vrije Universiteit amsterdam

Final Assignment.

Adobe AIR (Apollo) / Drag’n’Drop / File interaction

Date: 25 June 2008

Stefan van de Kaa - 1716158
Master: MMA
ska310@few.vu.nl

Final Assignment

For this assignment I wanted to make use of adobe Flex. I heard about the newer version of Flex namely Flex 3.0. Because I listen a lot to internet radio I thought it would be nice to make a player to listen to internet radio. This player should then be made in Flex and have a few requirements. It should be possible to listen to a couple of predefined radio stations. To start listening to a stream you should be able to drag and drop it into the player. Then the player will start playing automatically. As a feature I thought it would be possible to search for videos in YouTube. And after searching the applications lists the top 5 videos of the result. These can then also be dragged and dropped into the player.

After starting to build this application in Adobe Flex 3 I found out that is very difficult to display YouTube videos in a Flex application. I have looked for solutions, but these seems to be outdated with the new YouTube API. For this reason I changed my assignment to Adobe AIR. This because this is a new challenge for me and I'm very interested about this new way of making desktop applications.

My new Assignment

Since I have not worked with Adobe AIR I first did some sample applications and "Hello World" tutorials. After this I looked for some applications which use Drag'n'Drop technology. I found out that you have to use event handlers to get this to work.

```
//register for the drag enter event
addEventListener(NativeDragEvent.NATIVE_DRAG_ENTER, onDragIn);

//register for the drag drop event
addEventListener(NativeDragEvent.NATIVE_DRAG_DROP, onDragDrop);
```

This two lines register the event when someone hovers a document over the application and when someone drops the document on the application. We now need the two functions, onDragIn and onDragDrop, to get it to work.

```
private function onDragIn(e:NativeDragEvent):void
{
      //check and see if files are being drug in
      if(e.clipboard.hasFormat(ClipboardFormats.FILE_LIST_FORMAT))
      {
            //get the array of files
            var files:Array =
                  e.clipboard.getData(ClipboardFormats.FILE_LIST_FORMAT)
                        as Array;

            DragManager.acceptDragDrop(this);
      }
}
```

It is possible to check if there are multiple files dragged and dropped or that it is only one file. This is possible with the property *files.lenght*. Also it is possible to check here if the files have the extension you want to make it able to drop the files. For example check if the files are MP3's or are text files.

```
//called when the user drops an item over the component
private function onDragDrop(e:NativeDragEvent):void
{
      //get the array of files being drug into the app
      var arr:Array =
            e.clipboard.getData(ClipboardFormats.FILE_LIST_FORMAT)
                  as Array;

      for(var i:int =0; i<arr.length; i++){
            var file:File = File(arr[i]);

            if (file.extension != "mp3"){
                  Alert.show("This file is not supported!");
            } else {
                  // Do the right thing with the files.
            }
      }
}
```

This would make it possible to loop trough the files on the clipboard and do the right thing with them. For example add them to a itemlist. In my application these files are added to a cue which are played after addition. Therefore also the check if the files are MP3's.

I wanted this application to be transparent so that it does not have a title bar. I looked for the solution for this. In the projects XML file you should set the following properties:

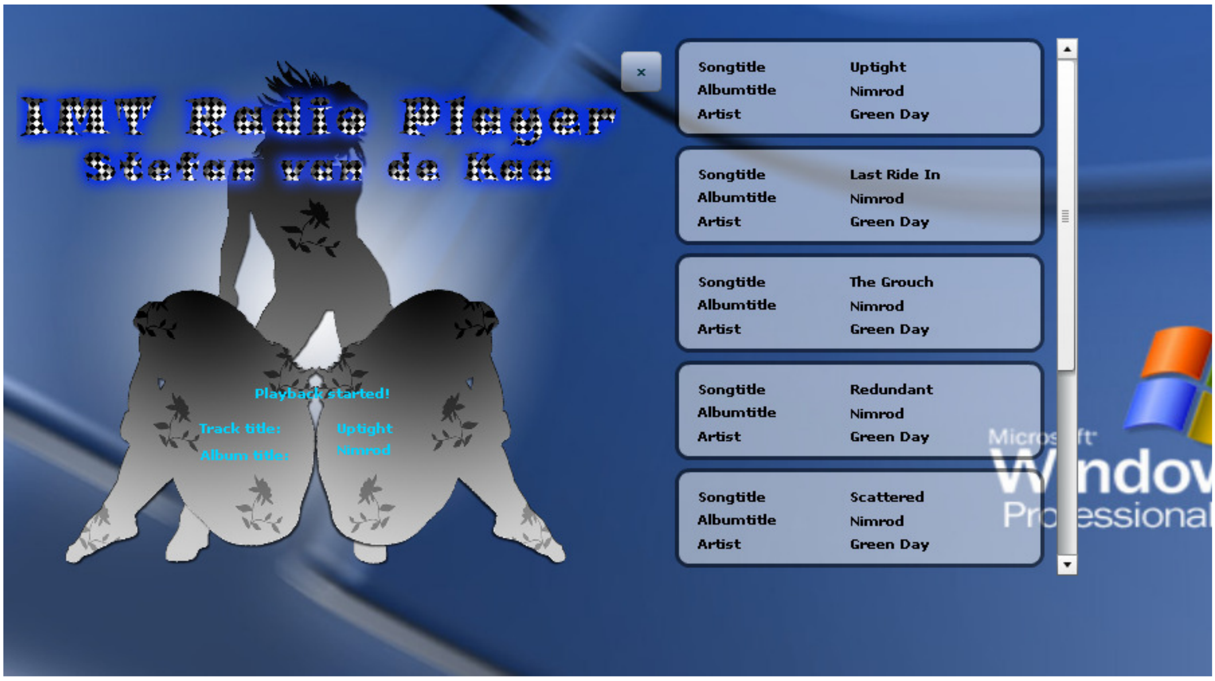
```
      <!-- The type of system chrome to use (either "standard" or "none").
                                          Optional. Default standard. -->
      <systemChrome>none</systemChrome>

      <!-- Whether the window is transparent. Only applicable when
                      systemChrome is false. Optional. Default false. -->
      <transparent>true</transparent>
```

But after this it didn't work for me. After searching for a long time I found out that in the application a property needs to be set.

```
<mx:WindowedApplication
      creationComplete="onCreationComplete()"
      xmlns:mx="http://www.adobe.com/2006/mxml"
      layout="absolute" xmlns:local="*"
      showFlexChrome="false"
>
```

Without the *showFlexChrome="false"* it doesn’t work. This all results in the next:



Playback started!

Track title: Uptight
Album title: Nimrod

Songtitle	Uptight
Albumtitle	Nimrod
Artist	Green Day
Songtitle	Last Ride In
Albumtitle	Nimrod
Artist	Green Day
Songtitle	The Grouch
Albumtitle	Nimrod
Artist	Green Day
Songtitle	Redundant
Albumtitle	Nimrod
Artist	Green Day
Songtitle	Scattered
Albumtitle	Nimrod
Artist	Green Day

Conclusion

I think Adobe AIR is a good way to make Desktop applications. Especially for people who are familiar with Adobe Flex. It works nice and is not to complex to get used to. Attached are the source files of my exercise and the complete working application. To install it you do have to install the latest Adobe AIR Runtime which is downloadable from the adobe website.