code – *C:/Program Files/Scaleform/GFx SDK 4.4/Resources/AS2/CLIK/gfx*

On Mac:

- Demos – scaleform_gfx_4.4_macos/*Resources/AS2/CLIK/demos*
- Source code – scaleform_gfx_4.4_macos/*Resources/AS2/CLIK/gfx*

Additional Documents:

- Scaleform CLIK User Guide – A guide to Scaleform CLIK framework and the components with detailed implementation instructions.
- Getting Started with CLIK Buttons – A guide to the Button components in CLIK.