1. Accessible Interaction Screen

RESOURCES

Getting Help
WORKSHOP DESCRIPTION

Overview
This workshop will present information on making multimedia products accessible. This includes captioning video, making podcasts accessible, and design guidelines when using Flash and Dreamweaver.

Prerequisites
A basic knowledge of multimedia applications and processes are recommended.

Objectives
These training modules are intended to give you the knowledge needed to make multimedia accessible whether in preproduction, production or post-production.
Overview of Best Practices
Designing accessible content requires designers and developers to pay attention to the user experience. Since many Flash CS4 Professional designers and developers are not familiar with the capabilities of screen readers and other assistive technologies, these design guidelines provide several desirable practices for creating content in Flash CS4 Professional. These hints are intended to provide a helpful starting point for designers and developers striving to create accessible content in Flash CS4 Professional.

Text Equivalents
Using Microsoft Active Accessibility (MSAA), Adobe Flash Player 10 software exposes content created with Flash to the screen reader. Text contained within a movie is exposed by default; however, graphic elements are not exposed automatically. As in HTML, graphic elements require a text equivalent that is read by the screen reader in place of the image. Using ActionScript or the Accessibility panel in Flash CS4 Professional, designers and developers can easily assign text equivalents for elements in rich media content. In most cases, the greatest challenge is deciding when to use text equivalents and what they should say.

Text in Flash Player 10
By default, Flash Player 10 exposes all text elements to the screen reader user. Designers and developers do not need to make any modifications. In the example below, the text was added to the stage, and a screen reader would read the text as "Clearly tasty!" without any developer effort.

Adding Text Equivalents
Now, take a look at another example that will require a text equivalent. In the example below, the Adobe Flash logo is displayed. The entire logo is not actually text, but a graphic.
Since the logo is not text, a screen reader will not read it. In this case, a text equivalent should be provided. To add a text equivalent, use the Accessibility panel. First, make sure the logo is selected. Before a text equivalent is added to this item, the object must be saved as a symbol in the library. Since text equivalents are not supported for graphic symbols, save the object as a movie symbol or a button.

To bring the Accessibility panel to the front, press Shift+F11.

Notice the two fields titled "Name" and "Description." In general terms, the Name field is used for shorter text equivalents, and the Description field is used for longer ones. (This parallels the use of the alt and longdesc attributes in HTML.) In more practical terms, the difference is less important since screen readers read both by default.

Deciding on the proper text equivalent is not always easy. In most cases, it is wise to consider the purpose of the image rather than giving it a merely literal description. In the example above, a description may read, "A plastic bottle containing a clear liquid." This will not be particularly helpful information to a screen reader user. In contrast, the text "Aquo Clear" might be a more meaningful equivalent for the logo since it is part of an advertising feature for this product.

In this case, a description would not likely be necessary. Descriptions are best used when the text equivalent needs to be longer than about 50 characters. One thing to keep in mind is that the description is generally read following the name. If the content does not flow when the two are read together, the designer/developer may choose not to use a name, only a description of the object.
**Accessible Movie Elements**

By default, Flash CS4 Professional can make text, input text fields, buttons, movie clips, and even entire movies accessible. While text elements do not require modification by the designer/developer, other elements do. Designers and developers can use the Accessibility panel in Flash CS4 Professional to add a text equivalent or even to hide elements from assistive technologies.

In the example below, the movie shows a moon orbiting a planet. In this case, it is not appropriate to provide a text equivalent for the moon and the planet separately. Instead, the relationship between these two elements needs to be conveyed.

To provide a single text equivalent for the entire movie, first make sure that nothing in the movie is selected. In the Name field, add the text equivalent "Moon orbiting planet." The name may differ depending on the context, but this equivalent will suffice for this example. Note that the text equivalent conveys not only the contents of the movie but also how the separate elements relate to one another.

To group the contents of the movie together formally, deselect Make Child Objects Accessible. This step tells the screen reader to use this text equivalent for the entire movie.

