UOC

Open University of Catalonia

Universitat Oberta de Catalunya

Bachelor in Multimedia: Undergraduate Thesis

Grau en Multimèdia: Treball Fi de Grau (TFG)

FeedPeep: Application to analyze color patterns and meanings at image sets from RSS feeds

FeedPeep: Aplicación para analizar pautas y significados de color en sets de imágenes desde RSS feeds

Student / Alumne: Eva Casado de Amezua Fernández-Luanco [e_luanco@uoc.edu]
Date / Data: 2011.06.10

Research Advisor / Consultor: Daniel de Fuenmayor López [dde_fuenmayor@uoc.edu]
Professor / Profesor: Carlos Casado Martínez [ccasadom@uoc.edu]

ABSTRACT

The production and communication of images, from photographs to digital illustrations, is benefiting from the possibilities that new technologies and the Internet bring. As the rate of production and publication increases, also does the challenges associated to their visualization and management. The number of images a user can visualize study or organize in a given time is limited. Thus, the development of new tools to support producers, managers and end consumers is more important than ever.

There are many subjects and situations where analytical tools are key to success. For instance, in art studies to gain knowledge on an author or artistic movements, discovering color trends in marketing campaigns, gather data for academic essays and, as is the case of the author of this undergraduate thesis, for discovering production and color patterns in photographic collections[1].

All these considered, the present undergraduate thesis is committed to develop a little tool that will focus on the visual analysis of image sets to discover color and production patters and meanings. Such tool is specially thought for those interested in gaining knowledge on image color and production trends and styles, as well as an author's mark and timeline of production.

For this thesis and taking into account time limitations –see Gantt diagram in PAC1-, the tool is to be developed from zero to version 1.0 and left ready for offering further development possibilities, some of which will be proposed in this document. Being relevant the application to be as portable, cross-platform and online functional as possible, the chosen key technologies are ActionScript, a language program developed by Adobe Systems Inc., and the RSS format –a standardized data model based on XML language-. The tools chosen for development and satellite tasks include several applications and services, such as Adobe Flash or Adobe Dreamweaver.

This dissertation document includes also detailed information about the project conceptual evolution, development process, theoretical frame, defense, conclusions and further information that may be required by the Professor and the Research Advisor to fulfill evaluation and quality requirements.

[1] Other interesting examples of visual data analysis are published by the UCSD Calit2's Software Studies Research Group at http://lab.softwarestudies.com/.

To Love, incarnated in my mother, my man and my family of blood and soul.

"Time is the substance from which I am made. Time is a river which carries me along, but I am the river; it is a tiger that devours me, but I am the tiger; it is a fire that consumes me, but I am the fire."

Jorge Luis Borges

INDEX

CHAPTER 0
ABOUT THIS UNDERGRADUATE THESIS

Structure of dissertation

This document has been compiled during the process of development of the application as developer's log.

Previous to this final version, there were three versions correspondent to partial shipments of the project, each one integrating the contents of the past ones.

Thus, the contents as well, as the chapters, are ordered chronologically. That is, they follow and explain the project as it was in the moment of writing the given chapter, and such information has not been edited for the final version (excepting punctual additions, corrections and changes in Appendix 1). This is a dissertation that sums up documents and data from the past three and a half months.

Timeline of contents and shipments

The full shipment of this undergraduate thesis included the application developed, this dissertation and a presentation of the project. Both development and dissertation were developed and released by sections in partial shipments under a calendar designed by the professor (see Route book at Chapter 2 for further information).

Each partial shipment is called PAC (Prova d'Avaluació Continuada, in Catalonian) and implied the fulfillment of the following requirements:

PAC	CONTENTS	APPLICATION
1	Chapters 1st and 2nd Abstract.	
2	PAC1 Chapters 3rd – 5th Appendices 1 and 2 Bibliography	Alpha version
3	PAC 2 Chapters 6th and 7th	Beta version
Final	PAC3 Chapters 8th and 9th HTML Presentation	v1.0 Source files

Note on PAC3 and Final Shipment

As the reader will notice, this dissertation's final shipment date and the PAC3's shipment scheduled date are not equal. This is due to an error at the subject's calendar in the university campus, where no final shipment appeared and PAC3 took the role of final release. Fortunately, the error was detected and solved at the beta version release (with PAC3). I've left the project deadline as appears in the first chapters and the Gantt diagram as they were created and dispatched originally and enjoyed those couple of weeks more of extra developing time.

Copyright and credits

The application developed integrates some color functions from Justin Windle and m90, an unknown user from Kirupa forums (the code may have been modified partially for the sake of the application, detailed information further on this document). Also, for testing the application and demonstration purposes some third party visual data and work has been gathered and used from different public fonts under the doctrine of Fair Use in United States Copyright Law and academic purpose as specified by Spanish Intellectual Property Laws (Royal Legislative Decree 1/1996 of 12 April 1996). The copyright notes for this undergraduate thesis do not apply to those and other authors, products and/or software from third parties mentioned at this dissertation.

CHAPTER 1
INTRODUCTION

Definition of the project

The nature of the image

When looking at an image, firstly we perceive many elements scattered in a broad range of physical and psychological levels; and then comes our personal output from it: pleasure, disgust, indifference... After contemplating some creations from an author, we start developing a mental scheme of the complex mixture of visual entities such as forms, textures or lights that an author inserts into each and every creation and that is called author's style, author's mark or will of style.

Of such elements, color is one of the most relevant and variable ones. Many creators pass through color periods through their life (Picasso, for instance), for color is a powerful tool of expression, associated with both raw and complex emotional states.

But color is also used for many things: change the meaning of things, catch your eye or even attach an ethical or political trend to a logotype. Color can be used both consciously and unconsciously.

On the contrary, time is kept out of the picture when not a subject of the artwork itself. It's always there, because any art piece belongs to the history of the hand that created it, but the eye of the beholder usually forgets about the temporal dimension.

Thus, as in the dual nature of a quantum object -particle and wave at the same time-, we have two scopes of the nature of any image: color and time and, by them, not even Rothko can escape to us. Following the particle simile, color would be speed, time would be position and, fortunately, we can know both of them from our artistic particles[2]. We just need the appropriate tool, and that is what the project described at this document is about.

Starting point

The forest and the trees

At the internet, the amount of tools for displaying sets of images in interesting ways is huge. From gallery templates for websites to Flickriver widgets[3], they offer creative and beautiful ways of displaying and arranging imagery. Unfortunately, there are not so many centered in running analytical approaches to those sets, that is, there are not so many tools to add intelligence to such data. They are not *epistemological* tools, they are just digital frames[4].

Due to this lack of tools, it's difficult to get comprehensible information from a line of visual static artistic creations -images from now on this document- and their related information. Not to mention capable of analyzing everything done by an author, an artistic stream or other identifiable "hand", in order to portrait the lifeline of those creations.

For an individual gets incredibly complicated to produce this global view of the image set. That view turns, in many occasions, the goal by itself, because developing it into an instrument for further purposes requires too much time –and funds- for it to be profitable. As the speed of the image production is rising due to the increase of sophistication of production tools, the analysis gets more and more complicated. Even for the author him/herself. Can't see the forest for the trees.

This need of an "image of the images" is the starting point.

Goals

This project is intended to bring a tool that display information of a set of images in a visually comprehensible manner, as in an infovis[5], giving relevance to color and time parameters to help users analyze color similarities, production rhythms and other key data from a given image set.

The application should make the process of capturing and processing as automated as possible, getting the data from standardized files and offering the possibility of running server-side online with little modifications to the desktop version.

[2] As stated in the Uncertainty Principle of Heisenberg: http://en.wikipedia.org/wiki/Uncertainty_principle

[3] http://www.flickriver.com/

[4] See http://www.flickr.com/services/ for examples.

[5] As defined by Lev Manovich in his article "What is Visualization?", published at http://manovich.net/2010/10/25/new-article-what-is-visualization/

CHAPTER 2
SCOPE, TOOLS AND ROUTE BOOK

Scope

The scope is to focus on developing a fully portable all-in-one-file application as platform independent and centered in visualization as possible. The selection of technologies (see Fig. 1), process of development and layout design is key to success.

The technologies chosen are:

- ActionScript 3 for programming language: Flash-based applications run in the majority of computers and devices, is a strong, consolidated, language and more powerful and resource efficient to develop RIA than other languages. Also, it has been taken into consideration that Adobe is developing tools to convert it to HTML 5[6], which would make the application almost platform-universal.

- RSS format for retrieving the data. As RSS 2.0 is the most used version, includes media tags and Flickr[7] is the biggest photography (and image design) social network and repository, RSS feeds from Flickr sources will be used for the first version.

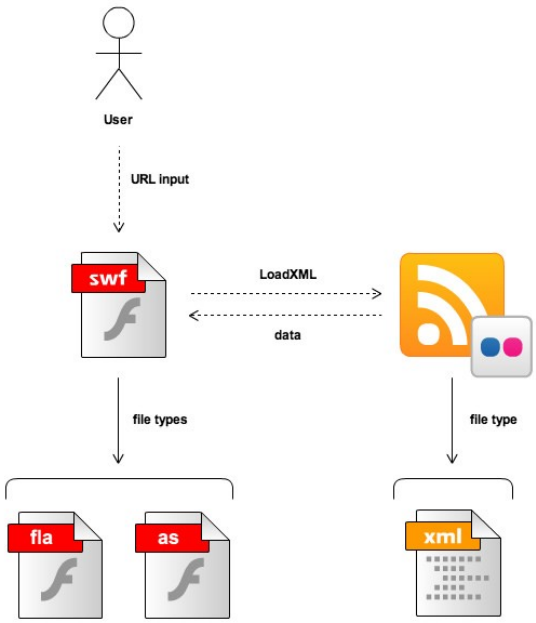


Fig.1. Application's main technological structure.

[6] http://labs.adobe.com/technologies/wallaby/
[7] http://www.flickr.com/

An example of the Flickr RSS feed is one from the Flickr account of the author of this thesis: http://api.flickr.com/services/feeds/photos_public.gne?id=19547713@N05&lang=en-us&format=rss_200.

Tools and technologies

The following technologies will be used for the development of the application, its publication and documentation.

Main software:
- Adobe Flash CS4
- FlashDevelop 3
- Adobe Dreamweaver CS4

Additional software:
- Microsoft Office Project 2007
- Microsoft Office Word 2007
- iMovie (Mac OS X 10.5.8)
- Adobe Illustrator CS4
- Adobe Fireworks CS4

Online services:
- Flickr

Languages:
- ActionScript 3.0
- RSS/XML
- HTML / CSS / JavaScript

Route book

The preliminary organization of the development is structured in a set of phases and milestones.

- Phase 1: Basic structure, functions and parameters. Load of data and capture of average color.
- PAC1
- Milestone 1: First data display.
- Phase 2: Custom image object.
- Phase 3: Polarizing. Capture and processing of additional data.
- Phase 4: Evolution of data display functions.
- Milestone 2: Alpha.
- PAC 2
- Phase 5: Interface design.
- Milestone 3: Beta
- Phase 6: Additional functionalities and display options.
- Milestone 4: Prerelease.
- Phase 7: Testing and clean-up. Project documentation.
- Milestone 5: Version 1.0.

- PAC 3: Shipment and End of project.

The development method, even lineal in time, would be visualized as a logarithmic spiral –see Fig. 2-: The process starts at the center and the phases grow to the exterior as time pass -the line of the spiral- and complexity grows -the distance between the turnings-.

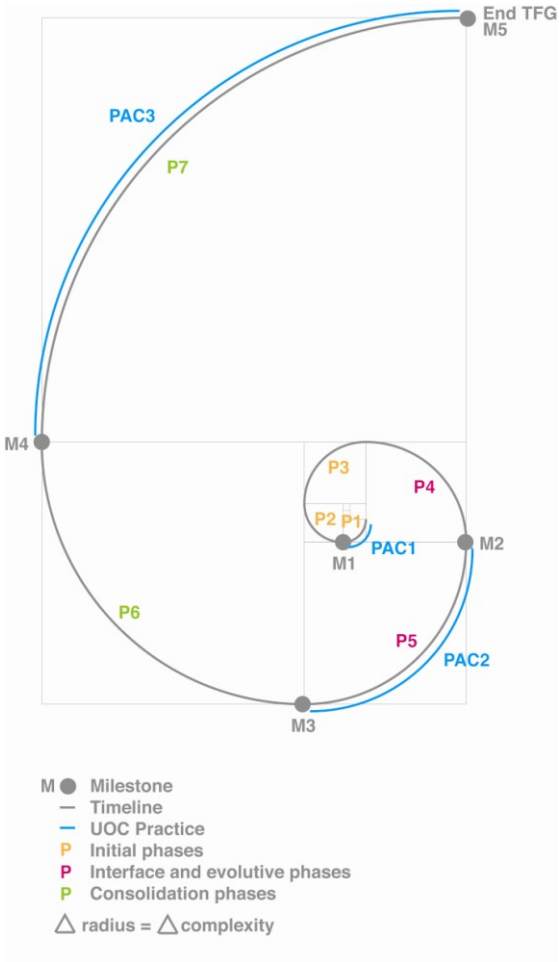


Fig.2. Diagram of the process of the project, the types of phases and its evolution in complexity.

Initial phases –from P1 to P3- are centered in sketching the project's goals and requirements, developing the technological basement and rising conceptual and technical questions both for these stages and further ones.

Secondary phases –P4 and P5- take care of evolving interface and basic functionalities, as well as data processing.

Tertiary phases –P6 and P7- should care about consolidating the code, adding functionalities, testing and documentation.

The logarithmic diagram is especially adequate for it portrays not only the increase in complexity –and time consumption- as a dimension, but also a very common need in software development processes: returning to previous points or elements in the development to correct or modify in other to comply with the requirements of further phases, as in a nearly circular development pattern. For instance, it would be possible to connect with curves different milestones and phases that are not linked in time.

In the Gantt diagram at the following page -see Fig. 3- appears the organization in time and phases for the application until version 1.0. It includes the shipment of PACs –practices- such as the present one, necessary for the academic evaluation of the progress of the project.

