High-Performance Framework For Web Applications

Frank van Gemeren

December 18, 2009

Abstract

This paper investigates the possibilities of a web page resembling an end-user's operating system's graphical user interface and the handling of multiple types of data in a front-end focused high-performance environment to get as close as possible to mimicking an operating system for handling data. The abstractions and modifiability of the proposed solution makes it easily customizable for any kind of web application, with a focus on Content Management Systems.

1 Introduction

One of the most useful characteristics of humans is our capability to remember and recognize. We base all of our actions on data that was gathered in the past. Past experiences, expectations and beliefs affect the way people process information. Studies have shown that recognition is even more important than remembering when it comes to deriving new knowledge from an external source like our surroundings[1].

Many end-users who use a computer are familiar with its user interface by gradually learning the ins and outs of the interactions and the results. Most CMSes have many options in many categories and it can be hard to navigate a new GUI for the first time because the designers break the chain of previous knowledge about actions by removing, modifying or adding behaviours. The options of the CMSes are usually categorized in lists, which may, or may not, be consistent with earlier knowledge from the user's Operating System (OS). This paper creates a general-purpose framework called "Frag-min" that is focused on extendability, modifiability and usability to tackle the problem of relearning the GUIs of new CMSes.

Nobody likes to wait on a website to load and with the rise of cloud-computing[3] services like Amazon's EC2/S3 and Google Apps, which are aimed at the business and corporate level, the amount of time people will be interacting with online data, be it for work or for private use, will most likely increase a lot.

Frag-min is meant as a general purpose framework, but because of its appearance it's well-fitted for content management systems in websites (CMS), customer relationship management (CRM) and many more tasks where people need to work directly with potentially lots of online data. This paper will put Frag-min in a role as CMS.

The main goal of Frag-min is to redesign the human-computer interaction to speed up the workflow. This is done by providing a new paradigm that is based on transformations and an intuitive drag-'n-drop interface. Most of the actions that an end-user can do on his or her computer will be possible in Frag-min.

2 Design

This chapter describes the general direction of Frag-min and flaws of other systems.

2.1 User Analysis

Frag-min focuses on powerful capabilities for its users, the module-developers, like modifiability and rapid prototyping by supplying easily extendable classes. This enables them to focus on the technical side of their customers' wishes, while Frag-min takes care of easy interactions between the system and the end-users.

2.1.1 Developers

Developers are professionals in Information Technology and they have knowledge about (web) programming languages, algorithms and data structures. They are proficient in creating standards-compliant xHTML, CSS, and JavaScript and back-end technologies like database interaction and manipulating the server's file system with scripts. They may come from a background in one of the system programming languages like C++ or Java, but they have the experience to make the switch to web-based languages quickly.

Developers expect a proper framework with a clear and easy to follow Application Programming Interface. Documentation and comments in the code

of the API needs to be as concise as possible. They expect a framework to be extendable with common Object Oriented Programming practices support, like information hiding and inheritance.

2.1.2 End-users

End-users are all the people who have to work with the created modules on a regular basis. The kind of work depends on the module and the job description of the end-user, but generally the end-user will not be trained in the development of web applications nor software engineering. Jobs like news-poster, clerk or e-marketeer are well-suited for the use of Frag-min.

End-users will think of Frag-min of being like a real program, running locally on his or her PC. They expect Frag-min to respond without much latency and expect it to have visuals that resemble the rest of the Operating System. They expect most, if not all, possible actions from a normal OS to be possible in Frag-min, like creating folders, renaming files and creating new items in modules.

2.2 JSON

JSON[13] (JavaScript Object Notation) is a subset of the JavaScript language and is a way of transforming variables into string representations. This resulting string can be parsed by the JavaScript interpreter to recreate the former object. The JavaScript parser is native to all browsers so JSON is a very fast and elegant way to transmit data from the server to the browser.

The following table shows some metrics done on a page that consists of a certain amount of list-items with an image and describing text. The following abbreviations are used: Parse is the time needed to parse HTML or add elements via the DOM, IPLT is Initial Page Load Time (when the "loaded" event is fired indicating that the normal DOM is ready for use) and TLT is Total Load Time. All the times are in seconds.

Amount	HTML-only		HTML + JSON		
	Parse	TLT	IPLT	Parse	TLT
100	0.11	1.37	0.42	0.20	1.42
200	0.23	2.29	0.43	0.40	3.40
500	0.83	15.53	0.43	0.83	5.25
1000	3.08	32.85	0.46	2.14	25.46
5000	crash	crash	0.43	12.61	45.80

During the JSON tests, an additional call to the Prototype-library was made, which took 0.18 seconds, and added 136KB of data (unzipped). The total download size of the extra JSON was less than the extra data of the pre-added HTML. This is because the amount of characters of the pre-added HTML is higher than the delimiters between the values in the JSON reply. Also, the HTML in the JavaScript that renders the JSON is only listed once, while this is added to each item in the non-JSON page.

