Flex Video Mashup

Intelligent Multimedia Technology

Rob Schuddeboom
Thijs Louisse

Vrije Universiteit Amsterdam

BETA

Concept

As a final assignment we want to make an application consisting of a mashup of several video services. We want to combine these services in a central interface using Flex and Actionscript 3.
The application will let the user search for videos and watch them in the Flex interface. The power of the application will be that it approaches different services and therefore delivers more search results than one single service.

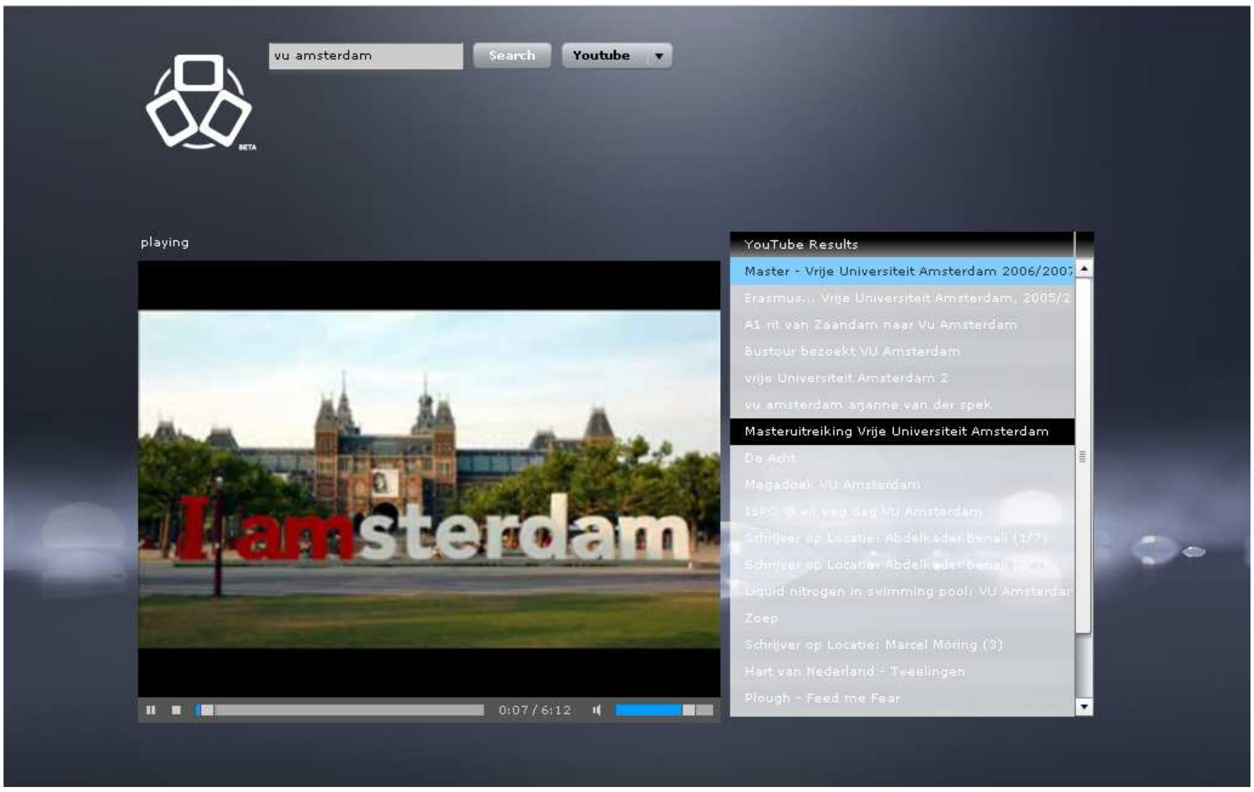
APIs
The most popular video service on the internet is YouTube. Because of its popularity and frequent use we assume this service has the highest amount of videos in their database. Therefore we assume that the most results our application will return, are coming from YouTube. For this reason, we start building our application around this API and make sure there's a perfect fit between our Flex application and the functionalities YouTube offers.
Once our basis has been established, we will add several other APIs. APIs that can be thought of are the Yahoo Video API, the Google Video API, the AOL Video API and others.

Flex
We want to create a dynamic , user friendly user interface with a professional and appealing look. The rich internet application properties Flex offers seem to be well suited for our application. Flex is a relatively new technique that offers great opportunities for user interaction. It also adds the possibility to convert the Flex application into a desktop application using Adobe Air in the future.