Student: Eva Casado de Amezua Fernández-Luanco [e_luanco@uoc.edu]
Research Advisor: Daniel de Fuenmayor López [dde_fuenmayor@uoc.edu]
Professor: Carlos Casado Martínez [ccasadom@uoc.edu]

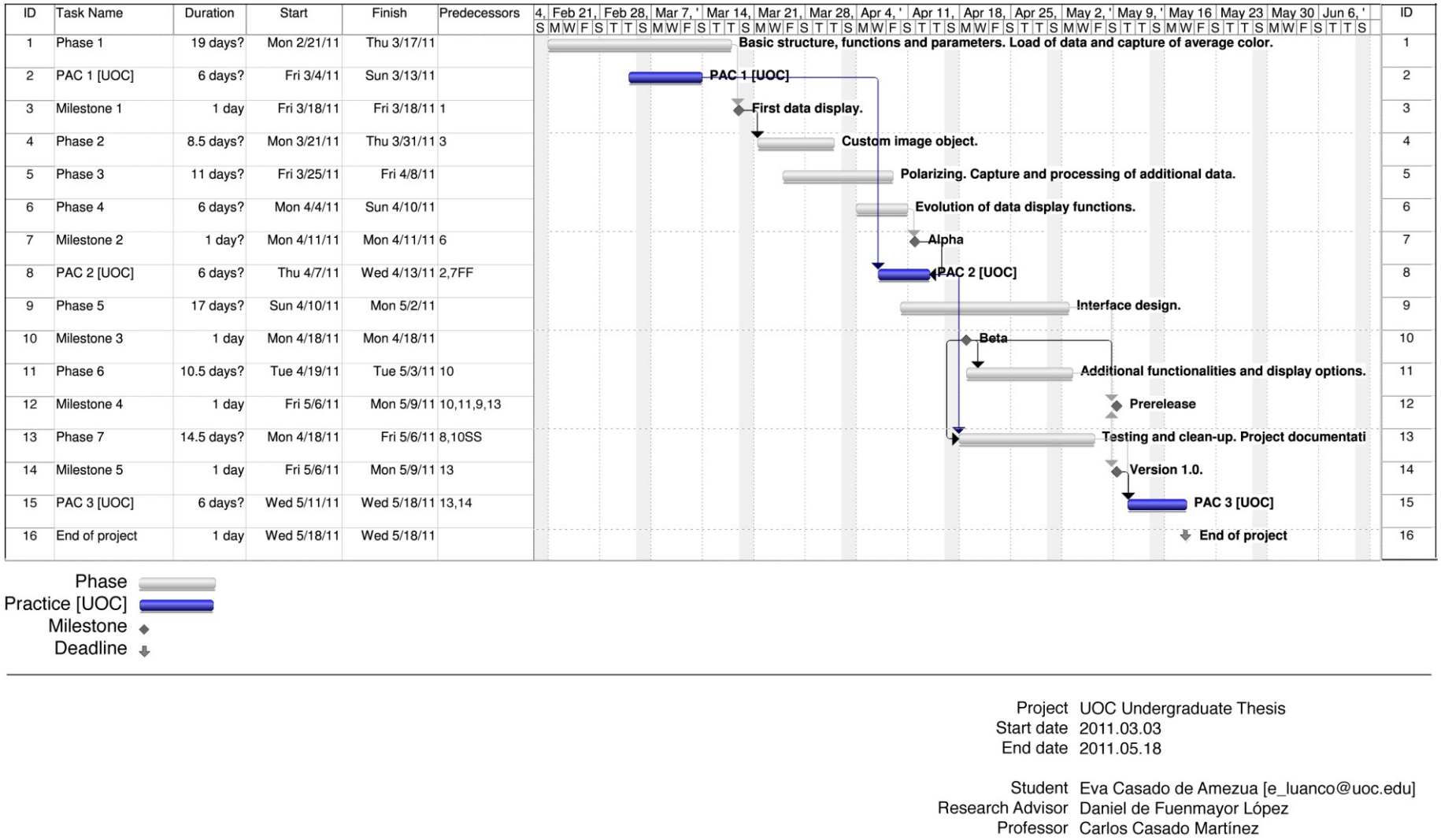


Fig.3. Gantt diagram of the project and PACs developed at the start of the project. Schedules from the conception of the application throughout its development until version 1.0.

CHAPTER 3
START-UP

Preliminary analysis

In order technology wouldn't distort the conceptualization of the project the first analysis was run mainly in paper, with punctual tests in AS3 (Adobe Flash) and research work through the Internet.

As displayed in the sketches at this chapter, an intensive work brainstorming ideas, both for visualization and logic of the application was done. The conclusions were:

- The application must offer as technical color data of the image as tools to discover patterns in the global palette beneath the images.
- There are many display possibilities. Thus, given the goal is a first version and the deadlines, the programming must be as focused in reusing code and integrating third party display libraries as possible. This is specially relevant for future versions of the application and in case other developers want to use it for their own projects.
- The application must have a standardized, classic interface/graph display section and one or more other sections focused on portraying the set. The standardized section is useful for users to have a fast and scientific approach to the data and the non-standardized one is useful as epistemological tool to visualize the data in new ways.

First tests and visualizations

The first work done in Adobe Flash was creating a Main.fla and it's related AS file, making it load a RSS URL and extract the images to load them at the stage.

The first significant problem was the very RSS format. It's most advanced version until now (2.0) was at the same time a foundation for the application and a source of problems.

RSS 2.0 format allows non-standardized XML tags. This, which is a blessing for services such as Flickr, who are using it for tagging their images with <media>, means that there is no standardized way of including images (URLs for JPEG files) in an RSS document. Each service creates its own standard but, fortunately, the <media> tag usage has been popularizing and, in general, one should just explore this tag and its children to get image data, if present in that RSS <item>.

Translated to AS3, this implies looping through the tag (with namespaces) and its children looking for a JPEG URL that leads to an image file of a reasonable size (not too small, nor big) for analysis purposes (see Appendix 1 for code).

Once loaded the images, the next logical step was extracting color data. It's important to bear in mind that Adobe Flash technology is, at graphic level, specialized in drawing, that is, creating forms, many times vectors, and giving them attributes such as color. But things like extracting the color of a random colored graphic, not to mention further data from a bitmap, are quite difficult because the built-in classes and functions in this matter are few.

Consequently, people tend to use other developer's custom classes or create their own ones (unusual). For the sake of a fast development and that it's quite absurd to reinvent something already done, I'm going to use some color functions from Justin Windle, a great developer who've studied in depth color at Flash[8] and include my own add-ons.

By now, the color functions I've used are `averageColour()` and `isCold()`, plus another function for reducing the number of colors to be processed. See appendix 1 for learning their code.

However, I don't find the resulting average color highly satisfactory, for it is a raw average method, not taking into account the relative weight of some colors at the images (for instance, shadows are usually present but not important, turning saturation value futile). Even when the resulting average color is clearly related to the image, the tone is usually altered and shadowed. Obviously, I'll have to run further work on this for the 1.0 version.

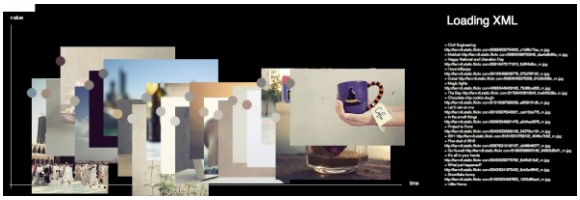


Fig.4. Load of a feed from Flickr. The points at the left of the images display their average color.

8 http://blog.soulwire.co.uk/code/actionscript-3/colourutils-bitmapdata-extract-colour-palette

Polarizing

If we are to display different values and measures in a standardized way, it'd be useful to do it in a single coordinate system in order of getting the maximum information in the minimum of space. Alas, one of the mayor problems with painting several numerical data in a single coordinate system is the noise provoked by the same data you are willing to portrait in an ordered way.

Researching about this I found an interactive infography created by the team of MOKI.tv[9] in which an incredibly amount of data was combined and displayed in a highly comprehensible structure. The whole infography was based on the concept of polarizing, where different values for each item were combined to calculate its y coordinate in a coordinate system (see chapter 5, Interface section).

Generally speaking (at least regarding visualization), polarization is the rate of separation from zero or any point of reference at an arbitrary dimension, which is calculated using other dimensions in a statistical formula or algorithm.

Saving the obvious differences in subject and data, I pretend to use the concept of polarizing for adding value to the visualization possibilities of the application.

At the moment of writing this paper I'm still evaluating the possibilities of polarizing but centering in extracting, calculating and painting data directly extracted from the bitmap object. I will not create polarizing functions in the application until it is in beta stage, because they would be pretty useless and incomplete given the fact that I'm still extracting data and developing display tools.

The solely approach I've made in this subject is displaying the average value between hue and value for each image. With no doubt difficulties on the calculation of polarity will arise, for instance achieving a resulting range of values with significance or getting them integrated in a given diagram, but I keep my hopes up in this matter.

9 http://moki.tv/blog/visual-evidence-movies-are-getting-worse

CHAPTER 4
ALPHA VERSION

Development system

Given the fact that Adobe Flash CS4 is not the best development application for projects with at least a bit of complexity, or several associated classes, I migrated the project to FlashDevelop, a useful open source AS3 editor (between other languages supported). As it runs on Windows, I used the program in a virtual machine running on VMWare Fusion and compiled with Adobe Flash CS4 in Leopard to see results and detect errors. Curiously enough, this seemingly complex method is faster than using just Flash CS4 thanks to the benefits that FlashDevelop offers (assistance to programming, indexing of classes, functions and objects, comfortable navigation system, etc.).

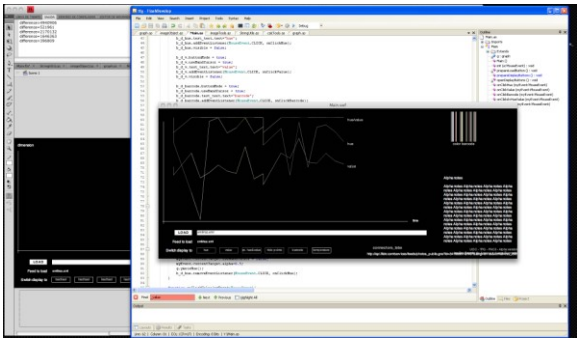


Fig. 5 Compiled application in Leopard over FlashDevelop running on Windows (VMWare, unity mode). Behind, Adobe Flash CS4 with the Main.fla file to compile and the AS files to locate errors (read-only mode).

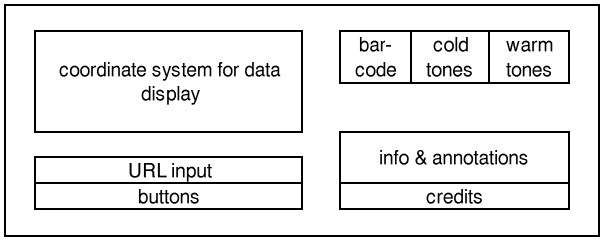
Also, after testing the loading method against RSS URLs, I downloaded three RSS feeds with images and stored them in local XML files for the sake of loading speed. The feeds are from two different sources: Flickr and FeedBurner, to avoid programming exclusively for a single format, which may cause problems when loading others with different image tagging approach, as explained in chapter 3.

The URLs of those feeds are:

- http://api.flickr.com/services/feeds/photos_public.gne?id=19547713@N05&lang=en-us&format=rss_200
- http://api.flickr.com/services/feeds/photos_public.gne?id=34196861@N07&lang=en-us&format=rss_200
- http://feeds.feedburner.com/Ellaing

Interface

As the interface has a relevant paper in the development and display of the application, it's not yet finished. By now spatial disposition of the available to elements is as follows:



Obviously, once a stable beta version is released a wireframe for the final interface will be designed.

Fig. 6 First stable interface version.

Structure of files

The file structure is:

```
Main.fla
Main.as
Main.swf
tfg.as3.proj
/com
    /core
        graph.as
        imageObject.as
    /utils
        calcTools.as
        imageTools.as
        stringUtils.as
/xml
    rss1.xml
    rss2.xml
    rss3.xml
```

All AS classes excepting Main.as are stored in com directory. XML files containing the feeds mentioned are stored in their own separated directory.

The file tfg.as3.proj is the project file used by FlashDevelop to manage the project.

Main FLA file

This file contains default visualization elements and tools, such as the coordinate system, the URL input text field, the buttons that trigger the different displays of data and static text such as version data and credits. The file does not contain programming at all.

Main class

The class which the FLA file associates directly with. It enables the elements on stage and their listeners, creates a Graph instance and triggers the load of the URL at the input text field into that object.

Graph class

This class models the graph object for the application, that is, contains all the data and main functionality for analyzing and displaying a set of images, including an array for storing image objects. Its key variables are:

- `xmlData`: where the XML is loaded into.
- `items`: array of image objects.
- `itemsDisplay`: movie clip where the individual movie clips of image objects are loaded into.
- `connectorsDisplay`: where the movie clips with lines that connect image objects are loaded into.

Its functions can be separated in two kinds:

- Load functions: those used to load the XML, extract data and store it.
- Display functions: position image objects' movie clips on the coordinate system at the stage, create their connectors and load additional displays (at present, barcode and temperature).

Image object class

For the sake of application modeling and image's bitmap and data storage I've created a class that should contain all the information regarding the extracted image.

This class is core in the application functioning. It's composed essentially of parameters and getter and setter functions. The main parameters are:

- `iUrl`: the image's URL.
- `iBitMapData`: the image's bitmapData.
- `loader`: a Loader object with the actual image.
- `iAverageColor`: the average color as calculated from Justin Windle function (uint type).
- `hsv_hue` and `hsv_value`: hue and saturation values for the image, calculated from my own functions (int type).
- `image_mc`: the movie clip created to display the average color and behave like a button to show additional data if clicked or rolled over.
- `imageCold`: a boolean variable that stores whether the average color is warm or cold, as calculated from Windle's `averageColour()`.

The key function of this class is `createMC()`. It creates the movie clip that will be displayed on stage with the image's info and operates in this order:

1. Sets `loader` visibility to false and add it to `image _mc`.
2. Prepares the space for the info, makes it invisible and adds it to `image _mc`.
3. Draws the bullet for displaying the average color of the image.
4. Creates the text with the info and add it to its space.
5. Adds the bullet to `image _mc`.
6. Enables `image _mc` as button and add listeners to it in order to turn info and/or `loader` visible on stage.

This order is compulsory for the different children added to `image_mc` behave as desired.

Utility classes

I'm separating some of the functions into utility classes to improve application modeling and code reutilization:

- calTools: functions that calculate and return numbers. The most relevant is `processDate()`, which turns the original date from the XML, difficult to read and with useless data such as the day of week, into a integer storing the date in ISO 8601 YYYYMMDD format[10], which is easy to use for comparison and x-coordinate positioning.
- imageTools: contains Justin Windle's functions and will contain also others now in imageObject class, such as those calculating HSV and RGB values.
- stringUtils: to store string functions related to texts and labels. See appendix 1 for learning `prepareTextField()`, the unique function at this class for the moment.

[10] http://en.wikipedia.org/wiki/ISO_8601

CHAPTER 5
TOWARDS BETA

Pending

The major tasks, capacities and errors yet to be solved are:

- A more pure and object oriented modeling and programming.
- Evolving the average color calculation for a more faithful range of colors output. Then saturation values will also be meaningful enough to be displayed.
- Display RGB data.
- Evolving the image info and barcode displays.
- Establishing a formula for polarization.
- Designing a definitive interface.