The equation to find out whether to use pure HTML or JSON with respect to file size is:

$$n * (HTML + item) = lib + cJS + JSON$$

n is the amount of items, $HTML$ is the added HTML per item, $item$ is the size of one item, lib is the size of the JavaScript library, cJS is the size of the custom JavaScript needed to do the AJAX call and render the JSON to HTML and $JSON$ is the size of the JSON reply itself.

For small amounts of items, the pure HTML takes less space, but as the amount of added HTML characters gets larger, JSON's lower overhead is preferable. Libraries and the needed custom JavaScript for the parsing can be optimized and cached, which saves space too.

When looking at speed, parsing the pure HTML is faster than generating DOM-elements from the JSON for small amounts. The reason why this stops at larger amounts and even crashes is unclear and needs more research that is specifically aimed at JSON and the DOM in general. An added benefit of JSON is that the page is responsive earlier, because the "loaded" event has already fired.

Because Frag-min is meant to show much data, JSON is used whenever possible. The libraries and custom JavaScript code have been amortized and are neglectable compared to the size and time of downloads of non-text items, such as images.

2.3 Requirements

Because of the general-purpose goal, the framework has to work cross-browser and cross-platform. This maximizes the amount of potential end-users and requires the use of some libraries to work around browsers' bugs.

Loading speed of the initial page can have a great effect on the behaviour of users. Results from Google, Yahoo!, Microsoft and others have shown that the faster the site loads, the more searches, visitors, time spent on-site, or revenue[6][7][8]. For a web application like Frag-min it's about getting the

end-users to use Frag-min. The end-users shouldn't think using Frag-min is a burden or barrier, but as easy, fast and responsive.

Visual feedback is of the utmost importance when it comes to providing a smooth user experience. Visitors will notice a delay of more than 100 milliseconds as "not immediately responsive"[9]. Frag-min tries to provide a smooth experience by showing graphical effects like animations to keep the end-user focused, while giving the system a few more milliseconds to do its task.

2.4 Elements

Frag-min aims to mimic the user interface of a desktop operating system. The main graphical parts of a desktop are icons and windows. Windows can be split up in several subsystems. A description of some of the important parts that Frag-min uses is detailed below.

2.4.1 The Canvas

Everything is drawn on the Canvas. The Canvas is the most low-level visible element on the screen. It can be compared to an empty desktop. The Canvas may or may not have a TaskBar.

The Canvas should handle most of the high-level window-interactions, like closing windows and detecting opening of modules by clicking on icons, because it has an overview of all the windows and icons.

2.4.2 The GUIWindow

This window is a standard window. It has no content and no special menu bar. It supports multi-layered windows with a depth-based hierarchy — windows can be on top of each other — just like a normal OS. The end-user can resize the window to accommodate for an increasing or decreasing amount of data in the window. Moving the location of the window is also possible. This makes it easy for the end-user to change the desktop environment to his or her liking, which is good for the workflow.

2.4.3 Regions

GUIWindows consist of a MenuBar and a Region. All the actual content of the module or folder is listed in the Region. The standard Region will display a list of icons, but it is possible for module-developers to override this. Regions should be able to use various templates. For example, a simple text-module might list the author of the text in a meta-data field, but a photo-module

might list a preview of the photo along with more advanced EXIF metadata. The templates can be used to provide the end-user with more, or less, information, depending on his or her wishes. The extra functionality will need to be provided by the module-developer.

2.4.4 Icons

Icons give the end-user a hint: is the item a module, a folder, a text-file or another kind of file? Icons need a good iconography so it's clear what the item is. Icons have behaviours assigned to them to indicated the various actions that are possible. An icon should light up when the end-user hovers over it to provide feedback that it's possible to do an open-action.

2.4.5 Selectables

Interactions are only possible on certain elements, like icons and windows. These elements will need to be stored somewhere so their onFocus and other methods are callable. The Selectables object will need to keep track of every object that is able to interact with the user.

2.5 CMS Flaws

Interaction with the end-user is really important for Content Management Systems, or CMSes in short. With the advance of Web 2.0 technologies like AJAX and fancy design like reflections, drop shadows and round corner boxes, the CMSes have tried to implement these to be up-to-date with the user's expectations. However, when an application wants to use a lot of new interactions, some icons or "hover-states" can be ambiguous or confusing for the user. Other design decisions in menu-structure can even be annoying. This section discusses some of the issues in Joomla! and Drupal, two free open-source CMSes that are widely used by consumers and businesses to power their websites. Metrics of their user base are covered later.