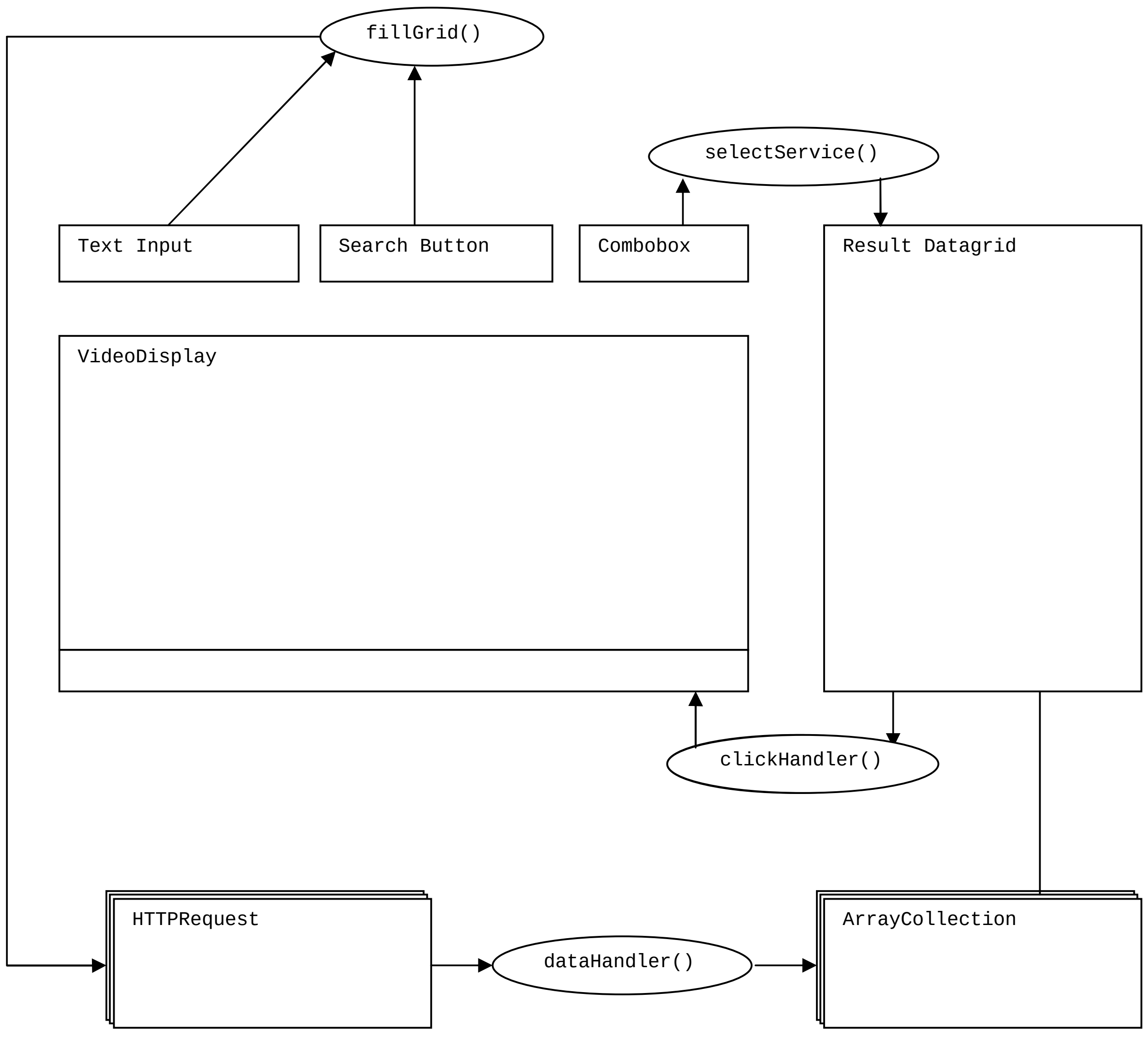
We will now give a brief description of the resulting application and any problems we ran into during the project and after that we will describe the code that was used to develop this application.

Results/resulting application

The resulting application is capable of retrieving and displaying the search results of three different RSS feeds: the Youtube API, the Yahoo Video API and the AOL Video API. Because of legal issues, we have chosen to display the results from each of these different sources separately. Users can switch between them by using a dropdown selection. The last search keyword is maintained and used for finding results from the newly selected source during this switching.
A limitation of our application is that it is only capable of playing flv files. This is a limitation of the Flex technology we used and we have accepted it as such. All videos from youtube can be played without any problems. This was one of our first goals for this project. From the results of Yahoo and AOL, only flv videos can be played. For this reason, we have included the video format in the list of search results, which makes it clear which videos can be played and which not. Another limitation is that flv videos belong to the category of flash videos, which also includes the swf format. Unfortunately, this format can't be displayed either.
For Youtube results there are no problems and for this reason we have chosen to display only the title of the video.



