coffee_shop: Investigations into modern word processors.

Nathaniel Welch, Dr. Clark Turner

California State Polytechnic University

June 10, 2011

Abstract

My senior project was spent building a desktop application similar to WriteRoom and OmmWriter. These two applications are both word processors for the Macintosh OS X operating system. Both of these applications are designed to be a reset on word processing software, bringing their interfaces back to the days of Microsoft Word 3.0 and Word Perfect. They do this by spending more time on focusing on the design of the software interface, and focusing on keeping it minimalistic, instead of filling the product with new obscure features, which is a common complaint against the current iterations of Microsoft Word.

The final application, named coffee_shop, ended up not meeting my expectations. Having spent most of my education developing applications for the internet, instead of the desktop, I ran into pitfalls which, if this had been an internet application, wouldn't have been problematic.

Contents

List of Figures

1 Introduction

Depending on the job, people use different tools. Some tools are incredibly specialised, such as post hole diggers and PVC pipe warmers. Others are much more generic, such as hammers and cameras. Notice though, that in the hardware world, the majority of tools serve one job. That job, such as hammering, can be applied to a wide variety of ways (hammering in a nail, breaking apart structures, putting stakes in the ground). In the software world, historically tools have been built to be much more generic. Microsoft Word for example not only let you write documents, but also create spreadsheets, resumes and many other things. While this has let Microsoft make more money by selling their software to a wider variety of needs, it has created software that is hard to use and hard to maintain. To fight this, companies and individuals are starting to create software that have less features and are easier to use.

2 Problem description

The problem coffee_shop tries to tackle is whether or not developing a desktop application focused on writing, instead of formating, is a good idea.

3 Survey of relevant work

Before starting the project, I surveyed a variety of word processor programs and interviewed individuals about their writing habits. The survey I distributed was simple and open-ended. I asked users what types of documents they wrote, what they disliked about their current word processor and what they liked and wanted in their word processor.

I discovered that there were two groups of users of word processors in my survey group. There were those who wrote in a corporate or academic environment and those that wrote for themselves. The users that wrote more for academia and corporate positions tended to want lots of formatting options. Users writing for themselves usually wanted something that hid everything and let them just write.

Every single one of my responders despised Microsoft Word's auto-correct but still wanted spell check.

I decided to name my user group Johnny, based on two different people that I interviewed and the responses from my survey. Johnny writes fiction in his free time and aspires to be a writer. He is currently employed doing other things, so he uses his word processor as he commutes and during his time off. Most of what Johnny writes tends to be one or two pages, but he has been known to turn out novels depending on his mood.

3.1 Professional Offerings

As stated earlier, there are a variety of choices in the word processor market. The ones I tried and examined were chosen both by their reputation and based on how easily I could get my hands on a copy.

3.1.1 Microsoft Word 2010

Microsoft Word 2010 is a pretty expansive program. It easily has more features and options than any of the other products I tested.

Its while it has a relatively easy to interpret selection of main controls on the top of the page, most options are hidden under a variety of menus. Disabling features that many who I surveyed complained about, such as auto-correct, took a few minutes to find the right check box.

Despite its weight, Word is incredibly powerful, and no matter what a user needs to do, Word should be able to do it.

3.1.2 Microsoft Word 5.5

Originally released in 1991[8], Microsoft Word 5.5 is an interesting beast. 5.5's initial downside is that it needs to be run in DosBox. The cool thing about DosBox though, is that it means I can run 5.5 on Linux and on just about every other machine that it supports. But 5.5 comes from before the days where everything is just one executable. The user install experience is incredibly slow and