2.5.1 Joomla!

One of the negative things in Joomla! is that the Save/Apply buttons are at the top of the screen. When creating new pages or editing pages where you only want to change something at the bottom, it can be annoying. Having two Save spots, one at the top and one at the bottom will be better with the visual flow. This goes for non-article pages, like banners, too.

The main menus are not responding while an end-user is editing an article: he or she first needs to Save or Close the page, before it returns to the Article

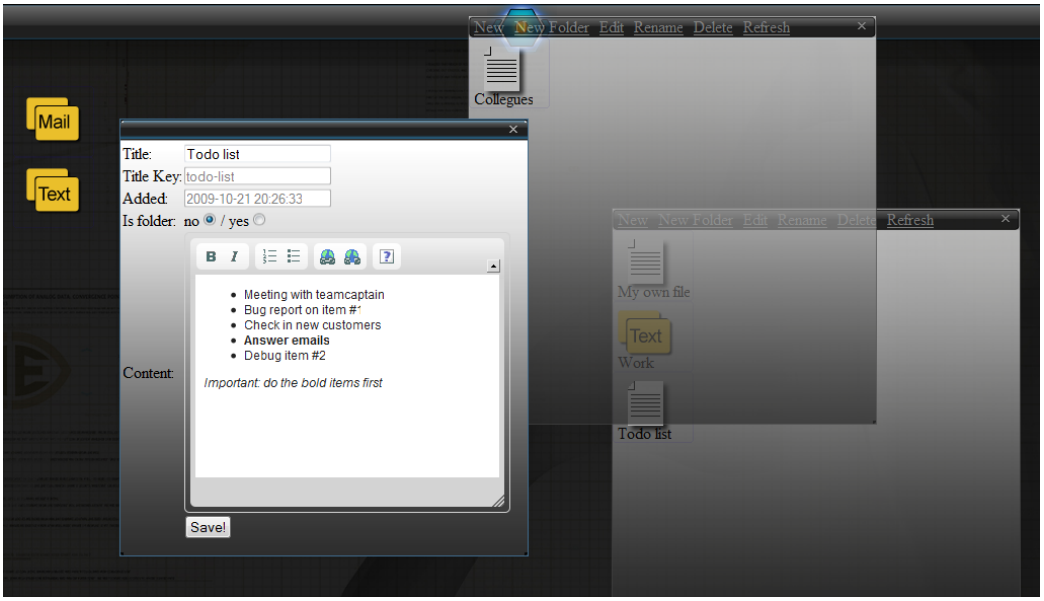

Figure 1: Frag-min: a multi-level structured GUI

Manager where the menus are working again. Changing order is done by clicking an up or down arrow. A missed opportunity for a nice drag-'n'-drop interface. Besides that most of the interfaces are well-designed.

2.5.2 Drupal

Drupal's standard skin doesn't use descriptive icons and sometimes shows too much information at a time. This is especially visible in the main action screen, where all the possible actions are listed. Actions are well-named, but giving a whole subsection of 'Clean URLs' to only a radio-button is a missed opportunity. For instance, it could be named 'SEO/Usability' and then give access to more advanced options like minifying scripts/CSS and using external static-file servers to improve the front-end experience. If an end-user is editing a page and wants to edit another page, he or she has to traverse the fully collapsed tree-menu again, which costs time and is annoying.

3 Implementation

This section covers the various techniques that Frag-min uses. It details the server side and the client side. The client side is run entirely in the end-user's browser.

3.1 Server Side

Frag-min uses PHP and MySQL on the server-side. The PHP implementation uses the Model-View-Controller design pattern, implemented by the Zend Framework. This makes creating new modules and extending or modifying existing modules easy for developers who have prior knowledge of the workings of Zend Framework or MVC-patterns in general.

On the MySQL side, attention was paid to make the queries and the database design optimized for speed. This meant choosing the best fitting storage engines for each module. The InnoDB storage engine was mostly used because of its good properties like foreign keys and row-based locking. For some modules other engines will perform better, like the ARCHIVE engine for logs or statistical analysis and MEMORY for fast session storage.

Important queries can be made into transactions and/or procedures. The PHP scripts will use Zend's prepared statements to ensure data integrity to enhance security.

Most of the results will be transformed at the server to JSON replies. The client handles the rendering of the results in the browser's DOM so that it's visible to the end-user.

3.2 Client Overview

Frag-min is made in xHTML 1.0 Strict, a subset of XML that is easily parsable in contrast to normal HTML. Frag-min uses the xHTML output of the index page as a 'base'. Most, if not all, other requests of data will return JSON or XML. This makes sure it's fast, because the output only has the necessary data in a format with a very low amount of overhead. This decreased file size makes it faster to download than ordinary xHTML. Depending on the estimated size of the result, the module-developer can return pure xHTML, which can speed up the rendering process for large result sets. This is not common and usually the normal JSON will perform at an acceptable rate.