Master - Vrije Universiteit Amsterdam 2006/2007
Erasmus... Vrije Universiteit Amsterdam, 2005/2
A1 rit van Zaandam naar Vu Amsterdam
Bustour bezoekt VU Amsterdam
vrije Universiteit Amsterdam 2
vu amsterdam arjanne van der spek
Masteruitreiking Vrije Universiteit Amsterdam
De Acht
Megadoek VU Amsterdam
ISRO @ wij weg dag VU Amsterdam
Schrijver op Locatie: Abdelkader Benali (1/7)
Schrijver op Locatie: Abdelkader Benali [illegible]
Liquid nitrogen in swimming pool: VU Amsterdar
Zoep
Schrijver op Locatie: Marcel Möring (3)
Hart van Nederland - Tweelingen
Plough - Feed me Fear



This is how our application works: a search term is given at the Text Input field. When the search button is clicked, the function fillGrid() is called. This function updates the HTTPRequest for the video service selected. The HTTPRequest calls a method dataHandler() which stores the result of the HTTPRequest in an ArrayCollection. This collection serves as input for the datagrid.
When a result is clicked, the VideoDisplay is provided with a source and starts to play.

The combobox provides the user the opportunity to select another video service. The method selectService provides the right data for the datagrid.

Variables (actionscript)

```
[Bindable]private var query:String;
```

contains the query filled in in the input field

```
Bindable]private var ytQuery:String;
```

contains the query filled in in the input field, while the service ‘Youtube’ is selected. Two quotation marks (‘ + ytQuery + ’) are added to the string, which makes the query suitable for the link filled in in the youtube HTTPRequest.

```
[Bindable]private var serviceSelectedInt:int;
```

For convenience, each video service is identified with a service. In this way, one can easily deduct which service is selected. Respectively, the following services are used.
1= Youtube
2= AOL
3= Yahoo

```
[Bindable]private var serviceSelectedString:String;
```

niet gebruikt

```
[Bindable]private var youtubeArray:XMLListCollection;
```

the result of HTTPRequest ‘youtube’ is stored in this object.

```
[Bindable]private var aolArray:ArrayCollection;
```

the result of HTTPRequest ‘aol’ is stored in this object.

```
[Bindable]private var yahooArray:ArrayCollection;
```

the result of HTTPRequest ‘yahoo’ is stored in this object.

Variables (mxml)

```
flvc:FLVConstructor id="flvConstructor"
```
An external mxml component that adjusts the Youtube link in such a way that it's suitable for the VideoDisplay from Flex.

```
controls:FXVideo id="fxVideoDisplay"
```
An external mxml component which extends the VideoDisplay functionality of Flex. It has controls such as play, pause, adjust volume etc.

```
mx:HTTPService id="youtube/aol/yahoo"
```
each video service's data is obtained by an HTTPRequest object via a RSS(xml) feed.

Methods

```
private function selectService(event:Event):void
```

This method is called when an option from the combobox is selected. It sets the right dataprovider for the datagrid with search results(or selects the right (view)state)

```
private function dataHandler(event:ResultEvent,service:int):void
```

This method is called from the HTTPRequests. It fills the arrays the youtubeArray, aolArray or yahooArray, depending on which service is selected.

```
private function faultHandler(event:mx.rpc.events.FaultEvent):void
```

catches a fault message when an HTTPRequest doesn't receive the right data.

```
private function fillGrid():void
```

when the user enters a search query, this function is called after clicking 'search' or pressing enter. The query is stored in a String. Also, this function 'refreshes' the HTTPRequests. It updates content of the HTTPRequests with the updated query.

```
private function clickHandler(event:MouseEvent):void
```

when the user clicks on a search result in the datagrid, this function is called. It sets the right source for the fxVideoDisplay object.

Resources

Flex

Tutorials and Guides

- Total Training for Adobe Flex 2: Rich Internet Applications

- http://learn.adobe.com/wiki/display/Flex/
  Examples and tutorials from Adobe

- http://blog.flexexamples.com/
  Blog with Flex examples

- http://www.abdulqabiz.com/blog/archives/flash_and_actionscript/constructing_youtube_1.php
  Explanation about making Youtube videos suitable for integrating with Flex

Components

- http://flex.org/software/components
  Collection of Flex Components

- http://flexbox.mrinalwadhwa.com/
  Collection of Flex Components

- http://www.fxcomponents.com/docs/fxvideo/com/fxcomponents/controls/FXVideo.html
  FXVideo Component

- http://code.google.com/p/flex-tube/source/browse/trunk/com/fs/FSVideoDisplay.as?r=8
  FullScreenVideo Component

API's

- http://developer.yahoo.com/search/video/V1/videoSearch.html
  Yahoo Video API

- http://developer.truveo.com/RESTAPIOverview.php
  AOL Video API

- http://code.google.com/apis/youtube/overview.html
  Youtube Video API