ThumbnailList Component

Technical Documentation

Thumbnail List is a navigation component that enables you to load a set of image files, swf files or symbols from library, fed from an xml file, through which you can scroll with the help of scroll bars, mouse movement or mouse dragging. It's an excellent component when used in photo galleries for example, because it was developed to adjust on any environment you could think of, technical and artistic, due to the roll over effects on the thumbs that will add creativity and style to your projects and due to the resizing types supported by the thumbs that will enable you to display the content in the best way possible. Also, the component could be used in portofolio presentations, since, through the xml file, each thumb could take you to a different web page.

ThumbnailList can be accessed from Components panel under UI Pro Components – jumpEYE.
Note: Works with Macromedia Flash 8.

Properties in the Parameters panel

background - a boolean value which sets a background to the component if the selected value is "true", or leaves the component without a background if the selected value is "false" - default is "true"

border - a boolean value which sets a border to the component if the selected value is "true", or leaves the component without a border if the selected value is "false" - default is "true"

builtInPreloader - the preloader used when the component has a display effect set; possible values are "none" (no preloader), "bar" (bar like preloader) and "circle" (a circle like preloader) - default is "bar"

displayEffect - the effect applied on the component when the thumbs are first displayed - default is "none"; possible values:
- "none": no effect is applied
- "fade in": thumbs are fading in, all at once
- "ordered show": the thumbs are faded in, one by one, starting with the first one in the xml file
- "random show": the thumbs are faded in, one by one, in a random order
- "slide from top": the thumbs slide from top, all at once

- “drop thumbs”: the thumbs are slided from top, one by one, starting with the first one in the xml file
- “random drop thumbs”: the thumbs are slided from top, one by one, in a random order

inverseRollOverEffect - a boolean value which inverses the roll over effect on the thumbs, if the value is “true” - default is “false”

matrix - the size of the thumb matrix - default is {lines:4, columns:4}; a value of 0 means “autoarrange” (it can be used for the lines, for the columns or for both); if the number of images in the xml file is greater or less than the number obtained by multiplying the number of lines and columns, the number of lines will be recalculated so that all the images will be loaded

navigation - the type of navigation in the component; possible values are: “scroll bars”, "mouse movement” and “mouse drag” - default is “scroll bars”

rollOverEffect - the effect applied on the thumbs when the mouse rolls over them; possible values: “none”, “alpha”, “black&white”, “blur”, “brigthness”, “colorLight”, “distortion”, and “scale” - default is "none”

targetDisplay - link to an external loader in which the images will be displayed if the user clicks on the thumbs or if the slide show is started

thumbBorderSize - the size of the thumbs' border – 0 for no border – default is 5

thumbHeight - height of the thumbs – default is 100

thumbSpacing - spacing between thumbs – default is 10

thumbWidth - width of the thumbs – default is 100

xmlPath - the path to the xml file containing the image list

Properties in the Component Inspector

backgroundColor - the color for the component's background – default is gray (#CCCCCC)

baseUrl - property that you can use in case you want to complete the path to the thumbnails in the xml file and let the attribute "thumbnail" with only the name of the file

borderColor - the color for the component's border – default is black (#000000)

crossCursorColor - the color of the cross cursor used when navigation is “mouse drag” - default is black (#000000)

effectAmount - the amount of effect **applied on the thumbs** – if value is 0, a default value is used for each effect

effectTimeIn - the time in which the effect **is applied on the thumbs**, when the mouse rolls over them

effectTimeOut - the time in which the effect is **applied on the thumbs**, when the mouse

rolls out of them
keepScrollButtonSize - if "true", the scroll button will not be resized, it will maintain it's original size from the movie clip in the library - default is "false"
preloaderColor - the color for the built-in preloader – default is black (#000000)

remainActiveOnPress - the possibility for the clicked thumb to remain in an active state (the rollOver effect will remain applied) until the user clicks another thumb – default is "true"

scrollAreaColor - the color for the scroll bars' s background – default is orange (#FF9900)

thumbBorderColor - the color of the thumbs' border – default is white (0xFFFFFF)

thumbBorderCornerRadius - the corner radius of the thumbs' border – 0 for normal corners, above 0 for rounded corners – default is 0

thumbResizeType - the resize type for the image or clip loaded into the thumb; possible values: "borderscale", "scale" and "resize" - default is "resize"

Properties available from ActionScript

scaleShadow - a boolean value which sets a shadow to the thumbs when the onRollOverEffect is "scale", if the value is set to "true", or leaves the thumbs without a shadow, if the value is set to "false" - default is "true"

horizontalScrollerSize - the height of the default drawn horizontal scroll bar – default is 15

verticalScrollerSize - the width of the default drawn vertical scroll bar – default is 15