Frag-min uses a lot of JavaScript. The Prototype library[4] is a cross-browser, cross-platform library that abstracts the quirks of different browsers away. This makes developing faster and easier. For the graphical effects the extension called Scriptaculous[5] is used.

To ensure fast loading of requested data from the modules, Frag-min can use lazy-loading[10] to pre-fetch needed data. The data is then processed and the appropriate elements are dynamically generated on the site.

3.3 Client Core Elements

In this section the basic core elements on which Frag-min is built are discussed.

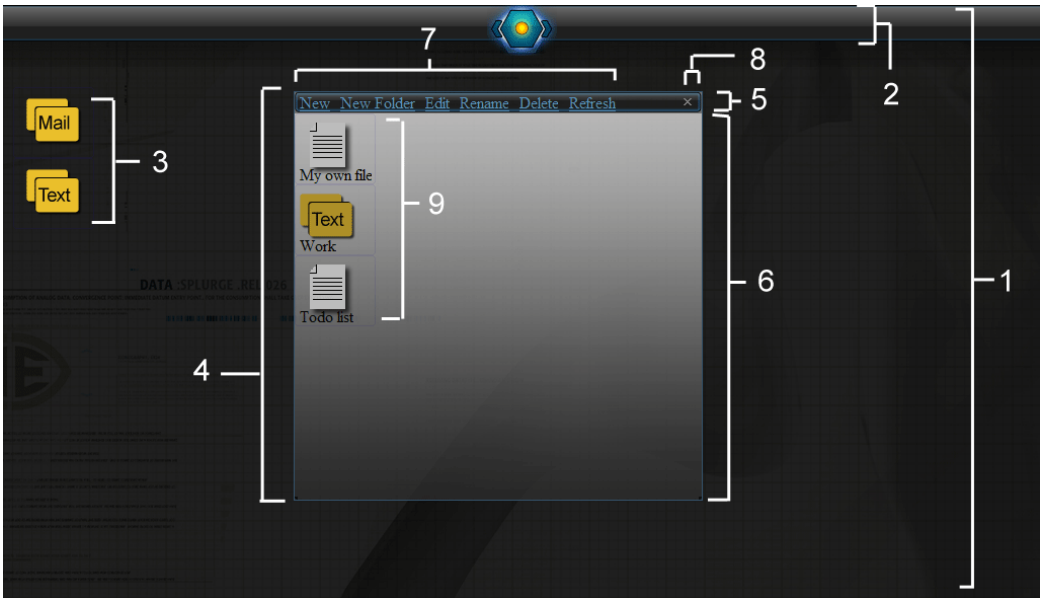

Figure 2: Frag-min elements

- 1: Canvas
- 2: TaskBar
- 3: Module Icons (Selectable, see Chapter 3.7)
- 4: GUIWindow (Selectable)
- 5: MenuBar
- 6: Region
- 7: TaskBarItems
- 8: TaskBarIcons
- 9: Module Items (Selectable)

3.4 The Canvas

During loading of the initial page, the server-side script scans a folder with modules. These modules are the content-containers. Each folder is placed on the Canvas and their initialization scripts are run, if any. These scripts can overrule standard elements and behaviors, like icons and event handlers.

There are two text-editors in the top right and bottom right part of the screen. These are normally faded, but after double-clicking them, they become active and focused: the user can start typing his or her document immediately. When the user is done, the only thing that needs to be done is to open the module where it needs to go, and then the user can use the drag-'n-drop interface to drag the text to the module. The module tries to apply a transformation (see Chapter 6.1) named canvas_to_modulename and then either accepts or rejects the inputted data. This makes inputting new blog posts, comments or HTML pages a very fast process because it bypasses a few clicks on module icons. Raw data like technical specifications can be inputted in the plain-text area and can for example be transformed to a CSV file before doing the final transformation to a custom technical module.

3.5 The GUIWindow

Each GUIWindow, or it's derivative classes, is made with xHTML's DIV-elements. After the Canvas has noticed that a new window should be opened, the GUIWindow's open-method spawns a new window with a transition effect. This eye-candy is meant for a smooth end-user experience, while the program gets a few milliseconds extra to set its content. The window can use multiple templates to distinguish between various window layouts in its Region.

After the Canvas has signaled that a window needs to be closed, its overridden close-method will stop any periodical AJAX/JSON calls. This prevents the scripts to update something that isn't there anymore.

For the hierarchical list of windows to work, they have to be attached to the same parent in the DOM. For this reason, each window is attached to the Canvas's DIV-element. Now the z-index can be automatically changed to get the desired effect of stacked windows that can be pulled to the 'front,' just like in an OS.