Known issues

The most remarkable bugs and errors are:

- The coordinate system not the additional displays adapt to the length of the set (load the third XML in the Alpha for an example).
- The error above is a direct consequence of a hardcode positioning of some elements in the stage, such as barcode and temperature grids.
- Info movie clip for each bullet should be visualized on top of the rest of movie clips. A swap depths function is necessary.
- There are yet some issues regarding the positioning of bullets when dates are equal: the distance between them sometimes is not enough for a clear visualization. A minimum distance to keep them apart is necessary.
- Display buttons appear too early, before the end of data load. This means an impatient user can click on them, stop the load and get diagrams with errors.
- The visualization of connectors is messy and unclear. They should behave like buttons and change visibility or transparency, for instance, in order to improve readability and usability.
- Dark bullets and connectors are difficult to see.

Interface

As mentioned before in this document, the interface has a long way to walk yet. However, as functionality and possibilities are getting into focus, a wireframe design will be the next task to be done in this area.

Even if the interface is at a very initial state, a research of style and usability has already been done for this application. At a conceptual level, its design will follow shokunin kishitsu[11], the Japanese craftsman's spirit: it should be simple, elegant and usable to the extreme, making itself disappear from the eye-hand as much as possible in order the true purpose, in this case the data, gets all the attention.

Some examples of interfaces that I consider may fit the purpose of this application are:

Fig. 7 Comparison of Vogue Magazine covers from UK and USA by Culture Maps.

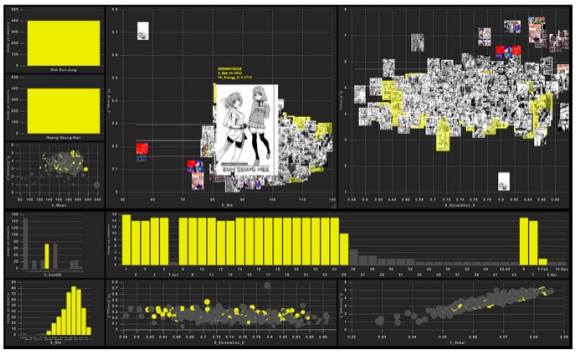


Fig. 8 VisualSense by1 Culture Maps.

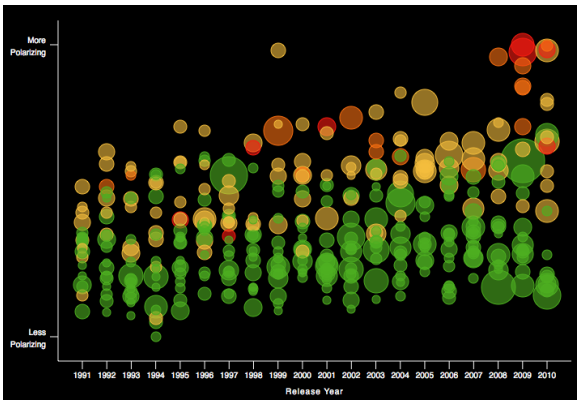


Fig. 9 Movies' quality analysis by MOKI.tv.

[11] http://37signals.com/svn/posts/2115-shokunin-kishitsu-the-craftmans-spirit

Fig. 10 Geckoboard interface.

Fig. 11 Colorendal by Mitsuhiro Sugimori.

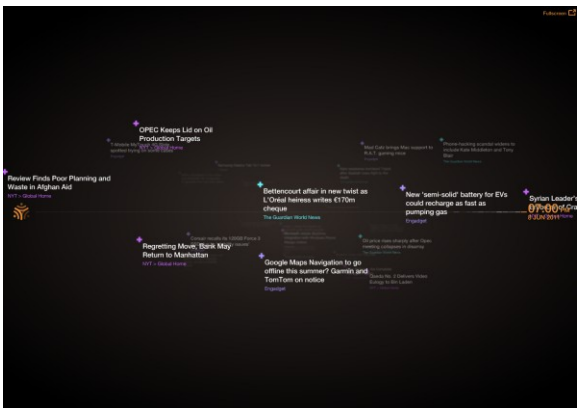


Fig. 12 Voyage RSS reader by Andy Biggs.

CHAPTER 6
BETA VERSION

Solved from alpha

Major issues solved from Alpha version are:

- The average color calculation has been evolved for a more faithful range of colors output.
- Saturation values are now displayed and used at the y-Axis.
- RGB and HEX values displayed.
- Evolution of barcode displays.
- Added function to swap depth between bullets to improve visualization.

Local versus remote RSS

The application supports now both local and remote RSS feeds without need of recoding. However, there are quite some restrictions yet:

- Local feed files must be stored in folder /xml.
- The images of the local feeds must be stored in /xml/rss folder.
- Not all remote feeds are valid. For now only feeds with media:content tags can be parsed.

Also, the applications behavior and response changes depending on the origin of the feed to be loaded:

- Local feeds' data loads faster, but also means (due to how the application is programmed) that images, if they are big, get into processing row fast and gives some memory management and performance problems to the application.
- Remote feeds are slower to load for images must be retrieved from remote servers. This is slower for the XML parsing but more amiable with performance, since the images are processed almost at the moment they are loaded.

The feeds for testing purposes are:

Local	Remote
rss1.xml	http://api.flickr.com/services/feeds/photos_public.gne?id=34196861@N07&lang=en-us&format=rss_200
rss2.xml	http://api.flickr.com/services/feeds/photos_public.gne?id=19547713@N05&lang=en-us&format=rss_200
-	http://feeds.feedburner.com/Ellaing

Files structure

The present file structure is:

```
Main.swf
Main.fla
Main.as
tfg.as3.proj
/com
    /core
        graph.as
        imageObject.as
    /utils
        calcTools.as
        imageTools.as
        stringUtils.as
/xml
    rss1.xml
    rss2.xml
    /rss
        *.jpg
```

Added functionality

Thanks to the correspondence with Daniel de Fuenmayor, a new analysis function has been developed (`prevailingColorFamily()` and related display function `displayMeanings()`): emotional and psychological meaning of color families at the set and their relative weight visualized as font size. Given the fact that color meanings is highly dependent on cultural roots, the output may not be astounding, but no doubt it'd be quite interesting, especially for those interested in design, marketing or anthropology. See appendix 1 for reading their code.

___

Also, commodity details and functionalities have been added, such as:

- On mouse over a color temperature bullet, the correspondent image's bullet displays its data.
- On mouse click on a color temperature bullet, the correspondent image's visibility switches to true.
- On mouse over color family over, all images' bullets from that family displays their data.
- On mouse click a color barcode, appears the HEX code for that color.
- On mouse over images, their correspondent image's bullet displays its data.
- On mouse click on image, its visibility switches to false.

Interface

The interface has undergone important changes that go from color of background to display data modules' design.

___

Interface's background color: Black presented the big problem of affecting to dark bullets' visibility. Contrary to the interface examples analyzed in previous phases, at this application the variety of colors is far higher, including dark tones range. Thus, black or dark grey are no neutral form the point of view of data display and a new interface color range had to be chosen. Beige and light tan tones (in varieties next to grey but not equal) were advisable, since they are used in designs for backgrounds were a neutral, paper-like disposition, is required for the sake of other color's visibility. Furthermore, these tones do take the characteristics of colors around them, almost mimetically.

___

The y-Axis display model has been inverted: In previous versions, higher hue and value where displayed at the bottom, near the x-Axis. Now a more intuitive, from zero to infinite style where zero touches x-Axis is used. Also, in previous versions one function was employed for each y-Axis dimension (hue, saturation or value), and now they are unified into a single one, `yAxisGlobal()`. See appendix 1 for reading its code.

___

Bullets are now tinted in two colors and have a concentric square form. The external color shows the average color for the image and usually refers to tones that give the image the prevalent look and feel, and the color of the inner square shows the first featured color, that is, a color highly used in the image but with some contrast (if possible) with the average color. See appendix 1 for reading the code of the involved functions, `uniqueColours()` and `getFeaturedColor()`, and further explanation.

___

Additional data displays: The disposition and part of the geometry have been changed in order to adapt and integrate with the overall interface style and wireframe structure, which tends now toward horizontality.

___

Credits: Now hidden when loaded, include a basic help for novices. It can be opened clicking on the name of the application.

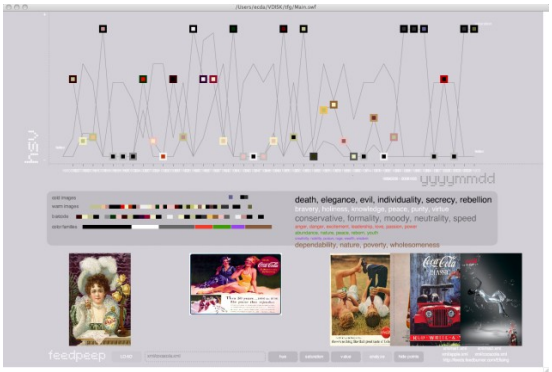


Fig. 13 Beta version loading a brand feed stored locally (advertisements of Coca-Cola).

Name of the application

No definitive decision upon the application's name has been decided. A temporal name, FeedPeep, is used for the sake of look and feel design process.

Other uses for the application

Since any image set in RSS format, given they dispose of temporal data, can be loaded into the application further image analysis purposes can be achieved:

- Analysis of the style of a brand, its featured images or advertising style through time.
- Analysis of products from a brand through time from the point of view of their featured images or photographs.
- Analysis of a video sequence through its frames.

A set of feeds to serve as examples is being compiled. Two of the feeds (about Apple[12] and Coca-Cola[13] advertisements) and the required images can be downloaded from:

http://www.ellaing.com/tfg/tfg_xml_test_feeds.zip
(18 MB)

The sets must be saved into /xml and /xml/rss folders as detailed in this chapter. The application will take some time to load them completely and the XML files are not finished; thus, errors might arise when testing them. A stable version of them and others will be available at version 1.0 release.

12 http://www.apple.com/
13 http://www.coca-cola.com/

CHAPTER 7
TOWARDS v1.0

Pending

The major tasks, functionalities and issues yet to be solved are:

- A more pure and object oriented modeling and programming.
- Evolve the image processing functions.
- On the coordinate system display, more than two images with same date have potentially overlapping issues. The fact that `yAxisGlobal()`only compares dates between the present image and the last one, not the next one, is part of the problem.
- Further beta-testing, especially on display and analysis functions.
- Normalization through a better naming of functions and variables.

Known issues

The most remarkable bugs and errors are:

- Some issues regarding bullets positioning. Bullets with same date are applied an offset that avoids them to be in same position, piling themselves up, but it distorts the graph and makes it unfaithful.
- Display buttons appear too early, before the end of data load. This means an impatient user can click on them, stop the load and get diagrams with errors.
- Data load gets too slow in long feeds (+30 items) or in feeds with big images. At least a function to pre-process the image reducing its size it's advisable.
- The visualization of connectors is messy and unclear. They should behave like buttons and change visibility or transparency, for instance, in order to improve readability and usability.
- The positioning of bullets under saturation values isn't satisfactory. The resulting graphic does not portraits in a rapidly comprehensive way the changes in values. Probably, the solution will have something to do with making its display values relative to the saturation range of the set.

Beyond version 1.0

One of the major goals to achieve would be to duplicate the application, that is, compare two sets of images. This has two deep implications in the application:

- A very refined and efficient OO programming is necessary. Encapsulation and reutilization are critical.
- And, also, a study on the implications for the display of time dimension as x-coordinate. Many questions arise from a scenery where two sets are compared: If the sets are completely separated in time... does it have meaning maintaining a single-time X axis? Do real dates have any relevance at all? Would be convenient for the sake of visualization a formal, artificial, data conversion for displaying in a single timeline? I don't expect version 1.0 to compare sets in a single coordinate system but in two separate ones, but such questions, important for further development, don't disappear.

CHAPTER 8
VERSION 1.0

Application functionality

At version 1.0 the application extracts data and load and process images from XML files compliant with RSS 2.0 model. Its main features are:

- Extracts, calculates and displays RGB, HSV, temperature, average color, color family and HEX code.
- Separates relevant colors from those similar to display a color palette for the set.
- Arranges item's representation and data ordered by time (in distances proportional to time gaps) and HSV values.
- Displays thumbs of images and turns them into buttons.
- Displays the psychological meanings of color families in the images and displays them with size proportional to relevance in the set.
- Offers visual display options for same information in order to gain knowledge of items both individually and as group.

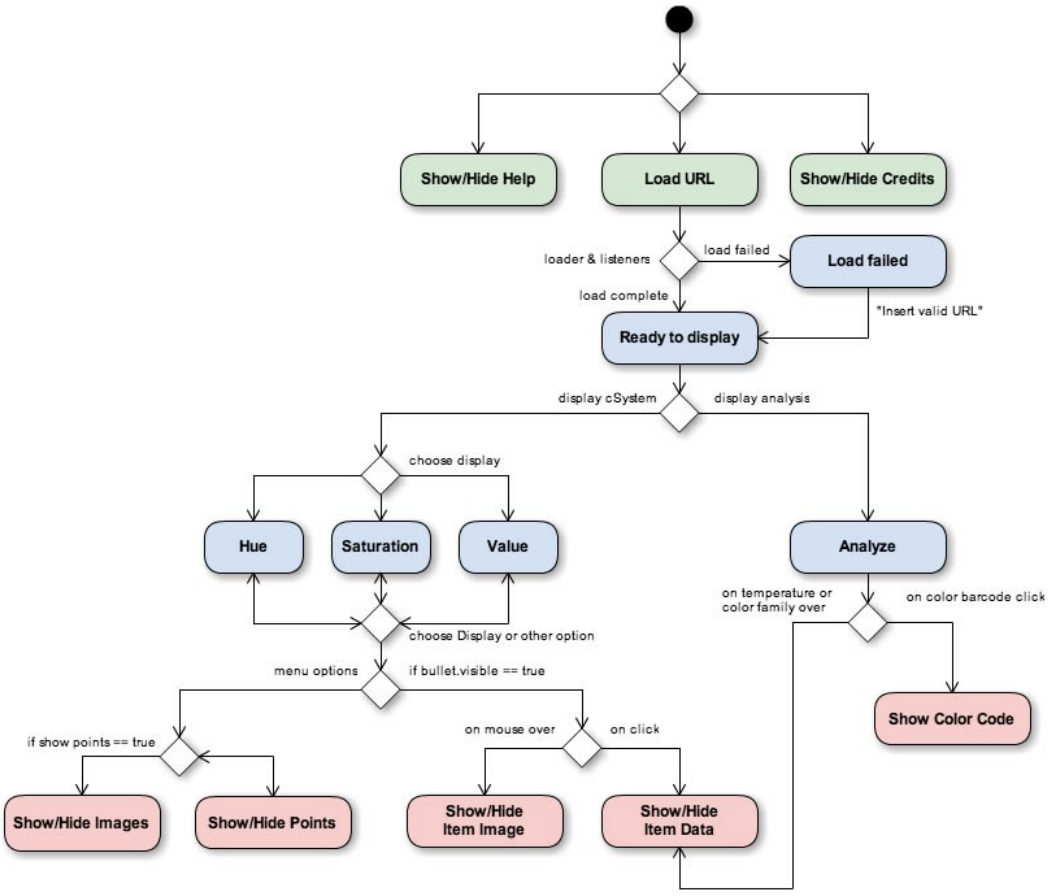


Fig. 14 State diagram (only main nodes).

