(intelligent) multimedia technology – 2008

everything you always wanted to program but never ...

(I)MT 2010 – new course topics
The course will (again) be of a practical nature, encouraging the students to produce appealing applications, such as **rich-media mashups**, **data-driven visualisation(s)** and to experiment with **new (media) technologies**.

A. Eliëns, update november 2010, (eliens@cs.vu.nl)

theme(s) – web services, visualisation, (multimodal) interaction(s)
An ongoing theme for **multimedia @ VU** is (serious) games. Web services, and visualisation(s) of on going activities, may contribute to authoring such games, by allowing for live/dynamic content.

- climate game(s)
- language game(s)
- social network(s)

Also social network technologies are of interest, not only as a source of dynamic content, but also for aspects of (real time) communication, among others to promote a sense of being connected.

inspiration(s) – (serious) mashups
Both the **social web** and the **instance web**, as exemplified in **twitter** messages, may result in intersting patterns of behavior. One of the challenge(s) here is to visualize these patterns.

inspiration(s) – (serious) mashups

- we feel fine – www.wefeelfine.org
- universe – universe.daylife.com
- twitter – twittervision.com
- flickr – flickrvision.com
- diggs – labs.digg.com
- mailgarden – human-centered-visualizations.com
- retrievr – labs.systemone.at/retrievr

Visualizations may intent to be nothing but pleasing or artful, but may potentially also lead to a better understanding of human behavior, as advocated in a discpline known as **Cultural Analytics**[1].

assignment(s)
There is considerable freedom in the choice of your project(s), however it must comply with the theme of **rich media (data-driven) mashup(s)**, and (in principle) make use of **recommended technologies**, among which the **flex/as3** framework.

assignment(s)

- see practicum – rich media (data-driven) mashup(s)

In delivering the project result, you must strive for a professional approach, that is an end product, downloadable as zip, with all sources, working examples, documentation as well as tutorial or getting started applications, that illustrate your project.

resource(s) – recommended technology
You are strongly encouraged to develop **rich-media mashup(s)** in the **flex/as3** framework, with (possibly) use of **PHP**, for server-side extensions, needed to circumvent security restrictions, using **XML** and/or **AMF** (Actionscript Message Format) for exchange.

ximpel / server(s)[2] **/ material(s)**

- **wave(s)** – googlewavedev.blogspot.com/2009/11/wavesandboxcom-federate-this.html
- tutorial(s) – **web technology** / flex/as3

[1] www.hpcwire.com/features/The_Next_Big_Thing_in_Humanities_Arts_and_Social_Science_Computing_Cultural_Analytics.html
[2] www.cs.vu.nl/∼eliens/media/code-server.html

- sound & visual(s) – multimedia / technology / camera / display(s) / vision(s)
- **flex** – www.adobe.com/products/flex
- **mobile** – **iPhone**
- **xml** – tutorial / example(s)
- **umap** – www.afcomponents.com/tutorials/umap_as3 / flex tutorial
- web orb – www.themidnightcoders.com/weborb
- labs – labs.adobe.com/technologies/flex
- **api(s)** – labs.adobe.com/wiki/index.php/ActionScript_3:resources:apis:libraries
- code – code.google.com
- **db** – **sqlite** / mysql / exist/xml
- wamp – **appserver** / server2go / webdeveloper
- **mashup(s)** – **wso2**: wso2.org/projects/mashup
- amfphp – sourceforge.net/projects/amfphp
- **multiuser** – **smartfoxserver.com** / code.google.com/p/gfs-server / osflash.org/red5

Alternatively, however, you may focus on **server-side mashups**, with a suitable **rich-media interface**, preferably availabe as a flex component.

lookat(s) – example application(s)
To get familiar with the various technologies, kearning by example seems by far the most efficient method. After looking at the examples, get some practical experience, and then decide on the goals and requirements of your project.

lookat(s) – actionscript[3]

- palette – map
- flickr – photo(s)
- alarm – clock
- filter(s) – workbench
- visual – algorithmic(s)
- rss – viewer
- view – stack
- garden – node(s)
- tile – eye
- recent – flickr(s)
- product(s) – amazon
- photo – cube
- map(s) – multimedia @ VU

Obvoiously, one of the major challenges here is to circumvent security restrictions of the flash player and connect to servers outside of the domain of origin. As in the last examples, you need a proxy, that you may download from media/proxy.zip[4]. By the way, do not hesitate to look at flashplayer 10[5].

deliverable(s)
At the beginning of the project, you have to create a web site that gives access to all deliverables, and finally a package for download (in **zip** format). During the course of the project, the website must reflect goals, progess, and achievements, preferably by demonstration(s) of partial results.

deliverable(s)

1. concept – idea & plan of approach
2. application(s) – code + documentation
3. essay – about approach and use of technology

[3] www.cs.vu.nl/∼eliens/media/lookat-actionscript.html
[4] www.cs.vu.nl/∼eliens/media/proxy.zip
[5] labs.adobe.com/technologies/flashplayer10

4. documentation – tutorial explanation of API/SDK with elementary examples

No need to repeat that all deliverables must be made available from your website. Finally, for documentation use PDF or HTML. In particular, .doc files will **not** be accepted!

scope(s) – aspiration(s)

Although it is tempting to develop the **ultimate application**, that will make you rich and famous for the rest of your life, there are great **risk(s)**, that you will not succeed, and as a consequence end up with **nothing** useful. Develop a **prototype**, is my advice, that runs on **localhost** and may be used later, perhaps by yourself, to realize the system that ultimately fullfills your **dream(s)** ... Another advice, why not do the obvious, and use **map(s)**?