Resizing the GUIWindow is done using a third-party extension to Scriptaculous called "Resizables"[19][20].

3.6 Regions

A region is the main content area of a GUIWindow and is implemented by a DIV-element. The module-developer is completely free in setting it up, so normal text or a form are also possible. This can be useful for creating configuration windows that don't have an icon-based set-up.

In general, the main content will consist of a list of icons. Regions can switch their appearance to allow for tabs or extra columns or rows to display more information. Depending on the icons that are shown, it's possible to make a button that sorts the items by price, date, author, views or basically anything that's being stored in the database. This makes a fully developed module with columns for sorting and meta data very easy to use.

3.7 Selectables

In order to keep track of which things can be selected and, more important, automatically deselected, Frag-min uses a Selectables-object in JavaScript. This object has a look-up table which can match an HTML DOM-element to an object. For example, a certain DIV-element is known to be a part of a news-window object.

Selectables like icons and windows can behave differently under various circumstances. Because of this, each object has a getFocus and loseFocus method so each behaviour can be tailored.

4 Data flow and classes

Figures 3 and 4 give an overview of the internal processes of Frag-min. The items on the red background are on the server side, the items on the green background are on the client side. Any arrows passing from red to green or vice versa denote network traffic.

As with everything on the web, the program starts when a request by the end-user reaches the server. On the server-side PHP[21] determines the installed modules and sends the initial Canvas to the end-user's browser. The Canvas' JavaScript code is always loaded and calls to create new Icons are generated by the PHP-script. Loading of the installed modules' init-files is also done. This gives the developer an incredibly powerful tool set, because it's possible to control everything that's done on the standard Icon. For consistency on the Canvas, changing the actual Icon call is not possible, unless a developer overrides the Core packages. The behaviour of double-clicking the Icon is now assigned by default.

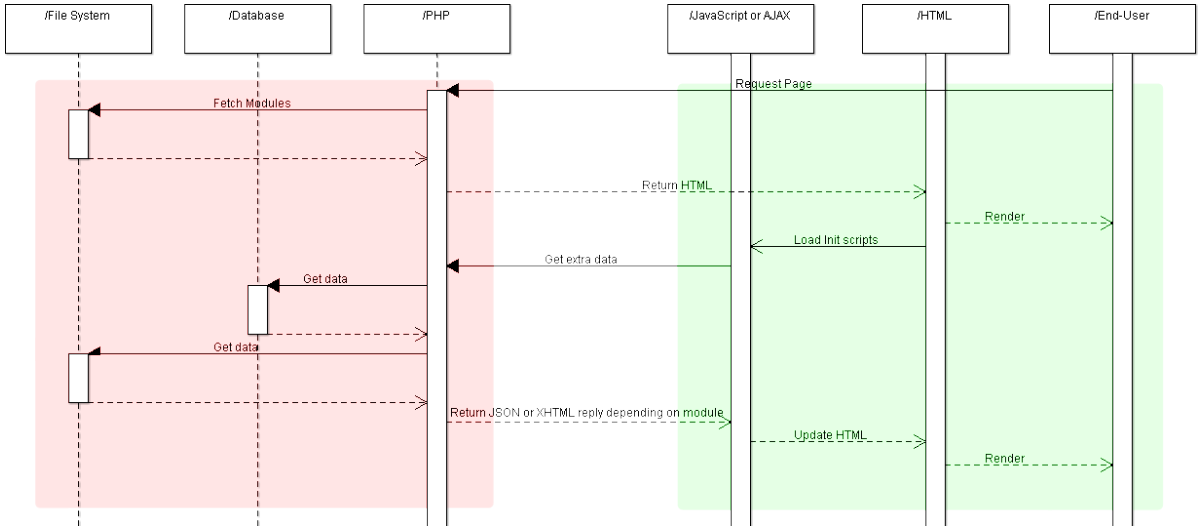

Figure 3: Loading the initial page

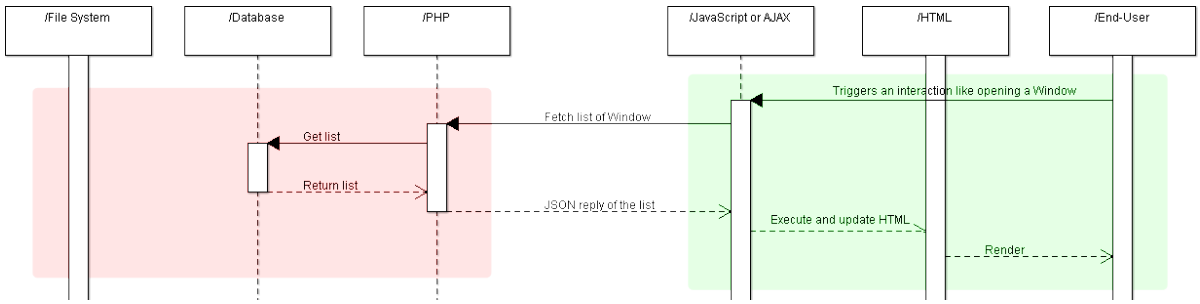

Figure 4: Activating a button

When an user double-clicks an icon, as seen in Figure 4, the code of the module is executed and this can lead to more AJAX/JSON-requests to internal or external files and more (lazy-) loading in general. The module-developer has complete control over what happens and is generally not restricted. In the case of Figure 4 the module only used the database.