---

**Animation**

Adobe® Flash® CS4 Professional software provides a powerful set of tools for creating animations and interactive content and applications. These tools allow designers and developers to create demonstrations of complex visual relationships, engaging interactions in buttons and
movies, and attractive interfaces in websites and web applications. Most assistive technologies, however, are not designed with these types of animations in mind. While working in Flash CS4 Professional, designers and developers need to be mindful of the ways in which assistive technologies such as screen readers will interact with their movies. Find out how to optimize accessibility when creating animation:

**Understanding How Assistive Technologies Handle Animation**

When a screen reader encounters content created with Flash, it loads the current state of the movie and notifies the user. With the Window-Eyes screen reader, the user hears, "Loading...load done." Once a piece of content has been read, the screen reader moves on to read other parts of the content and the rest of the page.

A unique feature of content created with Flash is that it may change over time. As the content changes, Adobe Flash Player 10 software sends a signal to the screen reader notifying it that there has been a change. When the screen reader receives this notification, it automatically returns to the top of the page and begins reading it again.

The following example illustrates the serious implications of content created without consideration for users of screen readers. A poorly designed banner ad placed at the top of the page might loop constantly through a few frames. When Flash Player encounters this banner, it will send repeated notifications to the screen reader of changes in the content, and the screen reader will continually return to the top of the page. This problem can seriously erode the experience for screen reader users.

To address this specific issue, Adobe worked with GW Micro to create a Halt Flash Events keystroke (Alt+Shift+M) for the Window-Eyes screen reader. This keystroke allows a screen reader user to suspend Flash notifications on the page. Pressing the keystroke again allows the user to resume Flash notifications.

**Providing Text Equivalents for Entire Movies**

A better solution for handling animation is simply to hide the animation from assistive technologies. Flash allows designers and developers to assign a text equivalent for an entire movie or for a collection of objects within a movie. Designers and developers might choose to provide a text equivalent for content for one of two reasons.

Animations are often used to illustrate visual relationships among elements on the screen. Adding text equivalents to the individual elements may not provide a sufficient description of the relationships among the elements. For example, in an animation of the solar system, a designer/developer might add text equivalents to the planets; however, these text equivalents would not convey information about how the planets move in relation to one another. A text equivalent for the entire movie could provide a better description of this relationship.
Making Child Objects
Notice in the example above that the option "Make child objects accessible" has been deselected. This serves two valuable purposes. First, it serves to group these elements formally as a single element. From a testing standpoint, this makes the movie easier to evaluate for accessibility. Second and more important, it hides the motion from screen readers. By making the child objects of this sample movie inaccessible, the designer/developer is effectively telling the screen reader to ignore everything within the movie. As a result, this movie will not send the constant updates to the screen reader that cause the screen reader to keep looping.

Avoiding Constant Motion
An important consideration for accessible animation relates to constant motion on the screen. While it is popular to include motion in movies as part of transitions and loading sequences, it is important that these animations settle to a static screen once the page loads. For people with learning disabilities, motion on the screen can be distracting and might even make other elements unreadable.

Components and Accessibility
Adobe® Flash® CS4 Professional software helps to accelerate accessible application development with a core set of user interface components. These components can automate many of the most common accessibility practices related to labeling, keyboard access, and testing and help to ensure a consistent user experience across rich applications created with Flash CS4 Professional. The set of accessible components available with Flash CS4 Professional includes:
For each ActionScript® 3.0 component, the designer or developer need only enable the accessibility object by using the command enableAccessibility(). This includes the accessibility object with the component as the movie is compiled. Because there is no simple means of removing an object once it has been added to the component, these options are turned off by default. It is therefore very important that the designer or developer enable accessibility for each component. This step needs to be done only once for each component; it is not necessary to enable accessibility for each instance of a component. Here is the sample code added for the checkbox component:

```actionscript
Import fl.accessibility.CheckBoxAccImpl;
    CheckBoxAccImpl.enableAccessibility();
```

It is best to attach this script to the first frame in the movie.

---

**Reading Order**

Controlling the reading order of an movie created with Adobe® Flash® software is perhaps the single most important and challenging aspect of accessible design. The default reading order of such a movie does not follow a predictable left to right, top to bottom order. As a result, its contents can be difficult to understand. In the example below, based on the visual presentation of the alphabet in three rows, it would be natural to expect the reading order to follow an alphabetical order.
However, the actual reading order jumps between letters in each row, resulting in the following order:

![Image]

There are three strategies for controlling reading order. The simplest is to keep the physical size of the movie small. The second strategy requires controlling the reading order using ActionScript®. A third strategy places a duplicate version of content offstage in a single column.