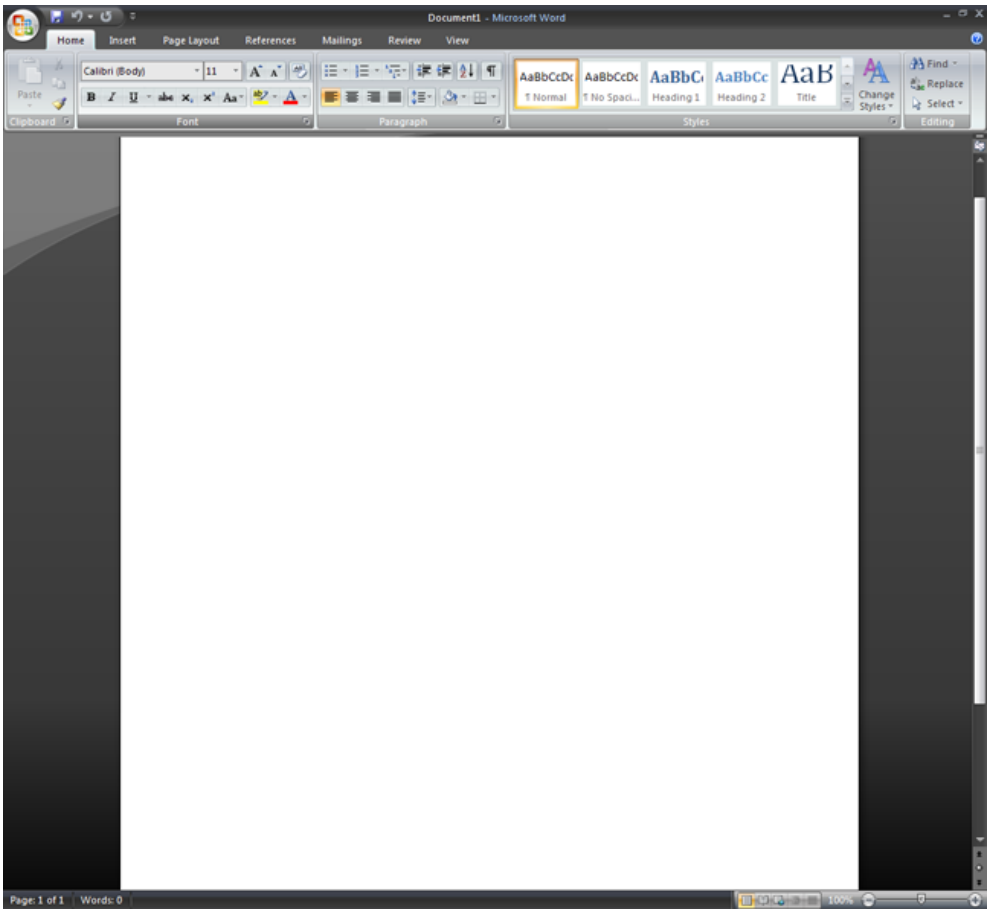

Figure 1: Microsoft Word 2010 with no text.

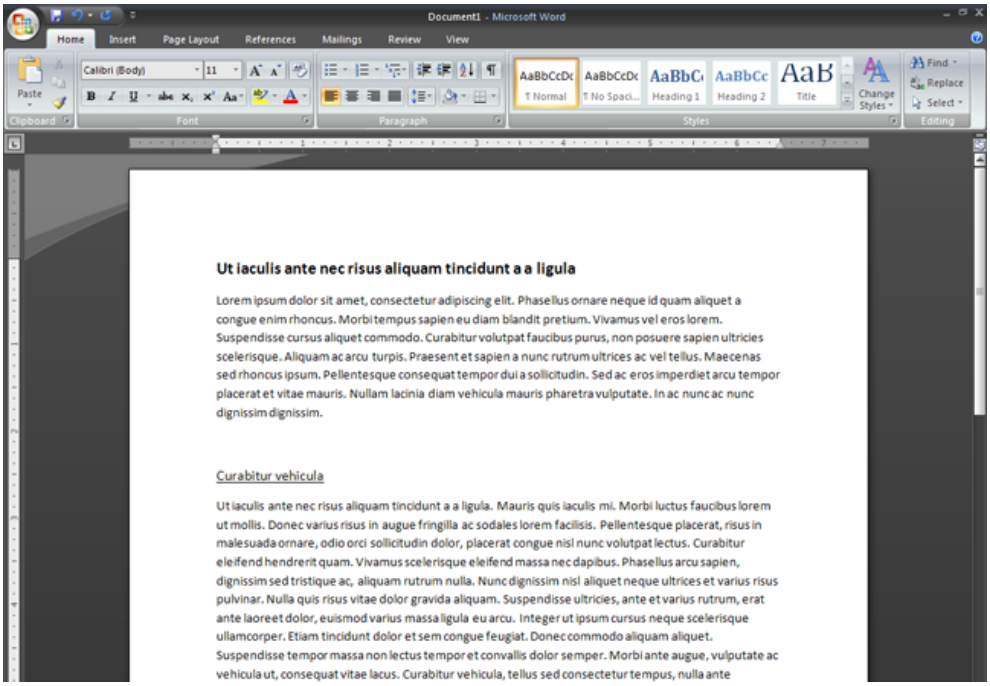

Figure 2: Microsoft Word 2010 with text.