Because of its modifiability, it is very easy to implement any extra behaviour or appearances. For behaviour, all that's needed is extending a base-class, as shown by the NewsWindow and FileSystemWindow classes in the class-diagram. The FileSystemWindow overrides the empty closeWindow-method of the base class because it needs a timed AJAX request. The NewsWindow uses multiple files and round-trips to the server to consume an RSS-feed and then generates a list of NewsIcons which brings the user to a window with the latest news.

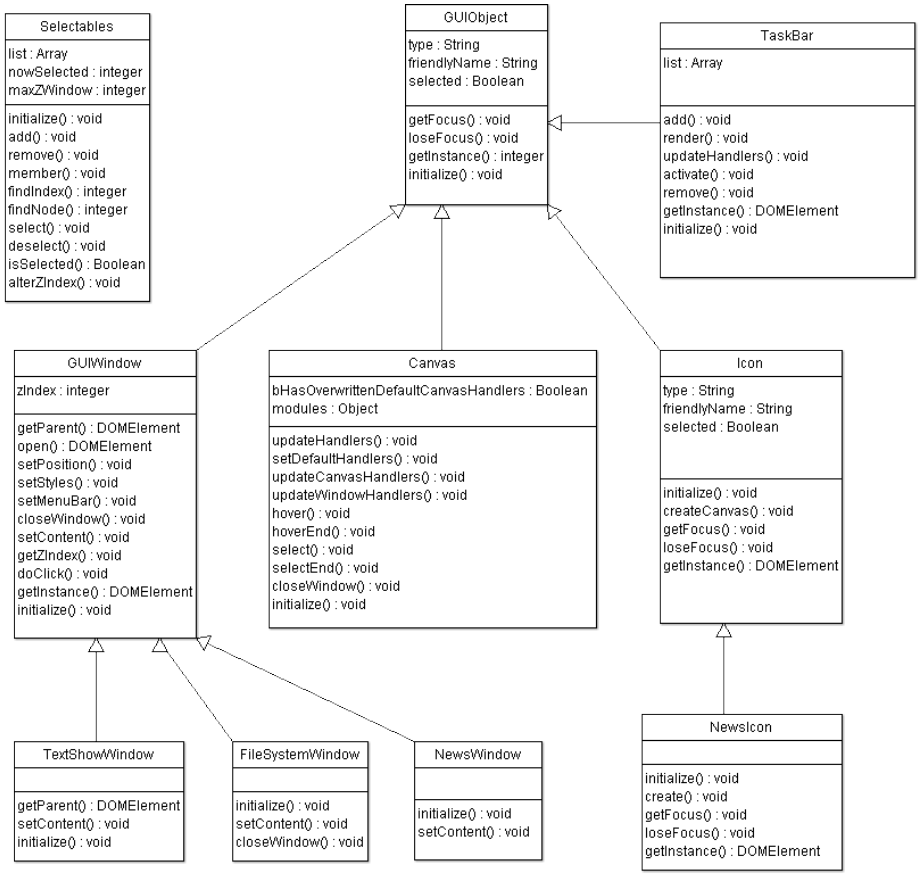

Figure 5: Example of a Class Diagram

5 Content Management Comparisons

In this section Frag-min will be compared with a number of commonly used solutions for CMSes that are available under Open Source licenses. The chosen CMSes are all targeted at consumers who want to exploit a website or blog and that have a lot of users. All CMS versions are the latest stable releases as of the time of this writing. The exact versions are listed in the list of references.

5.1 Joomla!

Joomla![2], the successor of Mambo, is the most popular CMSes for making good-looking websites[14]. Over 20% of the respondents in the aforementioned study currently uses Joomla!. According to the report, this makes it the most used CMS. It comes second in the "average weekly downloads" with almost 190 000 downloads. It supports customizable layouts, caching, user management, multi-level menus, multi-language, categories and has adapted an OS-style like interface with big icons and menus. Many advanced things are possible with Joomla!, or with one of its many plugins.

Joomla! clearly put a lot of attention into its layout. Large and descriptive icons are on all pages and collapsible sections give extra (meta) data about articles, like how many times it was visited, who the author is and more. The main edit-section has a built-in TinyMCE HTML editor.