It is very important that the reading order of a movie be tested from the beginning of the development process using a screen reader. This can facilitate understanding when a problem has been introduced and how to rectify the issue. It is significantly more work to modify the reading order of a movie once it has been completed.

There are at least three means of controlling reading order:

**Limiting the Size of the Stage**
A small movie that is less than 300 pixels wide and consists of a single column or a single row of objects does not generally require any specific control over the reading order. Examples might include small animations or applications that pop up in a separate window, a navigation bar that consists of a single row, or an application that consists of a single column.

**Controlling Reading Order Using ActionScript**
The most precise means of controlling reading order is to use ActionScript. This method allows the designer to control the reading order using the .tabindex property in ActionScript. There is no distinction in ActionScript between reading order and tab order. However, when ActionScript is used to control the reading order of a movie, all instances within the movie must be included in the list of .tabindex values, including all text fields and decorative elements.

*Include an instance name with every instance over the life of the movie*

In controlling the reading order, it is important to ensure that every instance on the stage has an instance name. This includes all text, movie clip, and button symbols as well as all components over the life of the movie.
1. DO NOT USE STATIC TEXT
   Since it is not possible to provide an instance name to static text objects, a single instance of static text will result in the entire reading order reverting to the default. Controlling the reading order using ActionScript requires the use of dynamic text fields. This has implications for the font used in the application and potentially affects the overall file size. Learn more about handling font symbols in Flash.

2. INCLUDE OFFSTAGE OR OBSCURED ELEMENTS
   The list of .tabindex values must include all instances over the life of the movie, including elements that are not visible and that sit offstage or are hidden under another instance. If these elements should be obscured from a screen reader user, the visible property should be set to false or the .silent property should be set to true. Also, elements not visible at the start of the movie that will be visible later must be included in the list of .tabindex values.

3. CONTROL READING ORDER WHEN LOADING SWF FILES AT RUNTIME
   In cases where a series of child SWF files is loaded into a parent movie, the .tabindex values must be listed in the child movie clip. However, it is important that the values list in the reading order of each child SWF file be unique. For example, if two child movies loaded into a parent movie each have three elements with .tabindex values of 1, 2, and 3, the screen reader will read the first value of the first movie loaded and then the first value of the second movie loaded. Next, the screen reader will read the second value of the first movie clip loaded and then the second value of the second movie clip loaded and so on. For the screen reader to read the contents of the first movie followed by the contents of the second movie, the list of .tabindex values for the first movie should be 1, 2, 3, while the list of values for the second movie should be 4, 5, 6. These values need not be sequential, but they should be unique.

Controlling Reading Order Using Offstage Content
   In cases where the application relies on highly dynamic content, it can be difficult to specify the reading order in advance. In this very small set of applications, it is possible to control the reading order by placing a second set of content offstage in a single column. The onstage content is rendered as inaccessible to prevent the screen reader from reading it. The second set of content is positioned in a column so the reading order does not jump between unrelated elements.

   There are two primary drawbacks to this method. First, it increases the number of objects in the movie, affecting file size and movie performance. Second, users of screen magnifiers may find the offstage content conflicts with the onstage visual content with which they are working. Screen magnifiers not only increase the size of elements on the stage, but they also move them to the center. If the onstage content is inaccessible, offstage content will draw the focus of the screen magnifier, creating a confusing visual experience. This method is often used in conjunction with screen reader detection, but it is important to consider that users may be using a screen reader and a magnifier simultaneously.
1. SCREEN READER DETECTION

Since this case is intended to benefit screen reader users alone, this method is frequently used in conjunction with screen reader detection. Flash has a unique advantage over JavaScript in that it can use Microsoft Active Accessibility (MSAA) to detect the presence of a screen reader. The method Accessibility.isActive() will return a value of true if a screen reader is present and it is currently focused on the content created with Flash. It is important that this method not be called in the first second or so in the life of the movie, or it could return a false negative. Rather than calling this method in the first frame of a movie, many designers attach this method to the first button in the movie.