Methods

load(path:String) - sets the path to the xml file and start loading and displaying the images or clips
parameters:
- path:String – the path to the xml file containing the list of images or clips

setSize(newWidth:Number, newHeight:Number) - sets the size of the component
parameters:
- newWidth:Number – the new width for the component
- newHeight:Number – the new height for the component

startSlideshow(interval:Number) - starts the slide show
parameters:
- interval:Number – the number of seconds at which the component broadcasts a message with the current image in the slide show (if there is a loader component set as targetDisplay, the images will be loaded into the specified loader)

pauseSlideshow() - pauses the slide show
resumeSlideshow() - resumes the slide show

stopSlideshow() - stops the slide show

Events

onLoadXml(success:Boolean) - returns a boolean value indicating the success of the xml file loading operation: true for successful loading and false if an error has occurred during the operation

onLoadProgress(imagesLoaded:Number, totalImages:Number) - broadcasted during the loading process of the images and returns the number of images loaded so far and the total number of images to be loaded

onLoadComplete(succes:Boolean, totalThumbnails:Number, firstThumb:Object) - returns a boolean value indicating the success of the image loading operation: true if all the images have been loaded successfully and false if an error occurred during the operation, the number of total thumbnails displayed and an object containing the attributes corresponding to the first xml node in the file

onRollOverThumb(xmlNode:Object, thumb:MovieClip, clipIndex:Number) - returns an object containing the attributes corresponding to the xml node of the image, the thumb on which the roll over happened and the index of the clip

onRollOutThumb(xmlNode:Object, thumb:MovieClip, clipIndex:Number) - returns an object containing the attributes corresponding to the xml node of the image, the thumb on which the roll out happened and the index of the clip

onPressThumb(xmlNode:Object, thumb:MovieClip, clipIndex:Number) - returns an object containing the attributes corresponding to the xml node of the image, the thumb on which the user pressed and the index of the clip

onReleaseThumb(xmlNode:Object, thumb:MovieClip, clipIndex:Number) - returns an object containing the attributes corresponding to the xml node of the image, the thumb which the user released and the index of the clip

onReleaseOutsideThumb(xmlNode:Object, thumb:MovieClip, clipIndex:Number) - returns an object containing the attributes corresponding to the xml node of the image, the thumb which the user released and the index of the clip

onDragOutThumb(xmlNode:Object, thumb:MovieClip, clipIndex:Number) - returns an object containing the attributes corresponding to the xml node of the image, the thumb that has been dragged out and the index of the clip

onSlideshowChange(xmlNode:Object, thumb:MovieClip, clipIndex:Number) - returns an object containing the attributes corresponding to the xml node of the image, the current thumb in the slide show and the index of the current clip

Notes

The **xml file** contains a list of thumbs that will be displayed in the component. Each node of the xml document contains the following attributes:

- thumbnail (required) – contains the image or swf file to be displayed in the thumbs, along with the local path or url path to that file, if any; also may contain a linkage id of a movie clip from the library;
- large (optional) – the path to a large image corresponding to the thumbnail, to be used in slides hows for example
- description (optional) – a description of the thumb's content.
- url (optional) – a link to a web page which will be loaded when the user presses a thumb
- target (optional) – the target window in which the web page mentioned in the url attribute will be loaded; possible values: "**_blank**" - opens the web page into a new window of the browser; "**_self**" - opens the web page in the same browser window; "**_parent**" - specifies the parent of the current frame; "**_top**" - specifies the top-level frame in the current window

All the attributes are broadcasted as parameters of the following events: onRollOverThumb, onRollOutThumb, onPressThumb, onReleaseThumb, onReleaseOutsideThumb, onDragOutThumb and onSlideshowChange. In case the xml file contains linkage ids to movie clips from the library, please make sure that all those movie clips will exist in the library, or else, if the component has a displayEffect set on it, the thumbs will not be displayed.

RollOverEffects:

- **alpha** – values: 1..70; minTime = 20; maxTime = 50;
- **black&white** – values: - ; minTime = 20; maxTime = 50;
- **blur** – values: 1..10; minTime = 20; maxTime = 100;
- **brightness** – values: -80..-1, 1..80; minTime = 20; maxTime = 50;
- **colorLight** – values: 1..100; minTime = 30; maxTime = 100;
- **distortion** – values: 1..50; minTime = 10; maxTime = 100;
- **scale** – values: 1..20; minTime = 20; maxTime = 50.

The scroll bars are drawn internally using the drawing API, if there are no movie clips available in the library to be used as scroll bar components. To use your own movie clips as scroll bar components, you'll have to add to the library movie clips with the next linkage ids: "hsLeftButton", "hsRightButton", "hsScrollButton", "hsScrollArea" for the horizontal scroll bar and "vsUpButton", "vsDownButton", "vsScrollButton", "vsScrollArea" for the vertical scroll bar (bare in mind that the linkage ids are case sensitive). The cross cursor used when the navigation is "mouse drag" can also be replaced by a movieclip from the library if it has the linkage id of "MousePointer". During the loading of images, when the component has a displayEffect set, by default, there is a preloader clip displayed (if builtInPreloader is "bar" or "circle"), but it can also be replaced by a movie clip from the library if it has the linkage id of "Preloader" and the value of builtInPreloader is "none". Another feature of the component is the Visited Thumb state (available when the value of remainActiveOnPress is true), which displays a red triangle in the upper left corner of the thumb, if a thumb had already been "visited". Also this triangle can be changed with a movie clip from the library if it has the linkage id of "VisitedThumb". The last customizable element is the thumb's border; you can create a movie clip with the same size as the thumb and assign a linkage id of "ThumbnailBorder".