Joomla - Administration.png

Figure 6: Joomla! Administration

Joomla - Administration2.png

Figure 7: Joomla! Article Manager

5.2 WordPress

WordPress[15] started as a blogging tool, but eventually more CMS features were added. It has support for multi-user, templating and user management and can create pages, blog posts and comments. It can be extended by a plugin system. It is easy to use, but without plugins it has limited support for multi-level menus and more advanced sites. WordPress is being used to power over 202 million different websites.

According to [14], WordPress was downloaded 433 767 times per week on average. This makes it the most downloaded CMS, but the counter used for counting was cited as not really reliable. Still the number is large and with the highest percentage on "which CMS have you used" and 13.7% of the respondents currently using it, WordPress is one of the three largest CMSes.

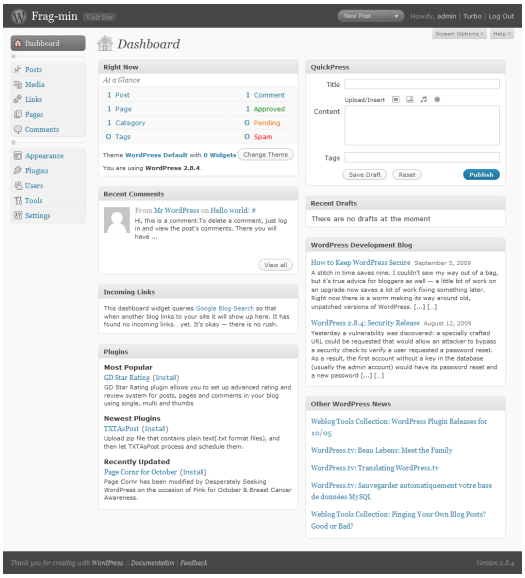

Figure 8: WordPress Dashboard

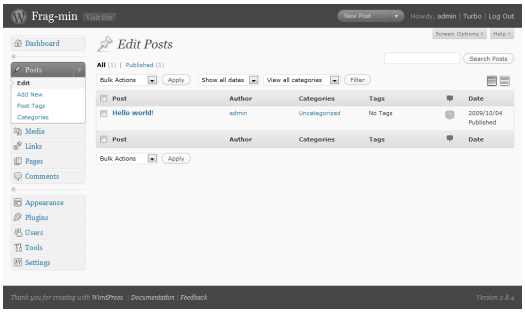

Figure 9: WordPress Post Editing

5.3 Drupal

Drupal[16] is another popular CMS that is usually used by more advanced users who only need support for pages and 'stories' with comments. It has got out-of-the-box support for RSS, advanced menus, dynamic blocks for easily moving content to another part of the site and more. The standard skin uses a tree-like structure for most menus. The creation of content and the management of earlier created content has been separated in two sections, which can be cumbersome if you want to quickly edit something when you're already in the depths of the 'Administer'-part of the menu. Luckily, there's an insta-edit button on the pages when you're logged in.

Overall, the standard Drupal is best-suited for simple websites without a lot of custom content. The block-system uses a handy drag-'n-drop interface and if a proper template is used, it can give very nice results. Administration is unfortunately less than optimal if you want to edit multiple pages in a short timespan. The 'quick edit' option is a good bypass, but doesn't solve the underlying problem of a tree-view menu that collapses too early.

Drupal comes second on the list detailing the amount of people using a particular CMS, with 14%. The average weekly downloads gets the third place, with 62 500 downloads.

6 Redesigning Interaction

In the context of a web application, you will be actively working with on-screen data. Data aggregation is important and the way most OS-es do that is by means of nestable 'folders' that can consist of more items.

Most basic operations, like creating a new item, editing an item, renaming an item, deleting an item and creating a folder, are available in Frag-min and are fully customizable by module developers. Sorting items/folders based on certain criteria is also possible, but is as of this writing not implemented.

Drag-'n-drop interfaces make moving and ordering items easy. This interaction is not limited to folders of a certain common parent module, but has inter-module capabilities which I will call "transformations".

6.1 Transformations: a new paradigm in web applications?

A transformation is defined as a functional mapping or relation from module A to module B, with A and B being distinct.

$$B(data) = Transformation(A(data))$$

One of the useful features of a desktop-like GUI is the ability to use a lot of drag-'n-drop to move items around. In Frag-min the module dictates the possibilities of what is being allowed to be dropped on what. Imagine you have an article that you want to use as a news post, but the article module uses a database field called 'title' and the news module's one is called 'heading'. Frag-min supports transformations which can be defined by the developer to transform an item of module A into a piece of module B, with A and B being 'article' and 'news' respectively. For advanced modules, an array of 'fall back'-modules can be defined. Specializations of classes can fall back to the main class for transformations. An example being a specialized Image_Fourier-module being able to use Image's resize action.