Application structure

The version 1.0 file structure is:

```
FeedPeep.swf
FeedPeep.app
Main.fla
Main.as
tfg.as3.proj
/com
    /core
        graph.as
        graphItem.as
        imageObject.as
    /utils
        calcTools.as
        imageLoadEvent.as
        imageTools.as
        stringUtils.as
/xml
    apple.xml
    bsg.xml
    cocacola.xml
    disney_films.xml
    rss1.xml
    rss2.xml
    runaway.xml
    /rss
        *.jpg
```

Usage recommendations

The application is offered in two file formats, SWF and APP (AIR application) which imply usage differences.

For SWF format:

- Local RSS/XML files can be loaded. The feed files must be stored at /xml and the images at /rss. The images must be in JPEG format and their name must be put into <media:content> tag, url attribute.
- Remote feeds can be also loaded.

For APP format:

- Only remote feeds can be loaded in version 1.0.

When running Main.swf file Adobe Flash Player may require adding the location of the file to its Setting Manager, section Global Security Settings. The application must be tagged with "Always allow", otherwise Adobe Flash Player will not let it access Internet for the images.

About creating local XML feeds their process of creation implies:

- The pubDate field must match the date for that image.
- In case there are no dates, the time gaps are not important or we are talking about studying frames from a video, each item must have a pubDate of a single day, for items with equal dates are position in same x coordinate. For instance, if a set have 30 images, we can use the dates of a 30 day month, each day for an item. This way the distances between bullets are distributed evenly.
- The use of one of the XML feed files included in this project as examples is recommended, for they are already tested in the application and free of unnecessary data.

Files for local analysis

A set of RSS/XML files with their correspondent images for analysis are available at http://www.ellaing.com/tfg/tfg_xml_test_feeds.zip and include the following feeds:

- Two Flickr photostreams copies (from users Reem and She-Noir).
- Apple printed advertisements (1977-2009).
- Coca-Cola printed advertisements (1889-2008).
- Disney animation films' original posters (1937-2011).
- Frames from the short film Runaway by Kanye West (the full version of a music video).

Remote feeds

It's important to remember that not all feeds comply yet with RSS 2.0 specifications. Feeds that, excepting punctual cases, are error free are those from Flickr, FFFFound and FeedBurner (those configured to include images in the feeds). Some examples of online feeds that can be loaded into the application are:

- http://feeds.feedburner.com/Ellaing
- http://api.flickr.com/services/feeds/groups_pool.gne?id=1498905@N25&lang=en-us&format=rss_200
- http://feeds.feedburner.com/life/todaystopphotos
- http://feeds.feedburner.com/life/editorspicks
- http://feeds2.feedburner.com/elise/simplyrecipes
- http://feeds2.feedburner.com/time/today-in-pictures
- http://ffffound.com/home/thatdayinthepark/foun

d/feed
- http://ffffound.com/home/strangeone/post/feed
- http://feeds.feedburner.com/time/photoessays
- http://api.flickr.com/services/feeds/photos_public.gne?id=13197057@N06&lang=en-us&format=rss_200
- http://lookbook.nu/rss

Main issues solved

Serious error at items array ordering. The item id didn't match with the load order, for the id was inserted in the creation of the item but the order was used in `fillImagesArray()`. Now the id is stored *after* ordering the array, not before. This is especially necessary for the correct functioning of `activateItemsDisplay()`.

___

Another important issue, probably the biggest, was at the function that processes dates. The problem is related in AS3 specification of Date type, in which dates are always related to a time model, being it locale or UTC and, once created the object, it processes the date inserted taking into considerations things like summer/winter hour changes, etc. Featured links on this matter are available at Linkography section. In practice, this issue made the application changed item's dates, making the pubDate in int type different to the original date. Given the fact that the important time information for the application is the difference in days between items and the years implied, my solution follows three steps to process the date correctly (see Appendix 1 for code):

1st: `Date.toString()` converts the date to an standardized string[14].

2nd: The function splits the string into datetime elements and stores them into an array.

3rd: Turns resulting strings into int type and calculates the ISO 8601[15] format (YYYYMMDD) as a single number. The higher the number, the closer to present day.

___

Problems in saturation display: Sometimes no values could be calculated (returning NaN), this can be for the function is wrong or the call to the function is too fast and for hue value extraction call (saturation is calculated with hue). Also, HSV functions are in graphItem class when they should be in imageTools class. A different HSV calculation function was required. I browsed the web and found first one at http://www.cs.rit.edu/~ncs/color/t_convert.html but didn't convinced me, so I ended up using my own version of a function published at Kirupa (http://www.kirupa.com/forum/showthread.php?t=322958). See Appendix 1 for code.

___

Issue in Analysis display: When many items are loaded, temperature, emotions and other movie clips did overlap. I checked their widths and heights and constrained them to 520. It's hardcode, not elegant at all, but works and it's enough for the purposes of a first stable version.

___

If Show/Hide Points button was clicked before selecting visualization display (hue, value or saturation), itemsDisplay remained hidden. Solved with flags.

Main improvements

Added a new function to visualize all images with one button click on Show/Hide Imgs (`onClickImages()` and `switchImagesVisibility()`).

___

Added a help section with definitions of displays and elements. The visual style of help elements is black on yellow in order to distinguish them easily from the data below. The yellow has been chosen not only for it is very bright, contrasting with grey tones, but also because it has reminiscences of post-it papers, getting their meaning of "helpful annotations on main data".

___

On input text field, now it gets disabled for selection/writing once loaded starts and doesn't change back unless load fails. Also, once the set is loaded, the image set title can be read at the input/output text field.

___

For avoiding painting connectors one and again I've added a flag control to `yAxisGlobal()`:

```
if (p_v == "hue" &&
flag_connectorsHue==false) {
      drawAllConnectors(connectorsDisplay);
```

Creation of a preloader that runs while the application is loading the XMl and processing items' images:

[14] http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/Date.html#toString%28%29

[15] http://www.iso.org/iso/support/faqs/faqs_widely_used_standards/widely_used_standards_other/date_and_time_format.htm

`preloader()`.This capacity is especially important for two reason: first, it informs user about the evolution of the load and, second, it gives an understanding of the load process in slow loading situations, for instance in case the images to process are big or the number of items is high. See Appendix 1 for code.

About mentioned load completion delay situations, a limitation of items/images to load has been added to Graph class: 20 for remote, online feeds, and 80 for feed files archived locally.

___

A process for scaling BitmapData is executed before processing. It boots up slightly the loading speed, which is especially noticeable at remote RSS feeds with large images

___

Display related buttons now do not appear until the feed has been loaded completely. I've added to `yAxisGlobal()`:

```
if (flag_itemsVisibility == false) {
      switchItemsVisibility();
      }
```

___

Now all Analysis displays are stored into a single mc and they refer to the upper movie clip display (temperature) for calculating their x and y coordinates.

___

Data at bullets now always appear over the rest of elements when mouse over the bullet. Added to `switchDataVisibility()`:

```
bringToFront(this.get_image_mc(),
```

The loading issue

Unfortunately, the load of many RSS feeds with images may fail due to two reasons:

- Publisher's attitude on RSS formatting has been, at least, highly relaxed. Most of them do not comply with standards and rely on third party software to deal with their errors. Even now, AS3 libraries are in develop to solve issues related with feed formatting. Which leads us to the second reason.
- I haven't run enough tests on this matter to create exceptions that would cover the majority of possible issues. This has been due both to my increasing focus on local RSS/XML files while developing and lack of time for debugging version 1.0.

I've made an effort for the code that looks for and load images behave the best way possible, jumping to next item if no image was found, looking for alternate locations for images, etc. However, it is far from error free. As usual in the software industry, version 1.0 will need an update to 1.1 for solving the loading issue as soon as possible. Regrettably, that will be beyond the shipment of this thesis.

J. Windle's colour functions

Some color functions by Justin Windle has been used for the application development (with punctual modifications) at the imageTools class:

- `averageColour()`
- `isCold()`
- `different() and similar()`
- `uniqueColours(`
- `indexColours()`
- `reduceColours()`

The full class they were taken from, ColourUtils, can be downloaded from:
http://blog.soulwire.co.uk/code/actionscript-3/colourutils-bitmapdata-extract-colour-palette

A selection of the functions used can be read in this document at Appendix 1.

CHAPTER 9
CONCLUSIONS

Retrospective

This has been a very interesting project that has taken the best of me, even things I wasn't aware to possess in relevant quantity (especially mind-control to relax when everything went wrong).

The project started with a clear simple idea and lots of ideals (which obviously, the first version of the application can't cover much), but the scope and utility of the tool has been revealing itself throughout the process quite smoothly. Even at the moment of writing down these lines, new ideas to try out pop-up in my mind. Now I see FeedPeep as a seed, with many opportunities to explore; far from completed, it's just starting to bloom.

I have only two regrets about this project, in the form of "I wish I had":

1. Started cleaning and debugging code far earlier. I'm a messy programmer (probably because I'm not old enough in the business yet) but enjoy an uncanny memory. This means I don't need the code to be much ordered because I know where everything is and what is for. If I were cleaner debugging and polishing, the development would have been faster. And time was scarce in this project for many reasons (some of them my own personal circumstances).
2. Explored different display possibilities. Along with the research of interfaces for the application, I imagined many concepts for displays very different, but decided on following a standard, scientific, look and feel, leaving many ideas for future versions (see for instance Polarizing concept in Chapter 3). Even if under productivity scope was a good decision, I think a far more spectacular and unique data visualization model is possible and that I should have, at least, experiment once with abstract, non intuitive, visualizations.

On the other hand, along the development process I've understood that this tool will probably be more useful for creating and analyzing feeds in local machine to study custom sets that reading feeds online (excepting Flickr feeds and suchlike).

The main benefit of this tool is to have in a very small file size a tool for gathering intelligence on a chosen set of images. In no time the user can create an XML with the link to the images to analyze and get "the big picture" of that set, the vector of intentions behind, the footprint of the hand that created it.

I think that can benefit from this tool specially scholars and professionals in visual arts and advertisement. At the next point I explain with an example one of its possible applications.

.

A practical example

Let's take one of the local feeds and have a fast look at what retrieves when loaded. For instance, a selection of frames from short film Runaway, by Kanye West:

When loading and displaying HSV and color analysis values, we can see the color pattern matches perfectly with the narrative of the film and that, curiously, we can divide it into three chapters perfectly distinguished:

- The first one, strong on shadows and reds, talks about the birth of love, strong emotions and a link to the end of the story.
- The second, clearly towards purples, yellows and greens, is related to the core of the story, were the main characters have to deal with society (in light colors, specially whites, as symbol of purity and social acceptance), and includes the acclaimed music video of the ballerinas dancing the song that gives name to the film.
- The third one is the catharsis, the tragedy (same dark colors of the start) and the outcome of the tale, where the nature of the woman is reveled (mostly with red and oranges, related to love and intense feelings).

In a color meaning level, West gives white a connotation of coldness and sterility, saturated reds and oranges of love and true feelings, and blacks and shadows of tragedy and sin. Given the fact that most of the shadows at the first third of the film are related to the end of the film (such feelings and vision danced by the ballerinas in black as announcement of what is to come), white in the narrative turns out to be the seed of darkness, that is, a homogeneous, static and strict society can only bring rebellion, destruction and breakout.

Thus, the application gives instantly a schematic but fairly accurate sketch of the set analyzed, both from a narrative, temporal and structural point of view but also going down to fine color details, connecting them to the set as a whole, reinforcing other communicative and formal analysis that the user may run with third resources and tools.

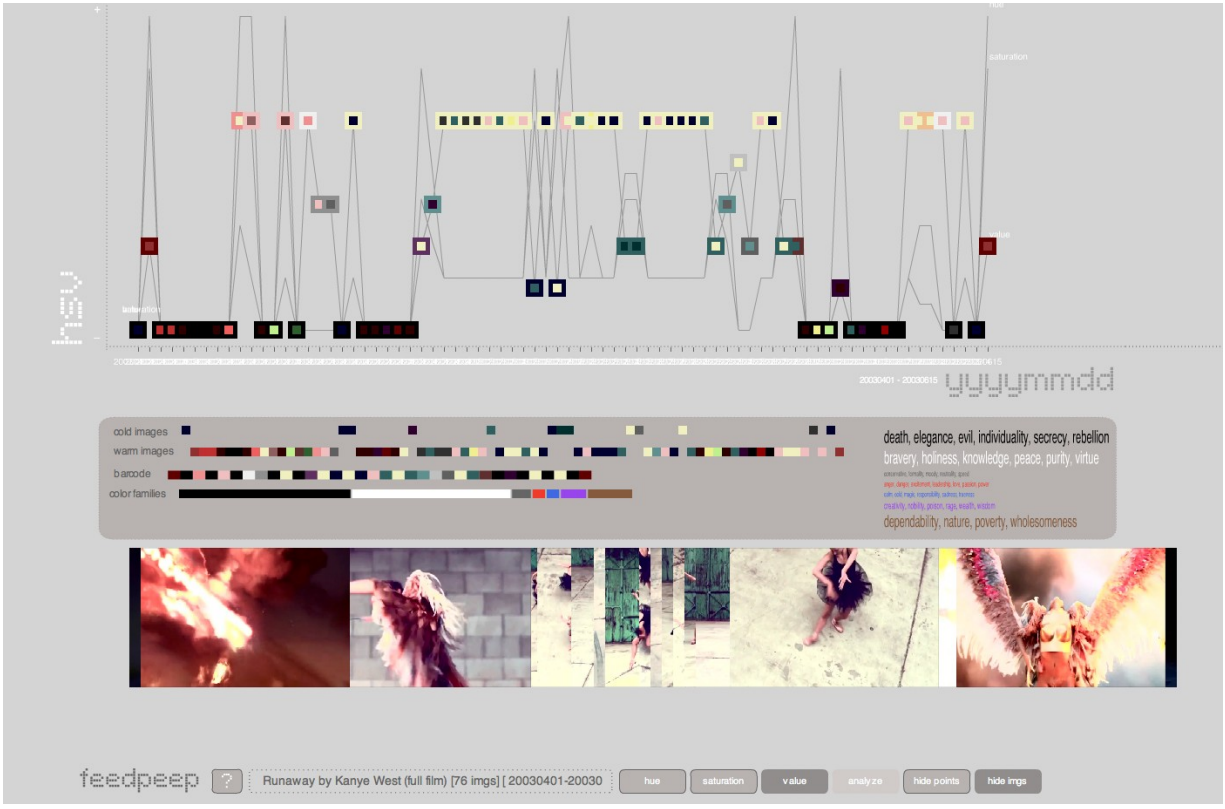


Fig. 15 FeedPeep displaying values from local feed of Kanye West's Runaway short film.

