Video Player
Audio & video in Flash7

by Armand Niculescu
Media Division

Flash 7 comes by default with a large set of components, including those needed to play video. To play a video you just drag and drop it on the stage, set a few parameters and the video will play.

The player component is 83Kb, not much, but considering that most video files you'll play are in the 200Kb-800Kb range, it's not insignificant. Another thing to consider is its appearance. It's rather difficult to integrate it seamlessly in your website; furthermore, you may need custom functionality that the default player doesn't provide.

Therefore in this tutorial you'll learn how to make your very own video player, one that you can tailor for you specific needs. The player is only 2Kb and you'll be able to control every aspect of its behavior.

Have fun,

Armand Niculescu
armand@media-division.com
www.MediaDivision.com
www.RichNetApps.com

The Components

First we need to decide what controls we want to have over our video. In order to keep the interface as simple as possible, I decided to have a simple play/pause toogle, a mute/un-mute button and a drag-able slider to see and set the current time. Additionally, since the video is streaming, we'll need to know how much of the video has been loaded.

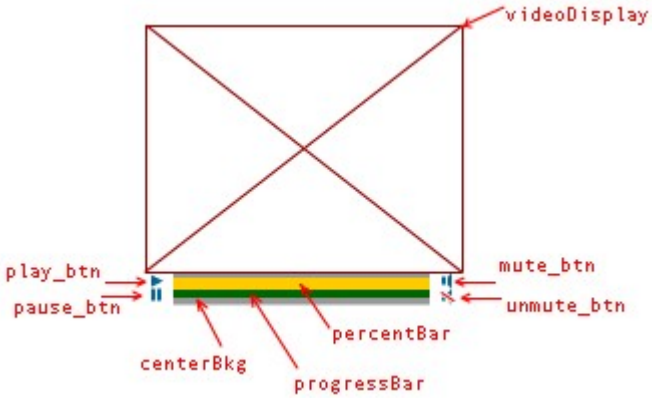
Create an empty movie clip. In it you need to add at least the following elements:

- a play button;
- a pause button;
- a mute button;
- an un-mute button;
- a background bar movie clip;
- a loaded percentage bar movie clip;

- a current position bar movie clip;
- a video element.

The buttons are simple native Flash buttons, not button components.
The bars are just movie clips, colored differently, about 100 pixels wide and 10 pixels high.
The video can be added using the New Video option from the Library command panel and then by dragging it on stage.

The elements on stage will then look like this:



Please name each instance according to the image above, so we can control them.

Later, you'll be able to create a player skin, but in this tutorial we'll be concentrating on functionality.

It's time to add some actions.

Initializing the video

Flash video was initially intended only for live streaming with Flash Communication Server and it was very poorly documented. Even now, making it work requires more work than, say, the sound.

```
1. var connection:NetConnection = new NetConnection();
2. connection.connect(null);
3. var video = new NetStream(connection);
4. videoDisplay.attachVideo(video);
```

The `NetConnection` class is used by Flash Communication server. The `connect(null)` method must be called exactly like that according to the Macromedia Actionscript Dictionary. Then, the video stream object is created using the null connection and then it's linked to the video object on stage.

In case you're wondering, I'm not declaring the data type for video in order to be able to use the undocumented `onMetaData` method. The flash compiler doesn't know `onMetaData` is part of the

`NetStream` class and would throw an error.
If you want to be able to manipulate the sound of the video, some more code has to be written:

```
5. this.createEmptyMovieClip("soundClip", 0);
6. soundClip.attachAudio(video);
7. audio = new Sound(soundClip);
```

So, apart from the video object, we need a sound object, that fortunately is just a simple movie clip, so we'll create it dynamically. The undocumented `attachAudio()` method does the magic of linking the sound in the video stream to the movie clip. Then, the audio (now part of the movie clip) can be controlled with the `Sound` class.

Is your head spinning yet?

Controlling the video

The good news is that the difficult part is over. All that's left to is to actually control the playback.

First, let's show by default only the controls that we want:

```
8. pause_btn._visible = unmute_btn._visible = false;
```

Then, we can switch the buttons:

```
9. function switchPlayButton(v:Boolean):Void
10.{
11.  if (v==null)
12.    v = !play_btn._visible;
13.  play_btn._visible = v;
14.  pause_btn._visible = !play_btn._visible;
15.}
16.
17.function switchMuteButton():Void
18.{
19.  unmute_btn._visible = !unmute_btn._visible;
20.  mute_btn._visible = !unmute_btn._visible;
21.}
```

The two methods above are there just to switch the visibility of the play/pause and mute buttons. The play/pause buttons can be forced set in a certain way.