Tutorial 1 – Simple photo gallery

Here's a short tutorial on how you can use the ThumbnailList component, along with another loader component, to create a simple photo gallery without having to write any ActionScript code.

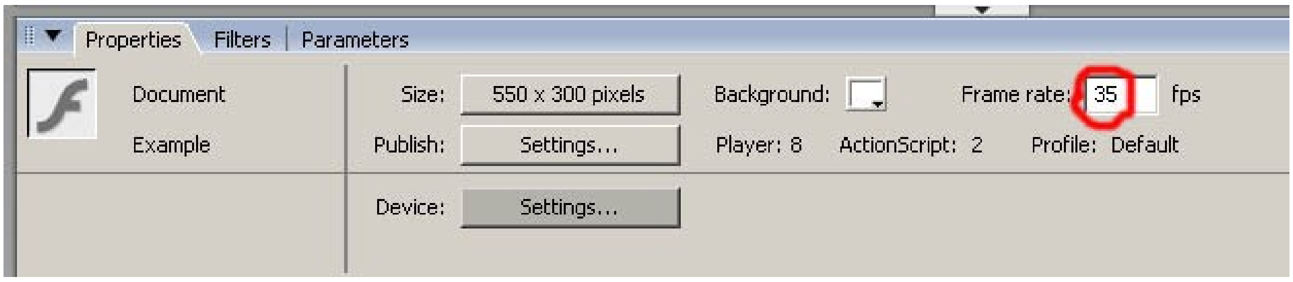
First, you'll have to install the component by double clicking on the mxp file you received. In the Macromedia Extension Manager you will be asked to accept or decline the component's disclaimer. If you decline, the component will not be installed and you won't be able to use it.

Now you need to create the xml file containing the list of images. Here is a XML file example that can be used with this component. The role of each attribute of the xml nodes has been explained in the previous paragraph. The file should be saved in the same directory with your swf file.

```
<?xml  version='1.0'  encoding='UTF-8'?>
<gallery>
    <img thumbnail='pictures/thumbnails/01.jpg' large='pictures/large/01.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/02.jpg' large='pictures/large/02.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/03.jpg' large='pictures/large/03.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/04.jpg' large='pictures/large/04.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/05.jpg' large='pictures/large/05.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/06.jpg' large='pictures/large/06.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/07.jpg' large='pictures/large/07.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/08.jpg' large='pictures/large/08.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/09.jpg' large='pictures/large/09.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/10.jpg' large='pictures/large/10.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/11.jpg' large='pictures/large/11.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/12.jpg' large='pictures/large/12.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/13.jpg' large='pictures/large/13.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/14.jpg' large='pictures/large/14.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/15.jpg' large='pictures/large/15.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/16.jpg' large='pictures/large/16.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/17.jpg' large='pictures/large/17.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/18.jpg' large='pictures/large/18.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/19.jpg' large='pictures/large/19.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/20.jpg' large='pictures/large/20.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/21.jpg' large='pictures/large/21.jpg'
description='image description'/>
```

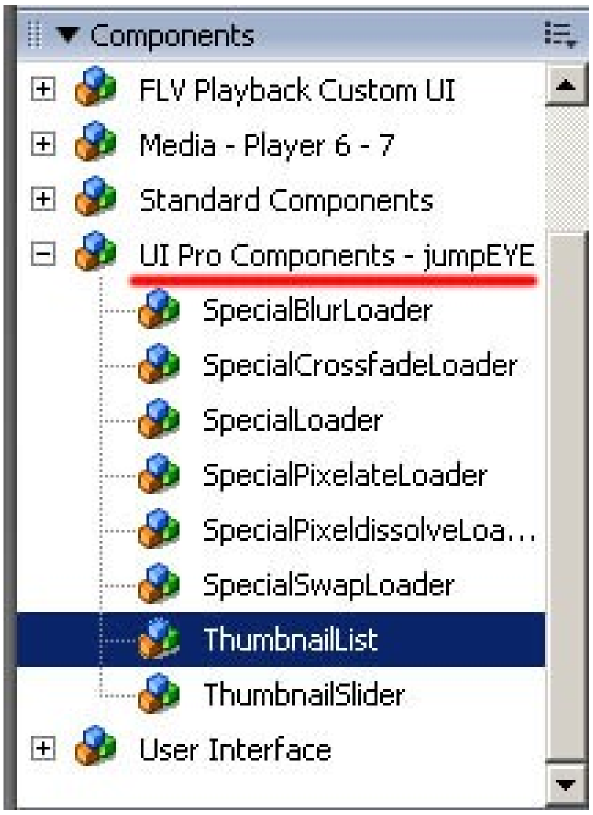
```
    <img thumbnail='pictures/thumbnails/22.jpg' large='pictures/large/22.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/23.jpg' large='pictures/large/23.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/27.jpg' large='pictures/large/27.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/28.jpg' large='pictures/large/28.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/29.jpg' large='pictures/large/29.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/30.jpg' large='pictures/large/30.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/31.jpg' large='pictures/large/31.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/32.jpg' large='pictures/large/32.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/33.jpg' large='pictures/large/33.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/34.jpg' large='pictures/large/34.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/35.jpg' large='pictures/large/35.jpg'
description='image description'/>
    <img thumbnail='pictures/thumbnails/36.jpg' large='pictures/large/36.jpg'
description='image description'/>
</gallery>
```

After the component has been installed, start the Macromedia Flash IDE. Create a new Flash movie by clicking on the File – New menu and then save it with the name Example.fla. First, you need to setup the stage. Set a size of 550 x 300 pixels to the stage and set the frame rate to 35 fps.