Towards future

Apart for the obvious improvements and bugs solving needed for future versions of the application (such as how offset in y Axis is calculated at `yAxisGlobal()`), some relevant features I'd like to add to enhance the application possibilities are:

- Parameterization on black/white sensibility to adapt to set heavy in shadows or lights.
- Parameterization of selected color to analyze HSV and color temperature: change to first featured color.
- Export resulting data to Excel (for starters, let's say displaying data in XML format at the SWF for the user to copy-paste it, this way we avoid using PHP in a first version).
- Display of information as fractals or networks.
- Compare two different sets with one instance of the application.
- In case of video analysis, to compare with sound parameters as volume, high pitches or instruments to learn how are they linked to color and meanings.
- Compile a full version in Adobe AIR.

Some of them, and many others I will imagine in the future, will take more effort or have priority over others, but I think that in an application that can be used both for data and artistic analysis the possibilities are endless, sky is the limit.

APPENDIX 1
FEATURED CODE

Parse media:content tag

This parsing function looks (in graph class) for images in the media:content tag of each item from an RSS feed, starting from the optimum, probable, image source (medium) to the less relevant and lowest in quality (thumbnail), creates an image object for each of them and run a function that triggers the loading process of data and movie clip for each object.

```
public function extractData(e:Event):void {
	stringUtils.writeTrace(0, "extractData");
	e.target.removeEventListener(Event.COMPLETE, extractData);
	xmlData = XML(e.target.data);
	var xmlTitle:String = xmlData.channel.title;
	this.set_imageSetName(xmlTitle);
	var media:Namespace = new Namespace(xmlData.namespace("media"));

	for each (var item in xmlData.channel.item) {
		var _pubDate:Date = new Date(item.pubDate.substr(0, 16) );
		var _descr:String = item.description;
		if(item.namespace("media") != undefined){
		var _imgUrl:String = "";
		if(item.media::content.@medium!=undefined){
			_imgUrl = item.media::content.(@medium=="image").@url;
			}
		if(_imgUrl=="" && item.media::content.@type!=undefined){
			 _imgUrl = item.media::content.(@type=="image/jpeg").@url;
			 }
		if(_imgUrl=="" && item.media::thumbnail!=undefined){
			 _imgUrl = item.media::thumbnail.@url;
			}
		}
		else if(_descr.indexOf("img")!=-1 && _descr.indexOf("src=")!=-1 &&_descr.indexOf(".jpeg")!=-1){
			var begin = _descr.indexOf("<img ");
			var after = _descr.substring(begin);
			var end = begin + after.indexOf(">")+1;
			_imgUrl = _descr.slice(begin,end);
			}
		if (_imgUrl != null) {
			var _image:graphItem = new graphItem(_imgUrl, _pubDate, this.itemsDisplay);
			items.push(_image);
			}
	}

	items.sortOn("processedDate", Array.NUMERIC | Array.DESCENDING);
	processDateRange();
	buttonSpace = (1200 / items.length) - 2;
	itemsDisplay.timeRange.text = firstDate.toString() + " - " + lastDate.toString();
	preloader();
	calculateMaxNumItems();
}
```

Selection of Windle's functions used

```
public static function averageColour( b:BitmapData ):uint{
	reduceColours( b, 64 );
	var R:Number = 0;
	var G:Number = 0;
	var B:Number = 0;
	var n:Number = 0;
	var p:Number;
	for (var x:int = 0; x < b.width; x++) {
		for (var y:int = 0; y < b.height; y++) {
			p = b.getPixel(x, y);
			R += p >> 16 & 0xFF;
```

```
                G += p >> 8  & 0xFF;
                B += p       & 0xFF;
                n++
            }
        }
        R /= n;
        G /= n;
        B /= n;
        return R << 16 | G << 8 | B;
}

public static function isCold( p_b:BitmapData ):Boolean{
        reduceColours( p_b, 64 );
        var R:Number = 0;
        var G:Number = 0;
        var B:Number = 0;
        var n:Number = 0;
        var p:Number;
        for (var x:int = 0; x < p_b.width; x++) {
            for (var y:int = 0; y < p_b.height; y++) {
                p = p_b.getPixel(x, y);
                R += p >> 16 & 0xFF;
                G += p >> 8  & 0xFF;
                B += p       & 0xFF;
                n++
            }
        }
        R /= n;
        G /= n;
        B /= n;
        return B>R;
}

public static function reduceColours( b:BitmapData, colours:int = 16 ):void{
        var Ra:Array = new Array(256);
        var Ga:Array = new Array(256);
        var Ba:Array = new Array(256);
        var n:Number = 256 / ( colours / 3 );
        for (var i:int = 0; i < 256; i++){
            Ba[i] = Math.floor(i / n) * n;
            Ga[i] = Ba[i] << 8;
            Ra[i] = Ga[i] << 8;
        }
        b.paletteMap( b, b.rect, new Point(), Ra, Ga, Ba );
}
```

Change y-coordinates for HSV values, global function

This function in graph class distinguishes which HSV display option has been chosen and calculates which values apply to items' bullets. The function treat the first item (`items[0]`) differently because for its case no previous item value can be used to compare and calculate its offset (that's the case for the rest, when items can share same date). I've left commented previous ways (faulty or not enough accurate) to calculate those offsets. Once all y coordinates are calculated, the the function calls connectors functions to draw the connecting lines and swap depths with their parent movie clip to make sure the bullets appear over the lines, not reverse.

```
public function yAxisGlobal(p_v:String):void {
        stringUtils.writeTrace(0, "yAxisGlobal");
        stringUtils.writeTrace(1, "flag_itemsVisibility", flag_itemsVisibility)

        displayImageSetName();
        if (flag_itemsVisibility == false) {
            itemsDisplay.visible = true;
            switchItemsVisibility();
            this.loadingMC.visible = false;
            }

        var cSystemHeight:int = 330;
        var yValue:int = 0;
```

```
	var yValue_Past:int = 0;
	var lastImageCoords:Point = new Point();

	items[0].get_image_mc().x = 0 + xOffset;

	if (p_v == "hue") {
		items[0].get_image_mc().y = ((-1)*items[0].get_hue())+cSystemHeight;
		}
	if (p_v == "saturation") {
		items[0].get_image_mc().y = (( -1) * items[0].get_saturation() * 300) + cSystemHeight;
		}
	if (p_v == "value") {
		items[0].get_image_mc().y = ((-1)*items[0].get_value())+cSystemHeight;
		}

	for (var i:int = 1; i < items.length; i++) {
		if (p_v == "hue") {
			yValue = ((-1)*items[i].get_hue())+cSystemHeight;
			yValue_Past = ((-1)*items[i-1].get_hue())+cSystemHeight;
			}
		if (p_v == "saturation") {
			yValue = (( -1) * items[i].get_saturation() * 300) + cSystemHeight;
			yValue_Past = (( -1) * items[i - 1].get_saturation() * 300) + cSystemHeight;
			}
		if (p_v == "value") {
			yValue = ((-1)*items[i].get_value())+cSystemHeight;
			yValue_Past = ((-1)*items[i-1].get_value())+cSystemHeight;
			}

		var iMc:MovieClip=new MovieClip();
		iMc=items[i].get_image_mc();
		var iMc_previous:MovieClip=new MovieClip();
		iMc_previous=items[i-1].get_image_mc();
		var t_date1:int = items[i].get_pDate();
		var t_date2:int = items[i - 1].get_pDate();

		if (t_date1 == t_date2) {
			items[i].get_image_mc().x = items[i - 1].get_image_mc().x; // ((i - 1) * buttonSpace) +
xOffset;
			var yOffset:int = yValue - yValue_Past;
			if (yOffset <= 30 && yOffset >=1) {
				yOffset = 30;
				}
			if (yOffset >= 30) { ;
				yOffset = 10;// yValue - yValue_Past; //Math.abs(yValue - yValue_Past);
				}
			if (yOffset <= 0) { yOffset = 10;}
			items[i].get_image_mc().y = yValue + yOffset;
			lastImageCoords.x = iMc.x;
			lastImageCoords.y = iMc.y;
			}else{
				items[i].get_image_mc().x = (i * buttonSpace)+xOffset;
				items[i].get_image_mc().y = yValue;
				lastImageCoords.x = iMc.x;
				lastImageCoords.y = iMc.y;
				}
		drawTimeLine(items[i], lastImageCoords.x , lastImageCoords.y);
		items[i].rearrangeLoader();
		}

	drawTimeLine(items[0], items[0].get_image_mc().x , items[0].get_image_mc().y );
	items[0].rearrangeLoader();

	if (p_v == "hue" && flag_connectorsHue==false) {
		drawAllConnectors(connectorsDisplay);
		flag_connectorsHue = true;
		}
	if (p_v == "saturation" && flag_connectorsSat==false) {
		drawAllConnectors(connectorsDisplay);
		flag_connectorsSat = true;
		}
	if (p_v == "value" && flag_connectorsValue==false) {
		drawAllConnectors(connectorsDisplay);
```

```
            flag_connectorsValue = true;
            }

      var ti:TextField = stringUtils.prepareTextField(p_v, items[0].get_image_mc().x-50,
items[0].get_image_mc().y-10,10);
      var te:TextField = stringUtils.prepareTextField(p_v,lastImageCoords.x-30,lastImageCoords.y,10);
      connectorsDisplay.addChild(ti);
      connectorsDisplay.addChild(te);
      addChild(connectorsDisplay);
      this.swapChildren(itemsDisplay,connectorsDisplay);
}
```

Draw single connector between the movie clips of two image objects

The previous version of this function (in graph class) drew a gradient line with the colors of the bullets to be connected. Even if it was visually appealing, it didn't offer additional service to the information already displayed, created noise and the lines were not always clearly visible. This is a simpler version, with lines in dark grey, which is fine for the purpose of the application.

```
public function drawConnector(b:int,a:int):void{
      var iMc:MovieClip=new MovieClip();
      iMc=items[b].get_image_mc();
      var iMc_previous:MovieClip=new MovieClip();
      iMc_previous = items[a].get_image_mc();

      var shape:Sprite = new Sprite();
      shape.x = 0;
      shape.y =0;
      shape.graphics.lineStyle(1,0x999999);
      shape.graphics.moveTo(iMc.x+30,iMc.y+50);
      shape.graphics.lineTo(iMc_previous.x+30,iMc_previous.y+50);
      shape.graphics.endFill();
      addChild(shape);
      }
```

String utility to prepare text fields

This little snippet at stringUtils class is very useful, and will be more in the future when getting polished: Creates dynamically a basic text field ready to use as label for connectors or displays.

```
public static function prepareTextField(t:String,cx:int,cy:int):TextField {
      var tf:TextField = new TextField();
      var f:Font = new Font1();
      var ff:TextFormat = new TextFormat();
      ff.font = f.fontName;
      ff.size = 12;
      tf.autoSize = TextFieldAutoSize.LEFT;
      tf.defaultTextFormat = ff;
      tf.embedFonts = true;
      tf.text =t;
      tf.textColor = 0xFFFFFF;
      tf.x = cx+60;
      tf.y = cy + 30;
      return tf;
}
```

Image function to infer color family from an image's average color

This function extracts RGB values from the average color of an image and deducts the color family it pertains.

```
public static function prevailingColorFamily(c:int){//, p_r:int, p_g:int, p_b:int):int {
	var iColor:int = -1;
	var r:int = extractRed(c);
	var g:int = extractGreen(c);
	var b:int = extractBlue(c);
	var r_g:int = Math.abs(r - g);
	var g_b:int = Math.abs(g - b);
	var r_b:int = Math.abs(r - b);

	//Red range conditionals
	if (r >= 127) {
		if (b > 86 && 212 >= b > 44 && r >= b && b > g && r_b <= 86) {
			iColor = 7;
			//pColor = "Pink";
			}
		if (g >= 44 && b >= 44 && g >= b && r_g <= 170) {
			iColor = 10;
			//pColor = "Brown";
			}
		if (g <= 44 && b <= 44) {
			iColor = 3;
			//pColor = "Red";
			}
		if (r >= 212 && g >= 212 && b <= 44) {
			iColor = 8;
			//pColor = "Yellow";
			}
		if (r >= 170 && 212 >= g && g >= 86 && b <= 86) {
			iColor = 6;
			//pColor = "Orange";
			}
		if (r <= 212 && g >= 44 && b <= 170 && r <= g && g >= b) {
			iColor = 4;
			//pColor = "Green";
			}
		}
	//Grey range conditionals
	if (r >= 44 && r == g && r == b || r >= 44 && r_g <= 20 && g_b <= 20 && r_b <= 20) {
		iColor = 2;
		//pColor = "Grey";
		}
	//Black range conditionals
	if (r <= 44 && g <= 44 && b <= 44 ) {
		iColor = 0;
		//pColor = "Black";
		}
	//White range conditionals
	if (r >= 212 && g >= 212 && b >= 212 || r + g + b >= 636 ) {//r_g + r_b + g_b < 60) {
		iColor = 1;
		//pColor = "White";
		}
	//Blue range conditionals
	if (b >= 44) {
		if (r <= 44 && g <= 212 && g <= b) {
			iColor = 5;
			//pColor = "Blue";
			}
		if (r <= 212 && r >= 44 && g <= 127 && b >= g && r >= g && r_b <= 86) { //r <= b
			iColor = 9;
			//pColor = "Purple";
			}
		}
	//Else
	if (iColor == -1) {
		if (r > g && r > b) {
```

```
                iColor = 3;
                //pColor = "Red";
                }
            if (g > r && g > b) {
                iColor = 4;
                //pColor = "Green";
                }
            if (b > r && b > g) {
                iColor = 5;
                //pColor = "Blue";
                }
            }
        return iColor;
        }
```

Function to get a featured color of the image different from prevailing color if possible

This function gets the first color of the array of unique colors of the image which family is different from the one of the prevailing color at the image, but also to black and grey tones. Even if at this first version not always is possible to get an enough different color, in the majority of cases this is enough for visualizing a singular couple of colors from the image that will give an overall idea of the image tones and contrast.

```
public static function getFeaturedColor(uniqueColours:Array, p_c:int):uint {
        var prevailingColorF:int = p_c;
        var featuredColourFamily:int;
        var fc:uint;
        for (var i:int = 0; i < uniqueColours.length; i++ ) {
            featuredColourFamily = prevailingColorFamily(uniqueColours[i].colour);
            trace("i="+i+" length:"+uniqueColours.length+"
+++prevailingColorF="+prevailingColorF+"+++getFeaturedColor+++"+uniqueColours[i].colour+"+++featuredColour
Family+++"+featuredColourFamily);
            if ( featuredColourFamily != p_c && featuredColourFamily != 2  && featuredColourFamily != 0) {
                fc = uniqueColours[i].colour;
                i = uniqueColours.length;
                }
            }
        return fc;
        }
```