2. BUILDING OFFSTAGE CONTENT

If Accessibility.isActive() returns a value of true, the contents onscreen are set to be inaccessible. Perhaps the easiest means of accomplishing this is to place all of the onstage content in a single movie clip and then set the .silent property of the clip to true. It is preferable to use .silent as opposed to altering the visibility of the movie clip since people relying on screen readers might be working with a sighted person who is visually relying on the onstage content.

Next, the offstage content is loaded into a single column. This column need not match the height of the movie. It is important that as changes are made onstage that offstage content is updated at the same time.

Keyboard Access

It is important that all controls that can be manipulated via the mouse also be accessible via the keyboard. This is intended to support users with mobility impairments as well as screen reader users. Adobe® Flash® Player software facilitates keyboard access on its own by automatically making mouse-defined events accessible via the keyboard. However, two specific techniques commonly used among Flash designers should be avoided. In addition, designers should add keyboard shortcuts to facilitate keyboard access in complex applications. Finally, designers should be aware of an issue with Flash Player 10 and earlier in pages that blend HTML and content created with Flash.

```actionscript
on (click) {
    getURL(index.html);
}
```

For example, the ActionScript® 2.0 script shown above might be used to open a web page. It is directly associated with the instance of the movie clip used as a button. This script should instead be placed in a frame, likely the first frame, of the movie. The revised script, converted to ActionScript 3.0, could be as follows. Note that this script uses the ActionScript MouseEvent, but this is method is also operational from the keyboard.

```actionscript
function gotoAdobeSite(event:MouseEvent):void
{
```

Accessible Technology Initiative Program: Flash Accessibility Used with Permission from Adobe, 2009 10
null

var adobeURL:URLRequest = new URLRequest("http://www.adobe.com/");
navigateToURL(adobeURL);
}

home_mc.addEventListener(MouseEvent.CLICK, gotoAdobeSite);

1. AVOID EMPTY HIT AREAS

a. Hit areas are empty button clips with a shape defined in the hit state. These allow designers to reuse a single library of objects repeatedly by placing them over text objects and varying only the scripts used. The problem with this technique is that screen readers assume that if the contents of the up state of a button clip is empty, then it is not a button at all. The solution to this issue is simple. When a transparent movie clip is placed in the up state, screen readers will recognize the button and allow the user to activate it.

2. ASSIGN KEYBOARD SHORTCUTS FOR MOST ESSENTIAL CONTROLS

a. In complex applications with multiple controls, it is helpful for users to navigate the application using keyboard shortcuts. For many users with mobility impairments, pressing keys may be difficult. Using keyboard shortcuts reduces the number of keystrokes required to perform important tasks.

b. Using the shortcut field in the Accessibility panel or the .shortcut property in ActionScript is not sufficient for this purpose. Creating a keyboard shortcut requires that a listener event be defined and a script associated with that listener. The shortcut field merely announces a shortcut value via Microsoft Active Accessibility (MSAA). It does not create the listener. Moreover, no screen readers support this feature in MSAA at this time.

Captions

Adobe® Flash® software is frequently used to deliver audio and video content. Any audio that contains substantive content should include a synchronized text equivalent in the form of captions.

Standards Support

Adobe Flash CS4 Professional software provides a component that allows developers to easily add captions to either FLV or H.264 video in Flash. The FLVPlaybackCaptioning component enables captioning for the FLVPlayback component. The FLVPlaybackCaptioning component uses a World Wide Web Consortium (W3C) Timed Text XML file (DFXP) and adds those captions to the FLVPlayback component with which the FLVPlaybackCaptioning component is paired.
Adobe has worked with many caption tool vendors and service providers to encourage the support of DFXP. Several options are available for developers who are interested in providing captions, including do-it-yourself tools and vendors that can provide DFXP caption data files for those that want to outsource this important and detailed work. A list of tools and vendors that support DFXP is available to help developers choose the method best suited to each situation.

You use the FLVPlaybackCaptioning component with one or more FLVPlayback components. In the simplest scenario, a FLVPlayback component is dragged onto the stage, an FLVPlaybackCaptioning component is dragged onto the same stage, the caption data file URL is identified, and showCaptions is set to true so that the captions are on by default.