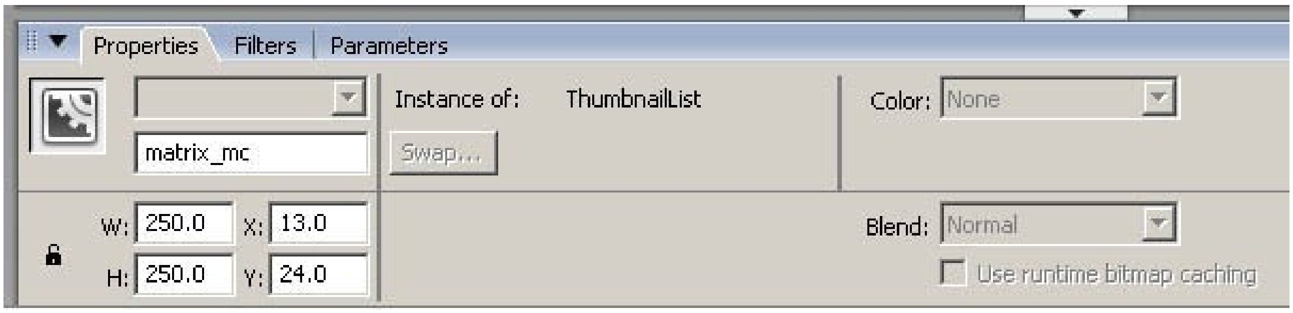


Size: 550 x 300 pixels
Background:
Frame rate: 35 fps
Player: 8
ActionScript: 2
Profile: Default

Next, you need to drag an instance of the component onto the stage. You will find this component in the Components Panel, UI Pro Components – jumpEYE folder.



After you position the component on the stage, you need to give it an instance name and the xml file from which it will extract the path and other details to the images or symbols you wish to add to the component. So, set the coordinates to X:13 and Y: 24 and the size to 250 x 250.



matrix_mc

Instance of: ThumbnailList

Color: None

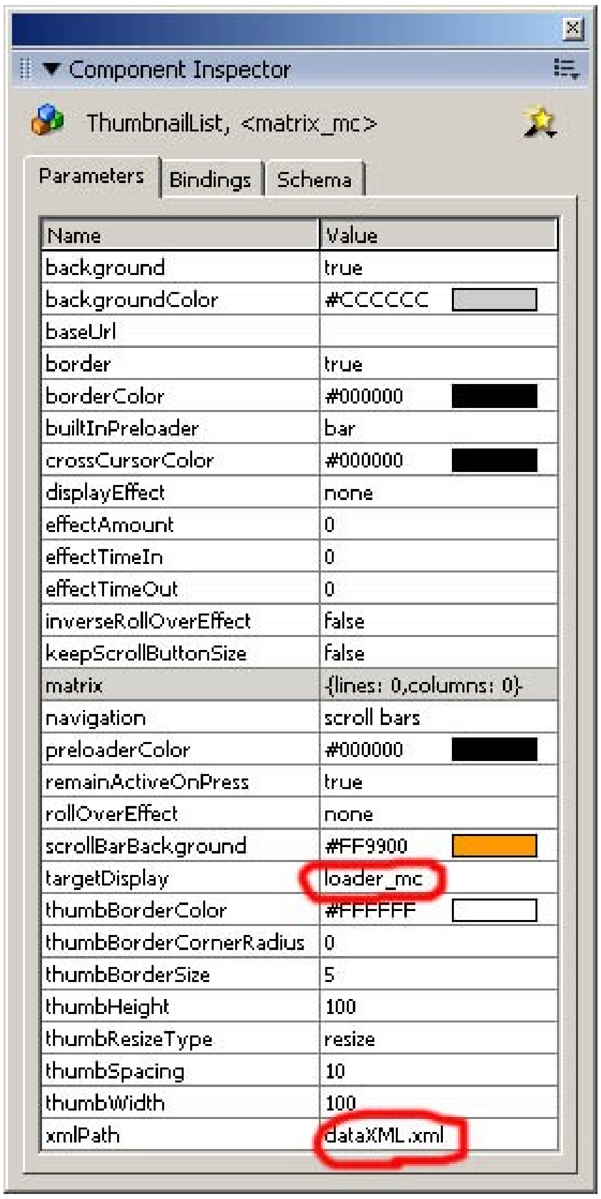
W: 250.0 X: 13.0

H: 250.0 Y: 24.0

Blend: Normal

Use runtime bitmap caching

Next, you need to set the **xmlPath** property, by giving the name of the xml file containing the images. If the xml file is not located in the same directory with your swf file, you need to write the complete path to the xml file. This component can be used along with any another loader component that has the contentPath property, to display the images in the ThumbnailList. In this case we will be using the SpecialLoader component from JumpEYE Creative Media (http://store.jumpeye.com/special_loader/index.html) and we'll give it an instance name of loader_mc. Now you have to set the **targetDisplay** property to the second loader by writing the loader's instance name. For any help, refer to the next image.