Function for displaying the emotions and psychological meanings of color families

```
public function displayMeanings():void {
        stringUtils.writeTrace(0, "displayMeanings");
        var e_width:int = 100;
        var emotions:Array = new Array();
        var familiesCounter:int = 0;
        var s:String = "\n";
        var e_unit_previous:MovieClip = new MovieClip();

        for(var i:int=0;i<items.length;i++){
            var n:int = items[i].get_prevailingColor();
            var c:int = emotions[n];
            c = c+ 1;
            emotions[n] = c;
            }
        for (var j:int = 0; j < emotions.length; j++ ) {
            var e_unit_width:int=e_width/emotions.length;
            var e_unit:MovieClip=new MovieClip();
            e_unit.graphics.beginFill(0xFFFFFF);
            var e_color:ColorTransform = e_unit.transform.colorTransform;
            var cFC:uint = imageTools.colorFamilyCode(j);
```

```
            e_color.color = cFC;
            var h:int = 20 * emotions[j];
            if (h > 0) {
                  familiesCounter += 1;
                  var mWeight:int = 10;
                  mWeight = emotions[j] * 3;
                  if (mWeight <= 9) { mWeight = 9; }
                  if (mWeight >= 22) { mWeight = 22; }
                  var t:TextField =
stringUtils.prepareTextField(imageTools.colorFamilyNameDesc(j,false,true), -60,0, mWeight);
                  e_unit.transform.colorTransform = e_color;
                  e_unit.graphics.endFill();
                  e_unit.addChild(t);
                  meaningsDisplay.addChild(e_unit);
                  e_unit.y = e_unit_previous.y + e_unit_previous.height ;// (familiesCounter * 10) +
e_unit.height;//(familiesCounter * e_unit_width) + 10;
                  e_unit_previous = e_unit;
                  }
            }
      meaningsDisplay.t_meanings.visible = false;
      meaningsDisplay.x = 680;
      meaningsDisplay.y = 450;
      addChild(meaningsDisplay);
}
```

Function for constructing an array without repeated colors [deprecated]

This function identified unique colors and returns an array with them. For some reason I couldn't make the original version of this function, by Justin Windle's, work properly in previous at the application's beta version and ended up losing my patience (mostly with myself) and creating my own version of the conditional using `IndexOf()`. However, I returned to it later in the development and made Windle's version, better than mine, works. Thus, this function is no longer used at 1.0 version.

```
public static function uniqueColours( colours:Array, maximum:int, tolerance:Number = 0.005 ):Array{
      trace("colours.length "+colours.length );
      var unique:Array = new Array();
      for (var i:int = 0; i < colours.length && unique.length < maximum; i++){
            //trace("???"+unique.indexOf(colours[i].colour))
            if ( unique.indexOf(colours[i].colour)==-1){
                  unique.push( colours[i] );
            }
            //WINDLE'S VERSION
            /*
            if ( different( colours[i], unique, tolerance ) )
            {
                  unique.push( colours[i] );
            }
            */
      }
      for (var j:int; j < unique.length;j++ ) { trace("unique"+j+" "+unique[j].colour)}
      return unique;
}
```

Recursive set of functions to load images sequentially

This group of functions are triggered with a call to `fillImagesArray()` and are necessary for the correct functioning of the preloader that reports load progress on stage. `fillImagesArray()` is called once, prepare the load of the first item and, when finished, starts the recursive call to `runLoad()` function, which will call itself each time an item's image load is completed, until all are and calls `activateItemDisplay()` for the final stage of the process. The correspondent listener is custom and stored in class imageLoadEvent..

```
private function fillImagesArray():void {
	stringUtils.writeTrace(0, "fillImagesArray");
	items[0].set_id(0);
	items[0].loadImage();
	itemsDisplay.addChild(items[0].get_image_mc());
	items[0].addEventListener(imageLoadEvent.LOAD_COMPLETED, runLoad);
	items[0].addEventListener(IOErrorEvent.IO_ERROR, riseLoadError);
	}

private function runLoad(e:imageLoadEvent):void {
	stringUtils.writeTrace(0, "runLoad");
	preloaderCounter++;
	var v:int = Math.abs(preloaderCounter*100)/(items.length-1);
	main.inputUrl.text  = v + "% loaded, please wait...";

	var c:int = 0;
	c = e.currentTarget.get_id() + 1;
	if (c <= items.length - 2) {
		items[c].set_id(c);
		items[c].loadImage();
		itemsDisplay.addChild(items[c].get_image_mc());
		items[c].addEventListener(imageLoadEvent.LOAD_COMPLETED, runLoad);
		}
	if (c == items.length - 1) {
		items[c].set_id(c);
		items[c].loadImage();
		itemsDisplay.addChild(items[c].get_image_mc());
		items[c].addEventListener(imageLoadEvent.LOAD_COMPLETED, activateItemsDisplay);
		}
	}

private function activateItemsDisplay(e:imageLoadEvent):void {
	stringUtils.writeTrace(0, "activateItemsDisplay");
	main.inputUrl.text = "Ready! Use buttons at the right to visualize data.";
	var n:int = items.length - 1;
	addChild(itemsDisplay);
	itemsDisplay.x += 20;
	itemsDisplay.y += 40;
	main.openDisplayButtons();
	}
```

loadBitmap function

This function loads the bitmap which URL is at the item's media:content tag resizing it if necessary. This helps to improve overall performance, because imageTools functions will have to deal with a smaller image in pixels once they are ordered to process it.

```
private function loadBitmap(e:Event):void  {
	controlBitmapSize();
	stringUtils.writeTrace(0, "loadBitmap");
	e.target.removeEventListener(Event.COMPLETE, loadBitmap);
	loadedBitmap = new Bitmap(e.target.loader.contentLoaderInfo.content.bitmapData.clone());

	var scale:Number = 1;
	var b_width:Number = loadedBitmap.bitmapData.width;
	var b_height:Number = loadedBitmap.bitmapData.height

	if (b_width >= b_height + 1 ) {
		scale = 1/(b_width / 240)
		}
	if (b_width <= b_height ) {
		scale = 1/(b_height / 160);
		}
	if (b_width == 240 || b_height == 240) {
		scale = 1;
		}

	var matrix:Matrix = new Matrix();
	matrix.scale(scale, scale);
	b_width = loadedBitmap.bitmapData.width * scale;
```

```
b_height = loadedBitmap.bitmapData.height * scale;
var smallBMD:BitmapData = new BitmapData(b_width, b_height, true, 0x000000);
smallBMD.draw(loadedBitmap, matrix, null, null, null, true);
var scaledBitmap:Bitmap = new Bitmap(smallBMD, PixelSnapping.NEVER, true);
loadMC(scaledBitmap);
}
```

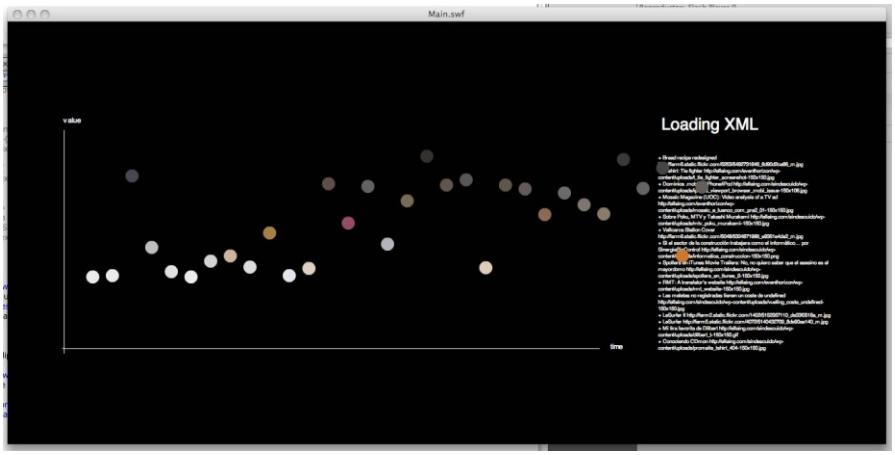
RGBtoHSV function from Kirupa user m90

The function was published at Kirupas forum by m90 as answer to the request of user IQAndreas for this function in AS3. See Linkography for URL.

```
public static function RGBtoHSV(R:Number,G:Number,B:Number):Object {
      var HSV:Object = {};
      var min:Number=R;
      if (G<min) {min=G;}
      if (B<min) {min=B;}
      var max:Number=R;
      if (G>max) {max=G;}
      if (B>max) {max=B;}
      HSV.V = max;
      var deltaColor:Number = max - min;
      if (max!=0) {
            HSV.S=deltaColor/max;
            if (R==max) {
                  HSV.H = ( G - B ) / deltaColor; // between yellow & magenta
               } else if ( G == max ) {
                   HSV.H = 2 + ( B - R ) / deltaColor; // between yellow & magenta
               } else {
                   HSV.H = 4 + ( R - G ) / deltaColor; // between magenta & cyan
               }
               HSV.H*=60;
               if (HSV.H <= 0) {
                   HSV.H+=360;
               }
           } else {
               HSV.S = 0;
               HSV.H = 0;
               HSV.V = 0;
           }
           return HSV;
       }
```

APPENDIX 2
SCREENSHOTS

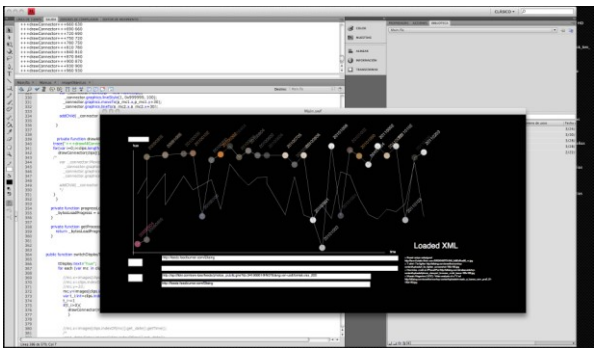
First load

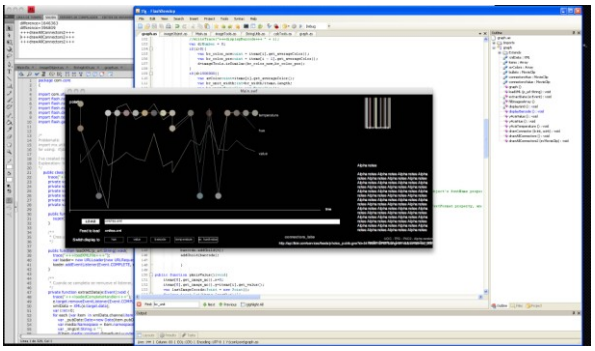


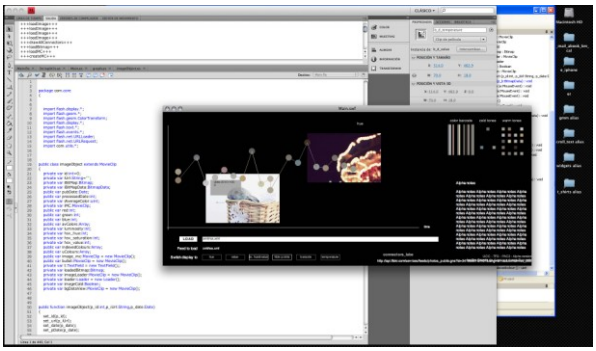
Alpha version

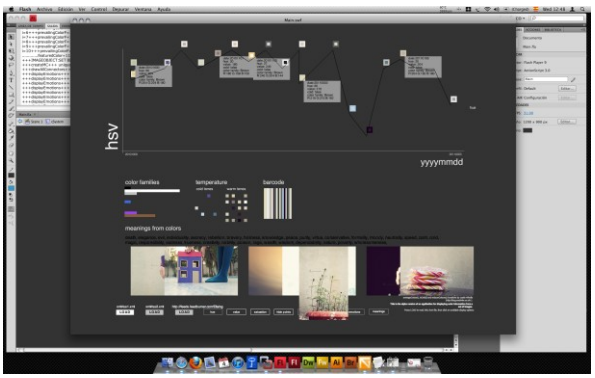
Alpha version loading a Flickr feed online (my own photostream):