Of course, by themselves, the above methods don't really do anything, so let's add the handlers for the four buttons:

```
22.mute_btn.onRelease = unmute_btn.onRelease = function()
23.{
24.  audio.setVolume(Math.abs(audio.getVolume()-100));
25.  switchMuteButton();
26.};
27.
28.pause_btn.onRelease = play_btn.onRelease = function()
29.{
```

```
30.  video.pause();
31.  switchPlayButton();
32.};
```

Pretty simple, right? The video is controlled with the video object, using the **pause()** method and the audio is controlled with the audio object. The sound volume is just toogled 0 and 100 – you can improve that area with two button arrows or a slider, if you want a finer control over sound.

Loading and starting the video playback is very simple, too:

```
33.function play(file:String):Void
34.{
35.  video.play(file);
36.  barsInterval = setInterval(updateBars, 50);
37.  switchPlayButton(false);
38.}
```

As you can see, besides starting the playback, we're starting a **setInterval()** that makes the method **updateBars** to be called very 50 milliseconds. Why do we need such a method you ask? It's because the we want to see the playback progress. Let's add the code:

```
39.function updateBars():Void
40.{
41.  progressBar._width = centerBkg._width * video.time / video.totalTime;
42.  percentBar._width = Math.round(centerBkg._width * video.bytesLoaded /
  video.bytesTotal);
43.}
```

As you know, there are two bars: **progressBar** shows how much of the video has played; **percentBar** shows how much of the video has loaded. Their dimensions are calculated relative to the width of **centerBkg** clip.

With the exception of **totalTime**, all properties are provided by the video object. Unfortunately, as I said earlier, flash video was designed initially for live streaming. Until Flash Video 1.1 there was no way to know the duration of the video.

In order to get the duration, you must use the afore-mentioned **onMetaData()**:

```
44.video.onMetaData = function(obj:Object):Void
45.{
46.  this.totalTime = obj.duration;
47.  setSize(obj.width, obj.height);
48.};
```

The method **onMetaData()** is called automatically when the the header information is loaded. At the time of writing some programs don't generate the header information correctly, so the width, height or duration may not be accurate. *C'est la vie*.

Speaking of events, another event you must use is this one:

```
49.video.onStatus = function(obj:Object):Void
50.{
```

```
51.  switch(obj.code)
52.  {
53.    case "NetStream.Play.Start":
54.      setSize(videoDisplay.width, videoDisplay.height);
55.      break;
56.    case "NetStream.Buffer.Full":
57.      setSize(videoDisplay.width, videoDisplay.height);
58.      break;
59.    case "NetStream.Play.Stop":
60.      video.seek(0);
61.      video.pause(true);
62.      switchPlayButton();
63.      break;
64.  }
65.};
```

`onStatus()` receives an object with a status code. We're using two of the codes to adjust the size (because the information provided by the object in `onMetaData()` is sometimes inaccurate) and to move the playback "header" back to the start of the video when it has reached the end. You may add to these events `"NetStream.Play.StreamNotFound"` if you want to display an error message when the file specified doesn't exist.

With this, you must have noticed calls to the `setSize()` method.
This method is largely dependent on your video graphic elements. It must resize the `videoDisplay` movie clip and adjust the position of the buttons, as well as any 'skin" related movie clips.

In our case, the code is not too complicated:

```
66.function setSize(w:Number, h:Number):Void
67.{
68.  if (w==null || h==null || w<1 || h<1) return;
69.
70.  videoDisplay._width = w;
71.  videoDisplay._height = h;
72.
73.  var controllerX:Number = videoDisplay._x;
74.  var controllerY:Number = videoDisplay._y + videoDisplay._height;
75.  var controllerWidth:Number = videoDisplay._width;
76.
77.  centerBkg._x = percentBar._x = progressBar._x = 10 + controllerX;
78.  centerBkg._y = controllerY;
79.  centerBkg._width = controllerWidth - 20;
80.
81.  percentBar._y = progressBar._y = centerBkg._y + (centerBkg._height -
  percentBar._height)/2;
82.
83.  play_btn._x = pause_btn._x = centerBkg._x - play_btn._width/2;
84.  mute_btn._x = unmute_btn._x = centerBkg._x + centerBkg._width +
  mute_btn._width/2;
85.  play_btn._y = pause_btn._y = mute_btn._y = unmute_btn._y = centerBkg._y
  + centerBkg._height/2;
86.}
```

There's only one thing left to do: the ability to click and drag the progress slider to change the time.

This can be accomplished easily:

```
87.centerBkg.onPress = function()
88.{
89.  dragInterval = setInterval(this._parent.drag, 10);
90.};
91.
92.centerBkg.onRelease = centerBkg.onReleaseOutside = function()
93.{
94.  clearInterval(dragInterval);
95.};
96.
97.function drag():Void
98.{
99.  video.seek((centerBkg._xmouse * centerBkg._xscale) / (centerBkg._width *
   100) * video.totalTime);
100.}
```

On pressing the background bar, a **setInterval()** is created that calls **drag()** every 10 millisconds. Upon releasing the mouse button, this interval is cleared and the method isn't called any longer.

The **drag()** method calculates the time corresponding to the mouse position, considering that the width of the bar equals the total time.