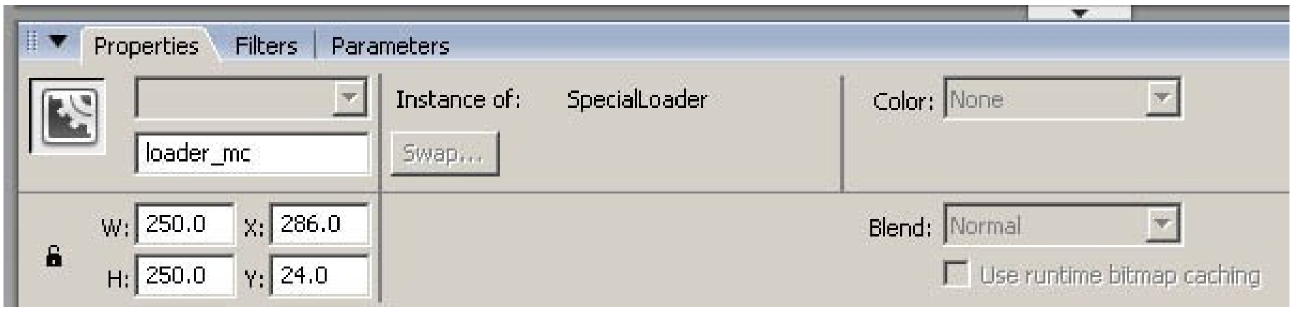

Component Inspector

ThumbnailList, <matrix_mc>

Parameters | Bindings | Schema

Name	Value
background	true
backgroundColor	#CCCCCC ⬜
baseUrl	
border	true
borderColor	#000000 ⬛
builtInPreloader	bar
crossCursorColor	#000000 ⬛
displayEffect	none
effectAmount	0
effectTimeIn	0
effectTimeOut	0
inverseRollOverEffect	false
keepScrollButtonSize	false
matrix	{lines: 0,columns: 0}
navigation	scroll bars
preloaderColor	#000000 ⬛
remainActiveOnPress	true
rollOverEffect	none
scrollBarBackground	#FF9900 🟧
targetDisplay	loader_mc
thumbBorderColor	#FFFFFF ⬜
thumbBorderCornerRadius	0
thumbBorderSize	5
thumbHeight	100
thumbResizeType	resize
thumbSpacing	10
thumbWidth	100
xmlPath	dataXML.xml

Next, you need to drag onto the stage the second loader component and set it up. Give it the instance name of loader_mc, set the coordinates to X:286 and Y:24 and set the size to 250 x 250.


Properties | Filters | Parameters

loader_mc

Instance of: SpecialLoader

Swap...