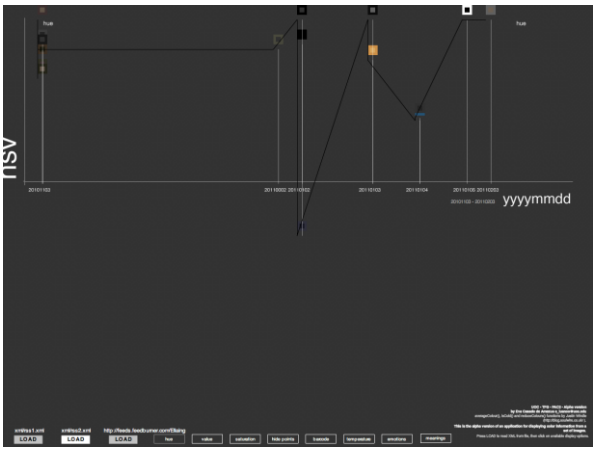
Development process for beta





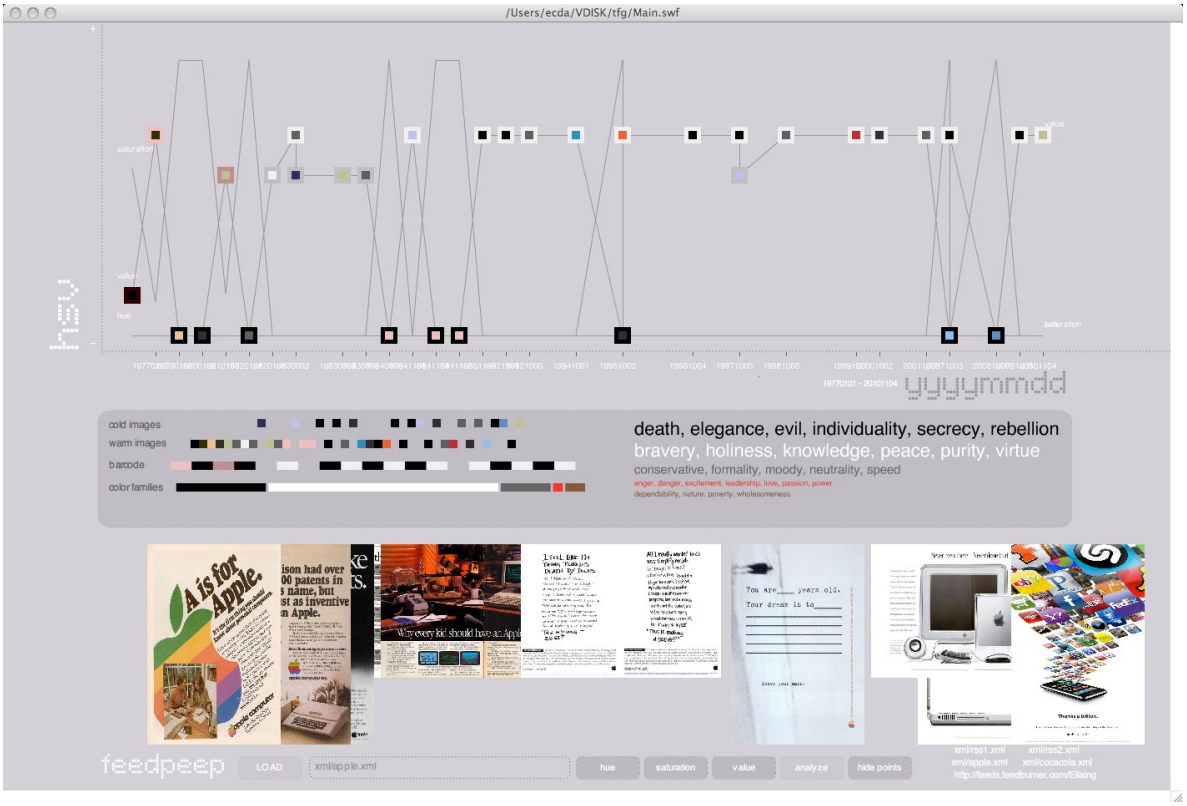






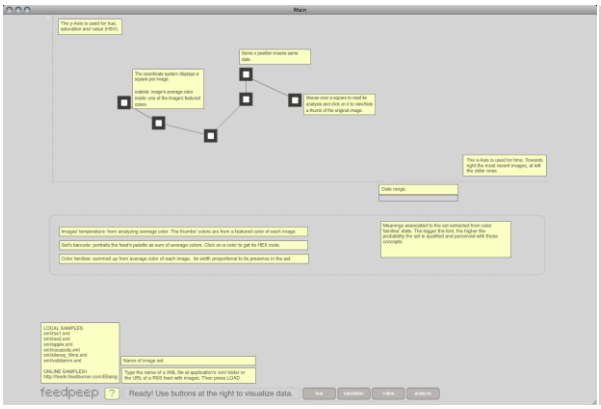
Beta version

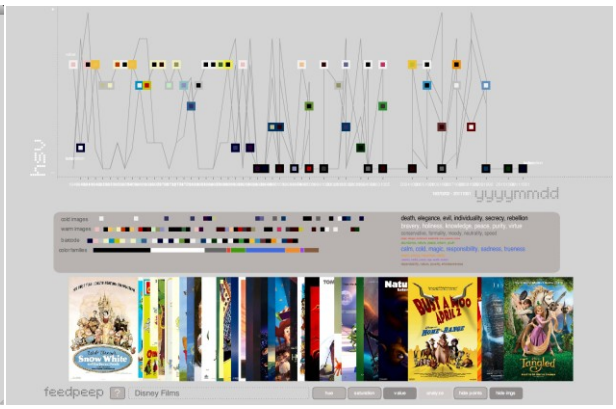
Beta version loading a brand feed stored locally (printed advertisements of Apple):



Beta version in full-screen mode loading Reem's Flickr feed (online version):