To understand this, here's the basic formula:

$$\frac{mousePosition}{totalLength} = \frac{currentTime}{totalTime}$$

To this we must take into account that the bar is scaled (because we're changing its width, so the formula becomes:

$$\left(\frac{mousePosition}{totalLength}\right) \cdot \left(\frac{scaleFactor}{100}\right) = \frac{currentTime}{totalTime}$$

Therefore

$$\left(\frac{mousePosition}{totalLength}\right) \cdot \left(\frac{scaleFactor}{100}\right) \cdot totalTime = currentTime$$

Wrapping it up

For your convenience, here's the complete code:

```
1. var dragInterval:Number;
2. var barsInterval:Number;
3.
4. // create the audio-video streams
5. var connection:NetConnection = new NetConnection();
6. connection.connect(null);
7. var video = new NetStream(connection); /* we don't use NetStream dataType
   because flash doesn't recognize onMetaData */
```

```
8. videoDisplay.attachVideo(video);
9.
10.this.createEmptyMovieClip("sound", 0);
11.sound.attachAudio(video);
12.audio = new Sound(sound);
13.
14.pause_btn._visible = unmute_btn._visible = false;
15.
16.function play(file:String, autoStart:Boolean):Void
17.{
18.  video.play(file);
19.  barsInterval = setInterval(updateBars, 50);
20.  switchPlayButton(false);
21.}
22.
23.function setSize(w:Number, h:Number):Void
24.{
25.  if (w==null || h==null || w<1 || h<1) return;
26.
27.  videoDisplay._width = w;
28.  videoDisplay._height = h;
29.
30.  var controllerX:Number = videoDisplay._x;
31.  var controllerY:Number = videoDisplay._y + videoDisplay._height;
32.  var controllerWidth:Number = videoDisplay._width;
33.
34.  centerBkg._x = percentBar._x = progressBar._x = 10 + controllerX;
35.  centerBkg._y = controllerY;
36.  centerBkg._width = controllerWidth - 20;
37.
38.  percentBar._y = progressBar._y = centerBkg._y + (centerBkg._height -
  percentBar._height)/2;
39.
40.  play_btn._x = pause_btn._x = centerBkg._x - play_btn._width/2;
41.  mute_btn._x = unmute_btn._x = centerBkg._x + centerBkg._width +
  mute_btn._width/2;
42.  play_btn._y = pause_btn._y = mute_btn._y = unmute_btn._y = centerBkg._y
  + centerBkg._height/2;
43.}
44.
45.video.onMetaData = function(obj:Object):Void
46.{
47.  this.totalTime = obj.duration;
48.  setSize(obj.width, obj.height);
49.};
50.
51.video.onStatus = function(obj:Object):Void
52.{
53.  switch(obj.code)
54.  {
55.    case "NetStream.Play.Start":
56.      setSize(videoDisplay.width, videoDisplay.height);
57.      break;
58.    case "NetStream.Buffer.Full":
59.         setSize(videoDisplay.width, videoDisplay.height);
60.      break;
61.    case "NetStream.Play.Stop":
62.      video.seek(0);
63.      video.pause(true);
```

```
64.       switchPlayButton();
65.       break;
66.  }
67.};
68.
69.mute_btn.onRelease = unmute_btn.onRelease = function()
70.{
71.  audio.setVolume(Math.abs(audio.getVolume()-100));
72.  switchMuteButton();
73.};
74.
75.pause_btn.onRelease = play_btn.onRelease = function()
76.{
77.  video.pause();
78.  switchPlayButton();
79.};
80.
81.centerBkg.onPress = function()
82.{
83.  dragInterval = setInterval(this._parent.drag, 10);
84.};
85.
86.centerBkg.onRelease = centerBkg.onReleaseOutside = function()
87.{
88.  clearInterval(dragInterval);
89.};
90.
91.
92.function drag():Void
93.{
94.  video.seek((centerBkg._xmouse * centerBkg._xscale) / (centerBkg._width *
   100) * video.totalTime);
95.}
96.
97.function updateBars():Void
98.{
99.  progressBar._width = centerBkg._width * video.time / video.totalTime;
100. percentBar._width = Math.round(centerBkg._width * video.bytesLoaded /
   video.bytesTotal);
101.}
102.
103.function switchPlayButton(v:Boolean):Void
104.{
105. if (v==null)
106.    v = !play_btn._visible;
107. play_btn._visible = v;
108. pause_btn._visible = !play_btn._visible;
109.}
110.
111.function switchMuteButton():Void
112.{
113. unmute_btn._visible = !unmute_btn._visible;
114. mute_btn._visible = !unmute_btn._visible;
115.}
```

Note: The Flash source file is available on our site for download.

Seeing it in action

Here's a screenshot of the player in action:

With a little more effort, you can add a simple skin, to make it look like this:

Further improvement

As always, lots of things can be added to the player: better control over sound volume, a timecode, file-not-found error handling, time triggers (the ability to call a method when the playhead has reached a certain time to synchronize with other flash events) and so on.