Color: None

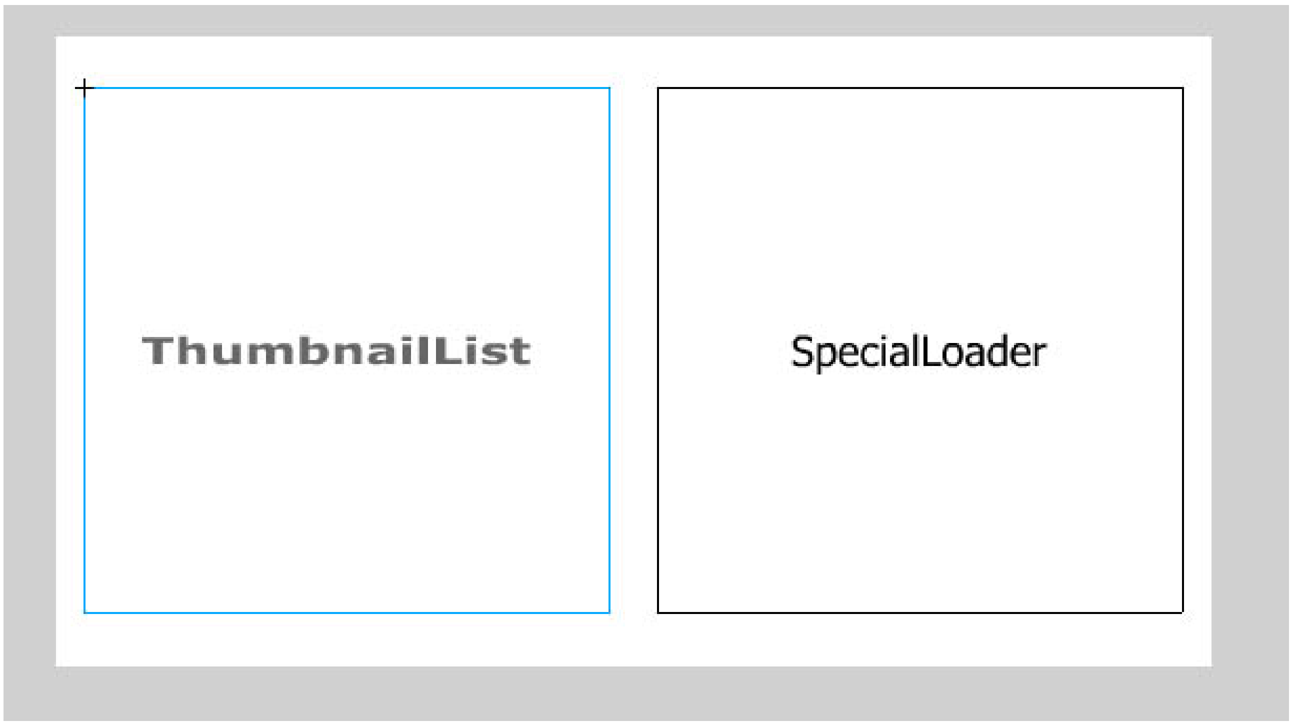
W: 250.0 X: 286.0

H: 250.0 Y: 24.0

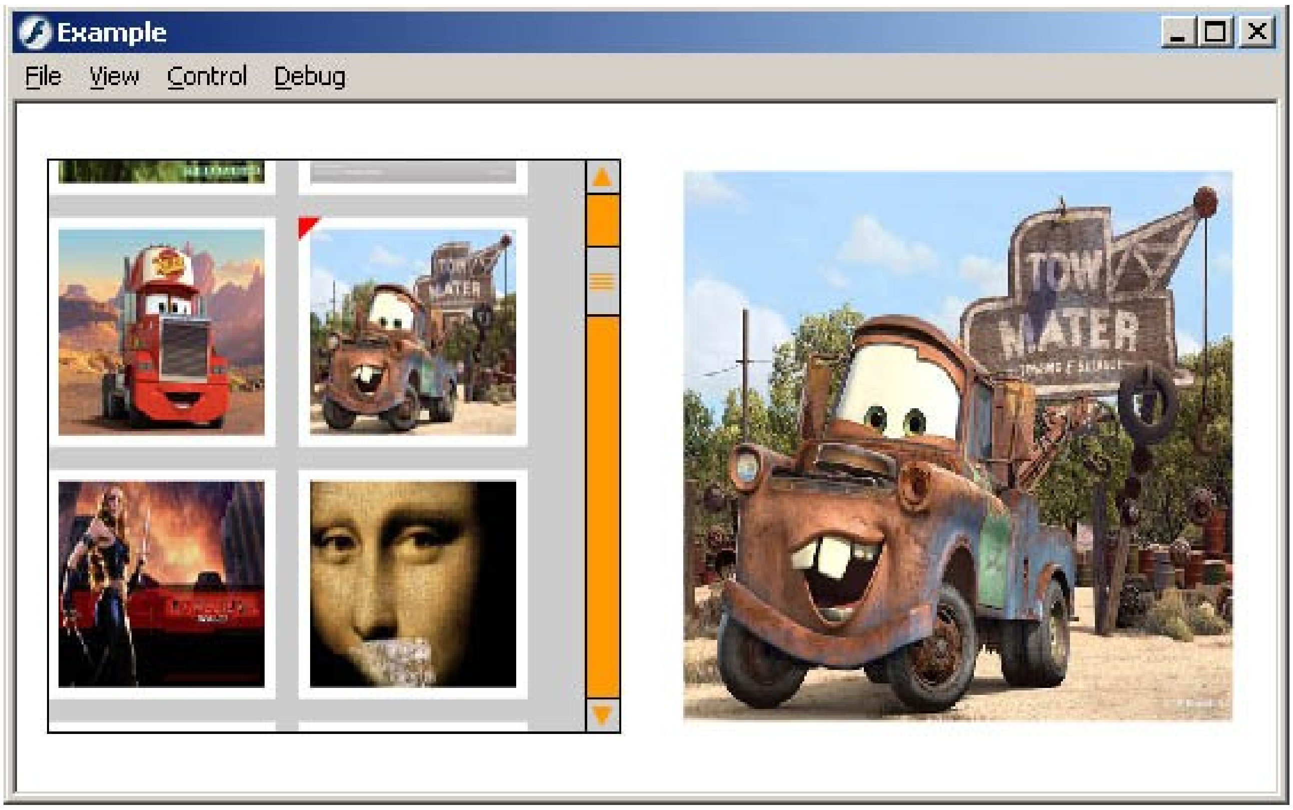
Blend: Normal

Use runtime bitmap caching

At this moment you should have the Stage look similar to this:

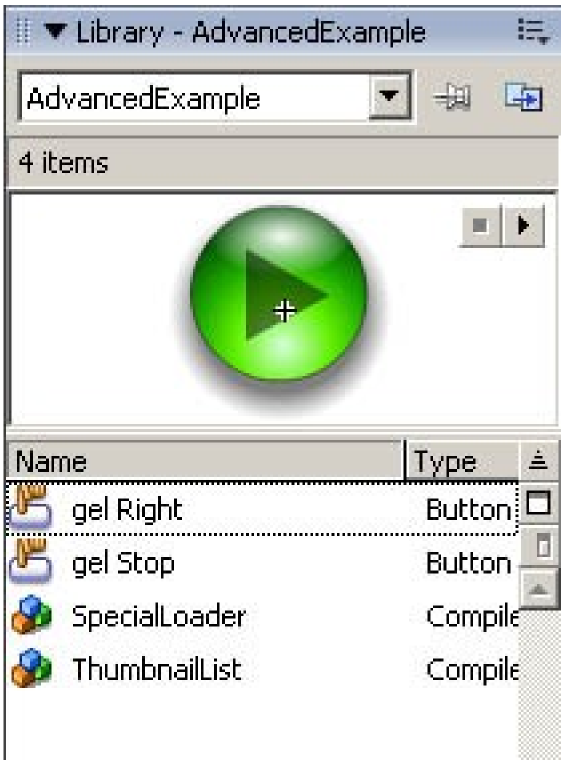


You can test the movie by pressing Ctrl+Enter. The final result should look like this:



Tutorial 2 – Advanced photo gallery with slide show
(requires ActionScript knowledge)

For those who need to customize the component and to use it along with ActionScript code, here is an example of how to set up the component. After you've created the xml file (refer to previous paragraph), in your new fla file you need set the Stage size to 550 x 300 pixels and the frame rate to 35 fps and then drag the component into the Library. Besides the ThumbnailList component, I'll be using JumpEYE's SpecialLoader component (http://store.jumpeye.com/special_loader/index.html) to demonstrate the slide show capabilities and two buttons, which I dragged into the Library and set the linkage id. You can find the buttons in Buttons Library which is located under the Window->Common Libraries menu. The buttons I used are gel buttons, that you can find in the Classic buttons folder, in the Buttons Library, but you can use any other buttons or movie clips. By now, you should have four elements in the Library:


Library - AdvancedExample
AdvancedExample
4 items

Name	Type
gel Right	Button
gel Stop	Button
SpecialLoader	Compile
ThumbnailList	Compile

Now, you'll start writing code in the Actions panel (F9 to bring up the panel). To set up the component you have to attach it from the Library, set it's properties and then, load the xml file. Note that the loading of the xml file should be the last function you call, all the properties should be set before the **load** function. You can set the component's properties either using the **initObject** parameter of the **attachMovie** function or by setting each parameter separately, after you have attached the component to the stage.

```
// matr object that will be assigned to the matrix property
// of the ThumbnailList component
var matr:Object = new Object({lines:4, columns:6});

this.attachMovie("ThumbnailList",   "matrix_mc",   this.getNextHighestDepth(),
{_x:0, _y:0, navigation:"mouse movement"});

// set some of the properties of the component
matrix_mc.keepScrollButtonSize = true;
matrix_mc.matrix = matr;
matrix_mc.setSize(300, 300);
matrix_mc.displayEffect = "fade in";
matrix_mc.rollOverEffect = "black&white";
matrix_mc.backgroundColor = 0x000000;
matrix_mc.preloader = "circle";
matrix_mc.preloaderColor = 0xFFFF00;
matrix_mc.thumbBorderSize = 5;
matrix_mc.thumbBorderCornerRadius = 5;
matrix_mc.thumbBorderColor = 0xFFFFFF;
matrix_mc.thumbWidth = 100;
matrix_mc.thumbHeight = 100;
matrix_mc.thumbSpacing = 5;
matrix_mc.remainActiveOnPress = true;

// load the xml file and display the thumbs
matrix_mc.load("dataXML.xml");
```

Next, you have to bring the rest of the components on the stage (the SpecialLoader and the two buttons) and set their position. After that, you must write the action for the onPress event of the buttons. When you press the first button (startSlide_btn), it will start the slide show of the thumbs. When you press the second button (stopSlide_btn), the slide show will stop (the component also has the pause and resume capabilities for the slide show).

```
this.attachMovie("SpecialLoader", "loader_mc", this.getNextHighestDepth(),
{_x:325, _y:25, _width:200, _height:200});

this.attachMovie("gel Right", "startSlide_btn", this.getNextHighestDepth());
startSlide_btn._x = 400;
startSlide_btn._y = 270;

this.attachMovie("gel Stop", "stopSlide_btn", this.getNextHighestDepth());
stopSlide_btn._x = 455;
stopSlide_btn._y = 270;

startSlide_btn.onPress = function(Void):Void {
      // the images will be displayed at a 2 second interval
      matrix_mc.startSlideshow(2);
}

stopSlide_btn.onPress = function(Void):Void {
      matrix_mc.stopSlideshow();
}
```

Finally, we will create a listener and listen for two of the events that the ThumbnailList component dispatches: **onPressThumb** and **onSlideshowChange**. The onPressThumb event is triggered when the user clicks one of the thumbs. The event also sends the user the details about that thumb: **xmlNode** – the attributes corresponding to that thumbnail in the xml file (the attributes are: thumbnail – the path to the image or swf file or the linkage id of symbol from the library; large – the large image corresponding to the thumbnail, used to be displayed in photo galleries for example; description – a short description of the thumbnail's content), **thumb** – the name of the thumbnail and **clipIndex** – the index of the thumbnail. The onSlideshowChange event is triggered at an interval given by the user and cycles through the thumbnails, in the order they were written in the xml file. When the slide show reaches the last thumbnail, it will automatically restart with the first thumb. The idea is to display the large image of each thumbnail, using the SpecialLoader component, when the user clicks on the thumbs or during the slide show. This can be done much easier, by setting the targetDisplay property of the ThumbnailList component to the name of the loader component (in this case, “loader_mc”) but this method was presented in the previous paragraph and this tutorial's goal is to show you how you can use the events that the component dispatches.