Transformations are a very powerful concept because there is no theoretical limit on the capabilities of the transformation. Anything that can be programmed, can be a transformation. Transformations can be used to do tremendously innovative things when coupled with a drag-'n-drop interface. A list of a few options that are possible is detailed below.

- Changing an article into a news post
- Auto-resizing an image when dropped into an 'Image Resize' module

- Transcoding a video with a 'Video' module using a server-side script
- Fast importing of a CSV-list with addresses into a database in an 'Email' module
- Merging a meta-data module with the data itself to form a new data-object
- Creating a video slide show/presentation based on a 'News' module
- Instantly synchronizing certain data across multiple locations or data centers with a 'Synch' module
- Navigating through large (ontologic) databases with parent-child relations
- Sending an email by dropping the text the user wants to send on the 'Email' icon

The transformations are made in a white-list fashion. A 'News' module doesn't accept anything by default, until the transformation is made possible by defining a News_to_Article function in the server-side transformation.php file for example.

6.2 Extending the Canvas for Interaction

Using the concept of advanced transformations it's possible to create some more interactivity that can resemble the Active Desktop[22] or its successors Gadgets/Widgets on some level.

In Web-focused CMSes there are two major types of data that will be inputted most of the time: non-formatted plain text and HTML. Having a short-cut to the creation of these types of documents saves users having to open the needed module first. It's also handy as instant reminder-notes.

7 Improving Frag-min

Frag-min does not have the big community and history or the amount of developers and designers that the other mentioned CMSes have. This means that Frag-min is still in its infancy and a lot can be improved and added. Parts that come to mind are:

1. Working TaskBar

2. Authorization
3. Per-module, per-action Access Control
4. Improved graphical design
5. More standard modules
6. More standard transformations
7. Improved speed
8. Multi-user support for collaboration

8 Field tests and future research

Frag-min is meant as a prototype or proof-of-concept so before I can analyze the effects, it will need to be tested. A few opportunities for testing Frag-min in a live environment are:

- Human Computer Interaction Website of LIACS
- Ontology website (Cyttron)
- Scouting website with multiple layer menus

9 Direction for the future

Building a high-performance web application that tries to mimic a desktop OS's graphical interface is possible. By using the latest technology, new paradigms can be found to speed up the workflow and transform items into new items with changed properties which leads to new handy features.

References

[1] Human Computer Interaction (3rd edition), Alan Dix et al., 2003, Pearson-Prentice Hall, Chapter 1

[2] Joomla!, http://www.joomla.org, accessed at October 21st, used 1.5.14

[3] Cloud computing, http://en.wikipedia.org/wiki/Cloud_computing, accessed at October 21st 2009

[4] Prototype JavaScript framework, http://www.prototypejs.org, accessed at October 21st 2009

[5] Script.aculo.us, http://script.aculo.us, accessed at October 21th 2009

[6] We're all guinea pigs in Google's search experiment, http://news.cnet.com/8301-10784_3-9954972-7.html, accessed at October 21st 2009

[7] Marissa Mayer at Web 2.0, http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html, accessed at October 21st 2009

[8] Design Fast Websites, http://www.slideshare.net/stubbornella/designing-fast-websites-presentation, accessed at October 21st 2009

[9] Response Time Overview, http://www.useit.com/papers/responsetime.html, accessed at December 16th 2009

[10] Lazy Evaluation, http://en.wikipedia.org/wiki/Lazy_evaluation, accessed at October 21st 2009

[11] AJAX: a new approach to web applications, http://www.adaptivepath.com/ideas/essays/archives/000385.php, accessed at October 21st 2009

[12] Ajax(programming), http://en.wikipedia.org/wiki/AJAX, accessed at October 21st 2009

[13] JSON, http://www.json.org, accessed at October 21st

[14] CMS Statistics, http://www.cmswire.com/downloads/cms-market-share/ , accessed at November 9th 2009, password-protected

[15] WordPress, http://www.wordpress.org, accessed at October 21st 2009, used 2.8.4

[16] Drupal, http://drupal.org, accessed at November 9th 2009, used 6.14

[17] High Performance Websites, Steve Souders, O'Reilly, First Edition

[18] Pro JavaScript Techniques, John Resig, Apress, First Edition

[19] Resizable Demo, http://www.prodevtips.com/demos/staculous/demo1.php, accessed at October 21st 2009

[20] Resizables, http://blog.craz8.com/articles/2007/01/02/resize-is-in-the-wild, currently down

[21] PHP, http://www.php.net, accessed at October 25th.

[22] Active Desktop, http://en.wikipedia.org/wiki/Active_Desktop, accessed at October 22th