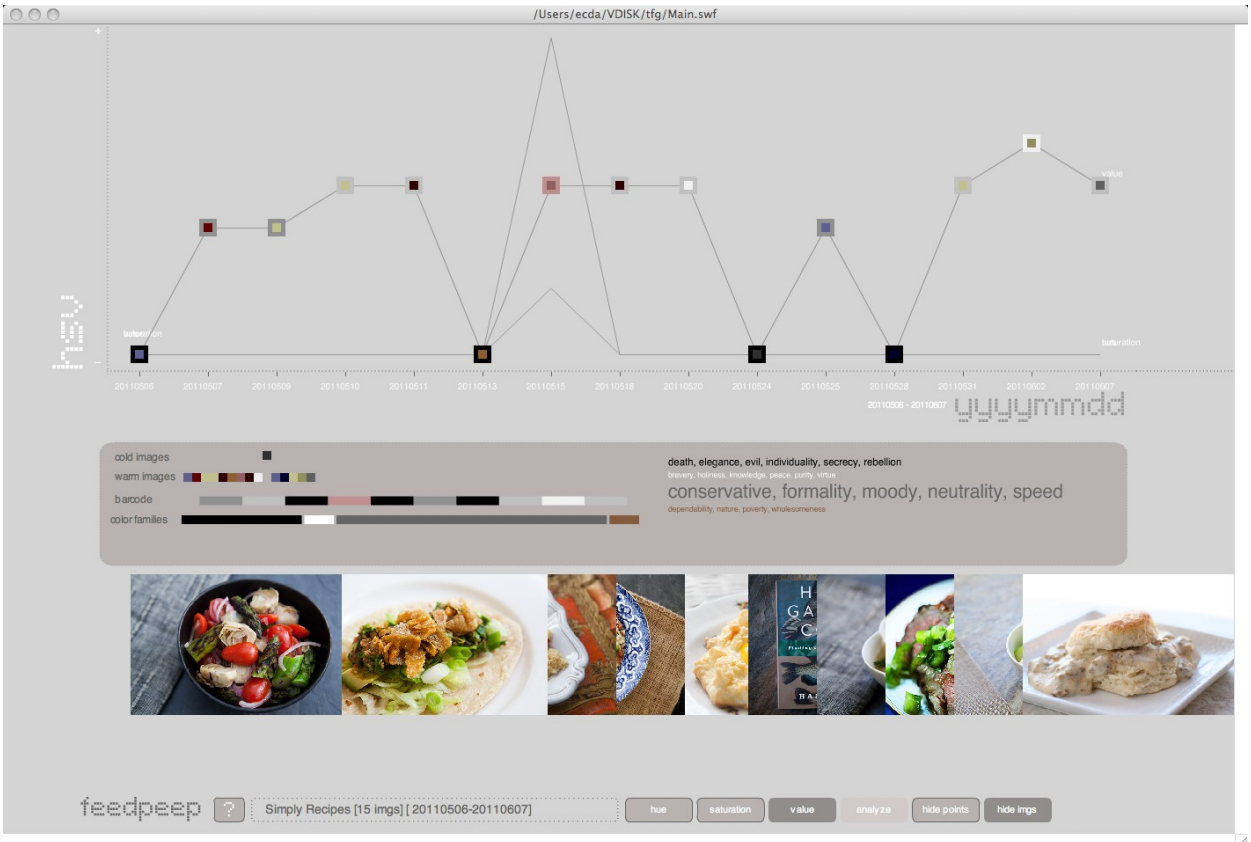
Evolution process for v1.0

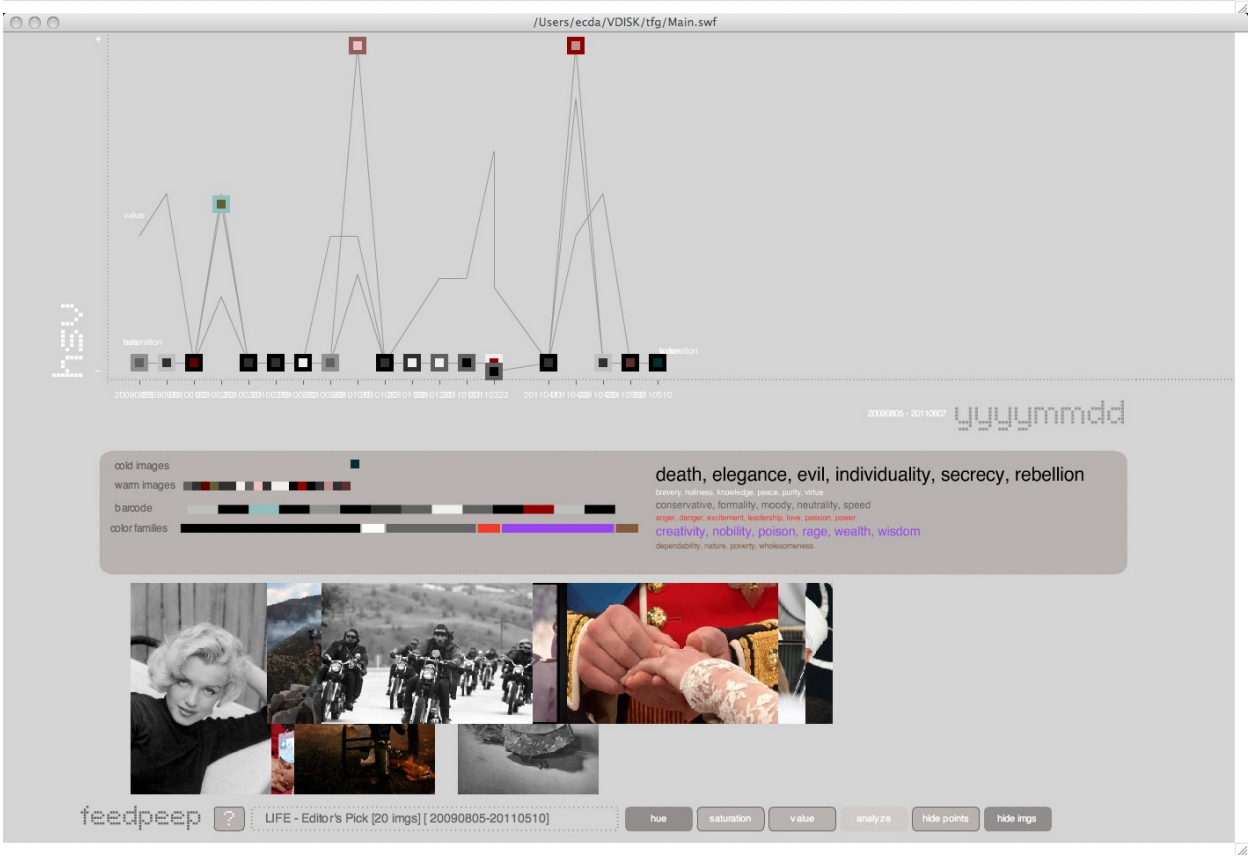


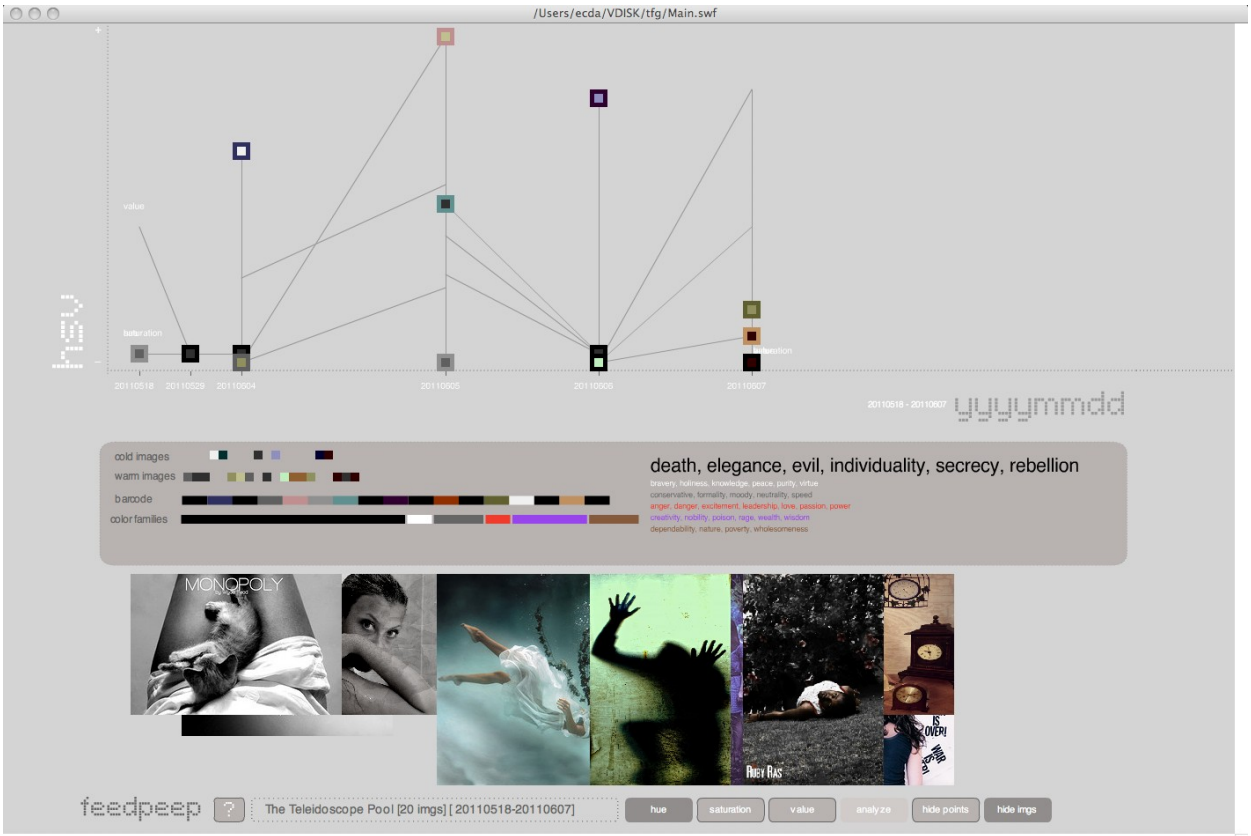


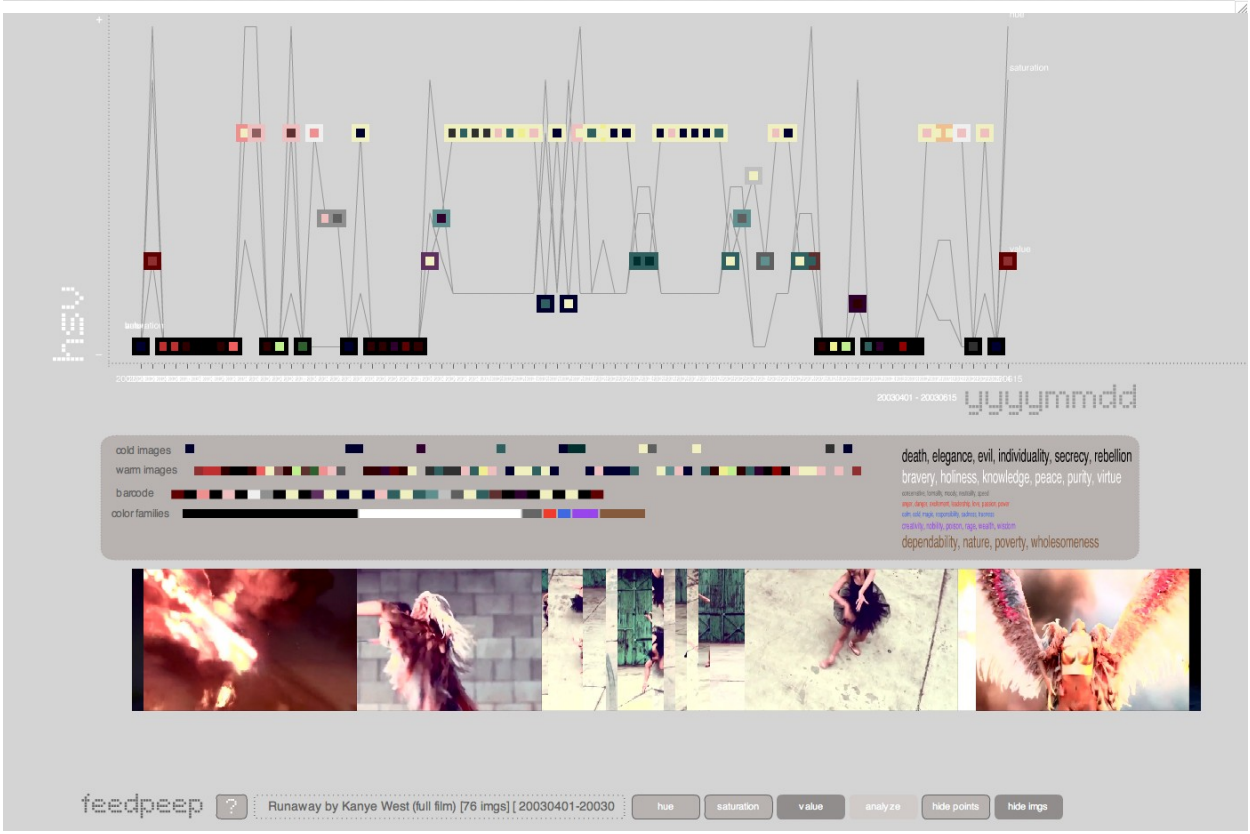
Version 1.0

Load of three online feeds and one locally stored (online feeds' URLs available at Linkography chapter):



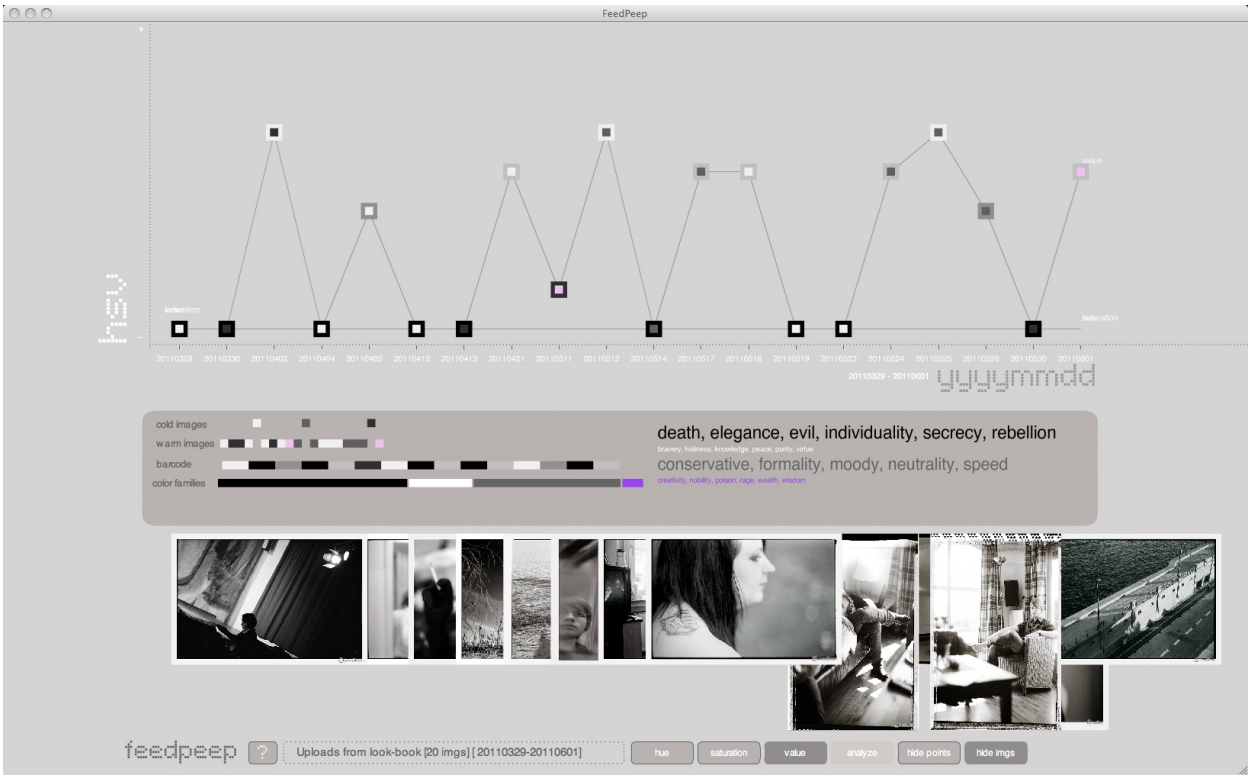



/Users/ecda/VDISK/tfg/Main.swf
value
saturation
20110518
20110529
20110604
20110605
20110606
20110607
20110518 - 20110607
yyyymmdd
cold images
warm images
barcode
color families
death, elegance, evil, individuality, secrecy, rebellion
bravery, holiness, knowledge, peace, purity, virtue
conservative, formality, moody, neutrality, speed
anger, danger, excitement, leadership, love, passion, power
creativity, nobility, poison, rage, wealth, wisdom
dependability, nature, poverty, wholesomeness
MONOPOLY
feedpeep
The Teleidoscope Pool [20 imgs] [20110518-20110607]
hue
saturation
value
analyze
hide points
hide imgs


saturation
20030401 - 20030615
yyyymmdd
cold images
warm images
barcode
color families
death, elegance, evil, individuality, secrecy, rebellion
bravery, holiness, knowledge, peace, purity, virtue
dependability, nature, poverty, wholesomeness
feedpeep
Runaway by Kanye West (full film) [76 imgs] [20030401-20030
hue
saturation
value
analyze
hide points
hide imgs

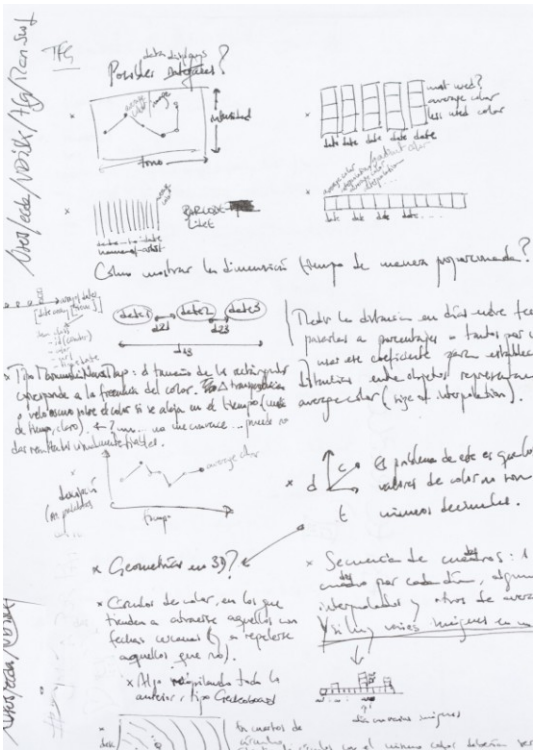
Version 1.0 [AIR]

Load of an online feeds with AIR version (URL available at Linkography chapter):

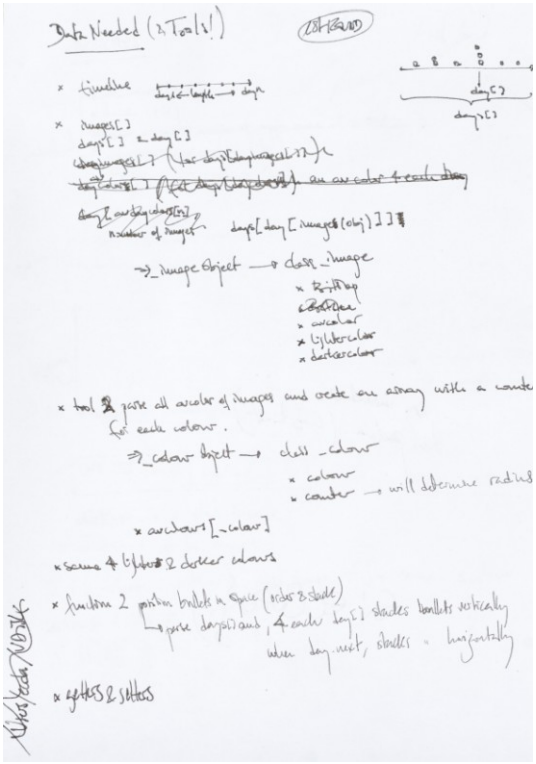


APPENDIX 3
NOTES AND DOODLES THROUGH DEVELOPMENT PROCESS

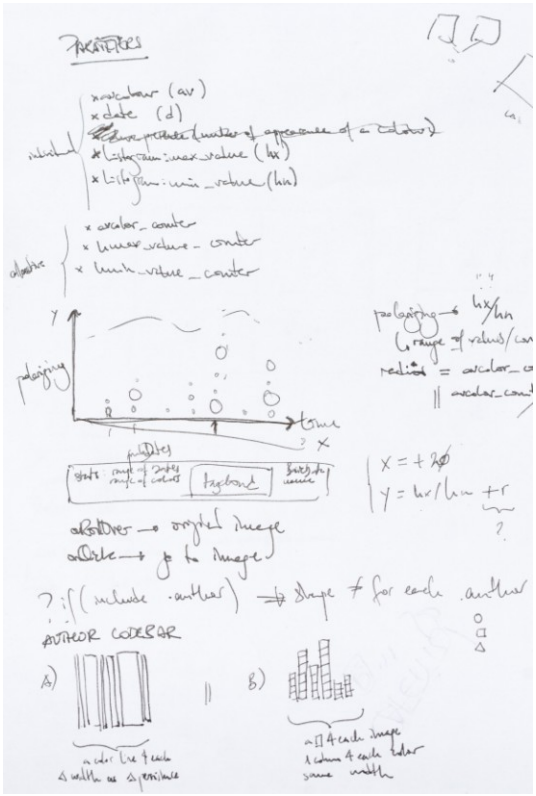
First annotations previous to development start:



Random notes on required basic data from image:



Notes on parameters for chosen display modes:



Testing on `prevailingColorFamily()`function:

APPENDIX 4 LINKOGRAPHY & BIBLIOGRAPHY

Data resources of local XML and JPEG sets

Disney theatrical animated features
http://en.wikipedia.org/wiki/List_of_Disney_theatrical_animated_features

Evolution of Apple ads
http://www.webdesignerdepot.com/2009/09/the-evolution-of-apple-ads/

Evolution of Coca-Cola ads
http://www.beautifullife.info/advertisment/history-of-coca-cola-in-ads/

Kanye West
http://kanyewest.com/
http://itunes.apple.com/es/album/my-beautiful-dark-twisted/id404005350

Reem's Flickr photostream
http://www.flickr.com/photos/reem_unique/

She-Noir's Flickr photostream
http://www.flickr.com/photos/19547713@N05/

Linkography on color

Color Meanings by Culture
http://www.globalization-group.com/edge/resources/color-meanings-by-culture/

Color Theory For Designers, Part 1: The Meaning of Color
http://www.smashingmagazine.com/2010/01/28/color-theory-for-designers-part-1-the-meaning-of-color/

Color Theory For Designers, Part 2: Understanding Concepts And Terminology
http://www.smashingmagazine.com/2010/02/02/color-theory-for-designers-part-2-understanding-concepts-and-terminology/

Colours in Culture
http://www.informationisbeautiful.net/visualizations/colours-in-cultures/

Colour Lovers
http://www.colourlovers.com/blog/2010/03/28/eclectic-color-roundup-colander-time-as-color-iphone-apps-pink-terror-slowmo-video

Color Theory
http://dev.opera.com/articles/view/8-colour-theory/

Relationship between color and emotion: a study of college students
http://findarticles.com/p/articles/mi_m0FCR/is_3_38/ai_n6249223/?tag=content;col1

Linkography on interfaces & development

Colorendal
http://app.birnimal.net/en/iphoneapp/colorendar

Culture Maps: VisualSense
http://culturemaps.net/softwares/visualsense/

Flare
http://flare.prefuse.org/

Geckoboard
http://www.geckoboard.com/

RGBtoHSV function (from Kirupa user m90)
http://www.kirupa.com/forum/showthread.php?t=322958

Journalism in the Age of Data
http://datajournalism.stanford.edu/

Justin Windle
http://blog.soulwire.co.uk/

Multicolr Search Lab
http://labs.ideeinc.com/multicolr/

Shokunin Kishitsu, the craftman's spirit
http://37signals.com/svn/posts/2115-shokunin-kishitsu-the-craftmans-spirit

Visual evidence: Movies are getting worse
http://moki.tv/blog/visual-evidence-movies-are-getting-worse

Voyage RSS reader
http://rssvoyage.com/

Linkography on date issues in AS3

The Programming Chronicles: ActionScript's arithmetic error in dates (in Spanish)
http://theprogrammingchronicles.com/2011/04/23/fallo-aritmetica-fechas-actionscript/

Issues on dates at AS3Syndication Library (that library, Flex based, has not been used in the development of FeedPeep but the discussion on date issues is a good information source)
https://github.com/mikechambers/as3corelib/issues/163
http://code.google.com/p/as3syndicationlib/issues/detail?id=13#c2
http://code.google.com/p/as3syndicationlib/issues/detail?id=8#c16
http://code.google.com/p/as3syndicationlib/issues/detail?id=15#c1

Bibliography

Lupton, Ellen and Cole Phillips, Jennifer; Graphic Design: The New Basics. Princeton Architectural Press, New York, July 1st, 2008.
http://www.amazon.co.uk/Graphic-Design-Basics-Ellen-Lupton/dp/1568987021/ref=sr_1_1?ie=UTF8&s=books&qid=1302721794&sr=8-1

Manovich, Lev; What is Visualization? Manovich.net, October 25th, 2010.
http://manovich.net/2010/10/25/new-article-what-is-visualization/