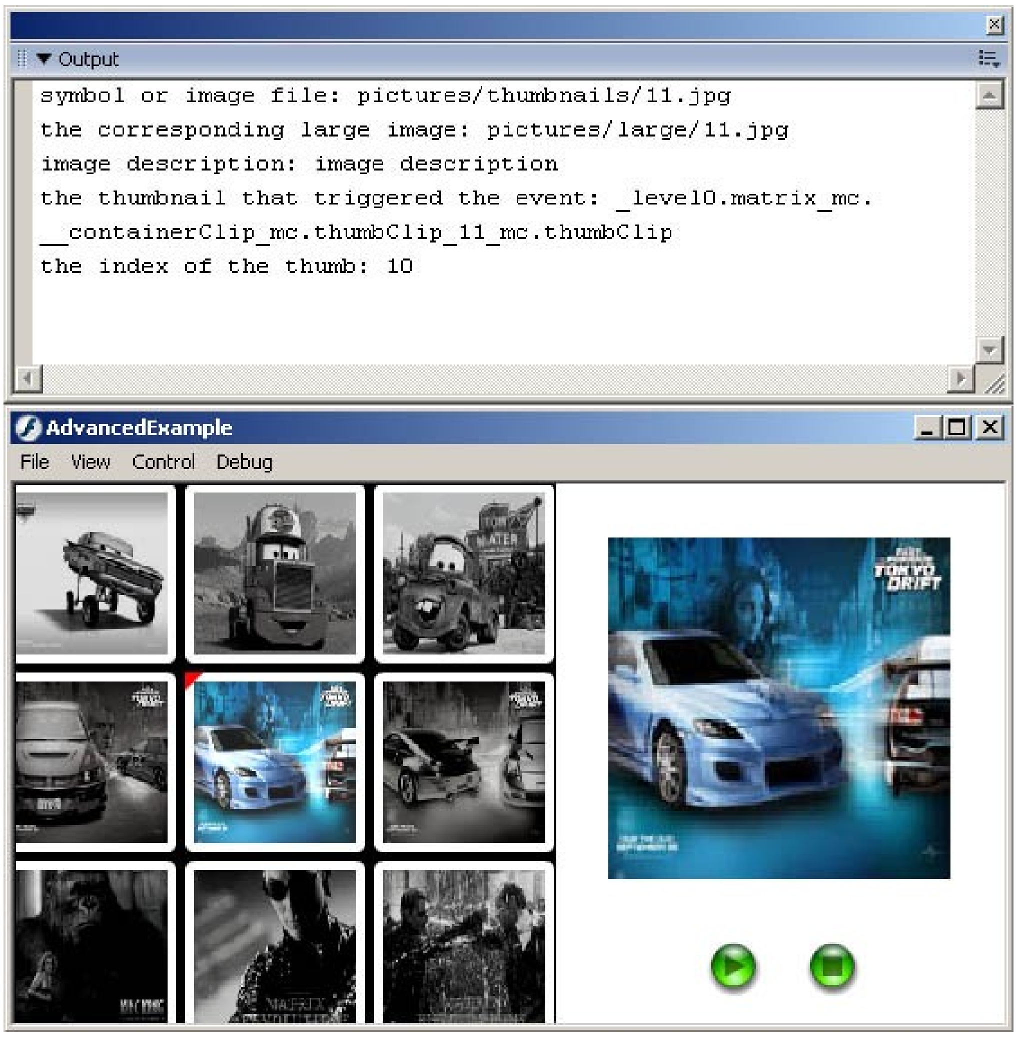
```
var listener:Object = new Object();

listener.onPressThumb = function(xmlNode:Object, thumb:MovieClip,
clipIndex:Number):Void {
	loader_mc.contentPath = xmlNode.thumbnail;
	trace("symbol or image file: "+xmlNode.thumbnail);
	trace("the corresponding large image: "+xmlNode.large);
	trace("image description: "+xmlNode.description);
	trace("the thumbnail that triggered the event: "+thumb);
	trace("the index of the thumb: "+clipIndex);
}

listener.onSlideshowChange = function(xmlNode:Object, thumb:MovieClip,
clipIndex:Number):Void {
	loader_mc.contentPath = xmlNode.thumbnail;
	trace("symbol or image file: "+xmlNode.thumbnail);
	trace("the corresponding large image: "+xmlNode.large);
	trace("image description: "+xmlNode.description);
	trace("the thumbnail that triggered the event: "+thumb);
	trace("the index of the thumb: "+clipIndex);
}

matrix_mc.addListener(listener);
```

In each function I traced the parameters returned by the events, to show you the value of each parameter. If you test the movie clip, the result should look like this:



```
symbol or image file: pictures/thumbnails/11.jpg
the corresponding large image: pictures/large/11.jpg
image description: image description
the thumbnail that triggered the event: _level0.matrix_mc.
__containerClip_mc.thumbClip_11_mc.thumbClip
the index of the thumb: 10
```