Figure 1 View of the Flash FLVPlayback Captioning component and settings

In Figure 1, the caption file is set in the source parameter. Additional parameters to customize your FLVPlayback captioning can be set, but in this example simply saving and publishing the SWF file results in captions as shown in Figure 2.
Content providers may want to deliver captions in a different location onscreen. The FLVPlaybackCaptioning component allows developers to identify a separate text field to display the captions. In the screenshot below, a new dynamic text field not only displays the captions in a new location, but also takes advantage of the ability of Flash to embed font information so that even users who do not own a font can view captions in the font selected by the developer.

**Support For Embedded Cue Points**

The FLVPlaybackCaptioning component also displays caption data contained in embedded cue points. One advantage of using FLV cue points is that caption data exists within the FLV file, reducing the number of files to track. Captionate is a captioning tool that allows authors to add cue points to FLV files.

**International Support**

The FLVPlaybackCaptioning component supports Unicode character display. This allows caption display for non-Western languages and for special characters such as the musical note that are familiar to viewers of captioning.
Accessible Interface

The captioning support in Flash is complemented by support for keyboard and screen reader interaction with the video playback controls, making Flash the most accessible way to deliver video on the web. For more information about accessible video controls in Flash, read the video design guidelines document.

Control Over Audio Playback

Music and audio that plays as the site loads presents a serious challenge to screen reader users. The audio from a movie created with Adobe® Flash® software can interfere with the end user's ability to hear the contents of a movie using a screen reader. As a result, it is important to make sure that the user has control over when music is played.

The simplest strategy for handling audio playback is simply to allow the end user to control audio with a play and pause button. Allowing the end user to initiate audio provides the experience of the audio without creating an additional hurdle.

A more advanced strategy for controlling playback relies on the use of keyboard shortcuts for audio playback. Providing global keystrokes that allow the user to control the audio can greatly enhance the experience for end users. Here are several controls to consider.

- Play and Pause
- Mute
- Volume

Play and Pause is typically controlled using a single keystroke, such as the letter "p," as a toggle. The first time the button is pressed, the audio starts to play. The second time the button is pressed, the audio is paused. A mute button, such as the letter "m" or the number "0," silences but does not stop the audio. This provides the screen reader user with the opportunity to listen to the screen reader temporarily without stopping the audio. When utilizing shortcut keys, it is important to consider potential conflicts with assistive technology keystrokes, such as quick navigation keys used by the JAWS screen reader.

Finally, volume controls allow the user to quietly play the audio in the background while still listening to the screen reader. This is most appropriate in cases where the audio does not require the focused attention of the user, as in the case of a music stream. Using the ability of Flash to detect assistive technologies, a developer might choose to automatically reduce or mute the audio for a movie to simplify the experience, while granting the user the flexibility to adjust the audio level if desired.
Accessible Video Controls
Adobe® Flash® CS4 Professional software provides industry-leading support for accessible video. Providing captions for video solves accessibility challenges for people who are deaf or hard-of-hearing, but people who are blind or who have low vision or other physical disabilities need the video playback controls to be keyboard accessible and to function properly with assistive technologies such as screen readers and screen magnifiers. Flash CS4 Professional offers improvements to the FLVPlayback video component that make the default player controls accessible automatically, without any coding required by the developer.

SIMPLE IMPLEMENTATION
All skins provided by Flash CS4 Professional support keyboard and screen reader access, so developers simply insert the FLVPlayback component on the stage in Flash — no configuration is needed.

ACCESSIBLE INTERACTION
Users who rely on keyboard access can utilize a variety of familiar shortcuts to interact with the video controls. Buttons such as Play/Pause, Stop, Rewind, Mute, and Closed Captions can be tabbed to and activated with the spacebar key. Slider controls such as the volume and playhead position controls can be accessed via the arrow keys, and the Home and End keys can be used to go directly to the beginning or end of the range. The volume slider also accepts numeric keys to set the playback audio levels in one quick step.

Users who use screen readers will find appropriate names for the video player controls that ensure that each control's purpose is clear. Some controls require that the screen reader is in form interaction mode (which is Forms mode in JAWS or Browse Off mode in Window-Eyes) because screen readers often utilize arrow keys and other shortcuts for other purposes. For example, many screen readers use right and left arrow keys to read individual letters in text and control labels, so these keys are not available outside of form interaction mode, as expected by users.

RESOURCES

Getting Help
Accessibility Help Listserv
accessibilityhelp@lists.calstate.edu