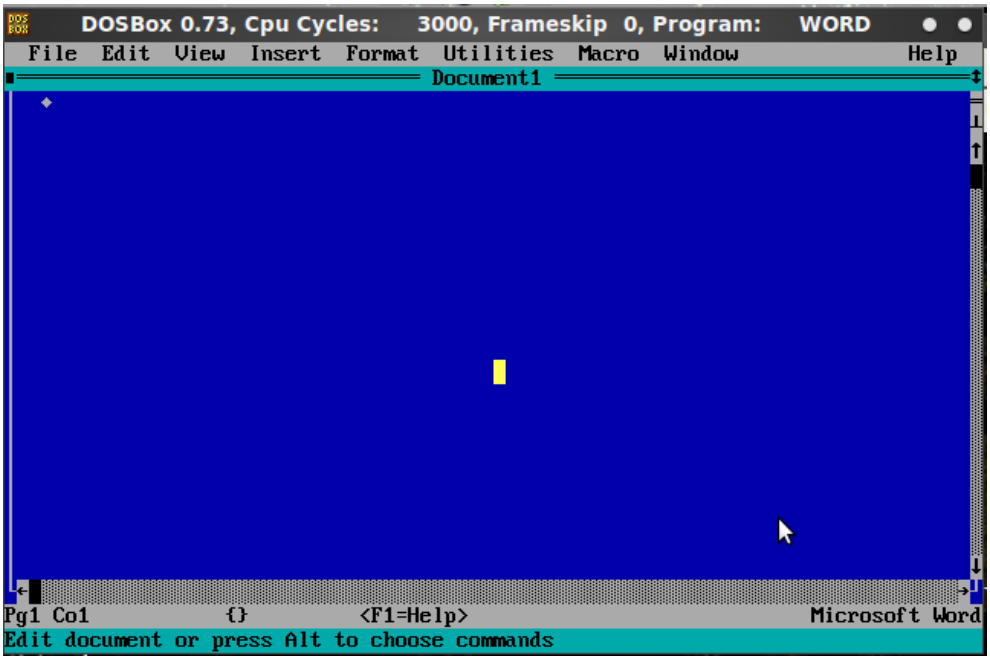

Figure 3: Microsoft Word 5.5.

time consuming. 5.5 extracts four hundred files before setup, and then another two hundred during the install process.

Once you are inside 5.5, the experience is relatively nice. Document navigation was done entirely via the keyboard. It reminded me a lot of Pico, and was a joy to use. The user can also select text with the mouse, and we can see how the experience in 5.5 turned into that of Microsoft Word 2010.

5.5 offered no color or font customizations, although the user could go full screen, which was very nice. Page breaks were implied with an emphasized dotted line.

3.1.3 Pages

The Apple offering in the word processor space. I found Pages rather nice to use, and its full screen mode was very similar to what I was looking to make. The downsides were simple though. Pages' default file type cannot be read by any other program or operating system, and you could do very little to customize the user interface.

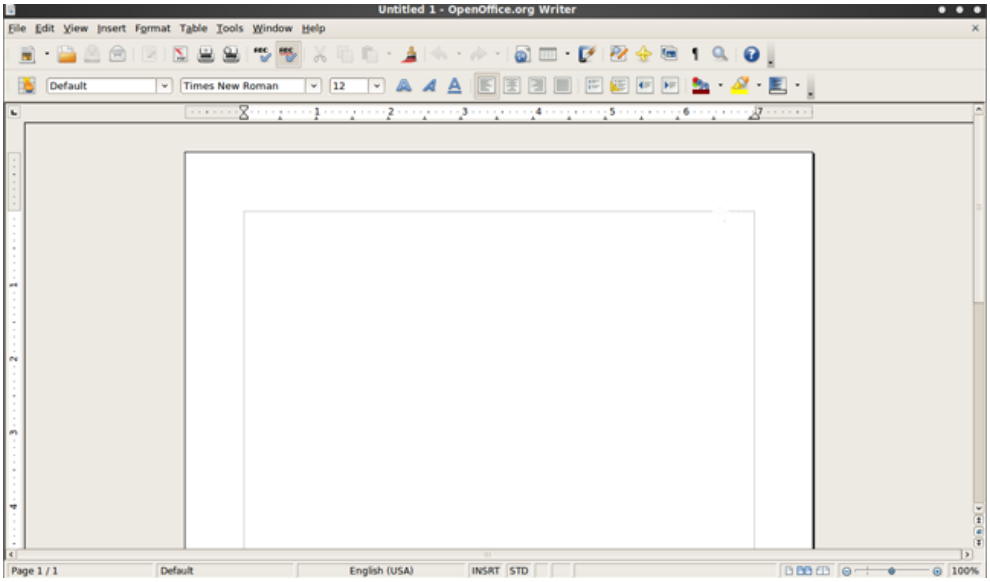

Figure 4: OpenOffice Writer 3 with no text.

3.1.4 OpenOffice

OpenOffice Writer 3 seems to be on-par with current Microsoft Word offerings. The product has a ton of features, most of which were hidden behind obscure icons and menus.

Writer was nice in that it always opened a new empty document so you can start writing immediately.

The full screen mode also looked pleasant, but tended to crash when transitioning in and out.

Writer also had an easy to use "export to PDF" feature. Spell check was easy to find compared to many of the other programs examined.

You could customize a lot of things in Writer, but it was annoying that you had to select all of the text to change the document color scheme. In contrast though, Writer supported more fonts than any of the other programs.

Overall Writer is a nice offering considering it is free, but the interface can be intimidating, especially if you have no idea what all of the buttons do.

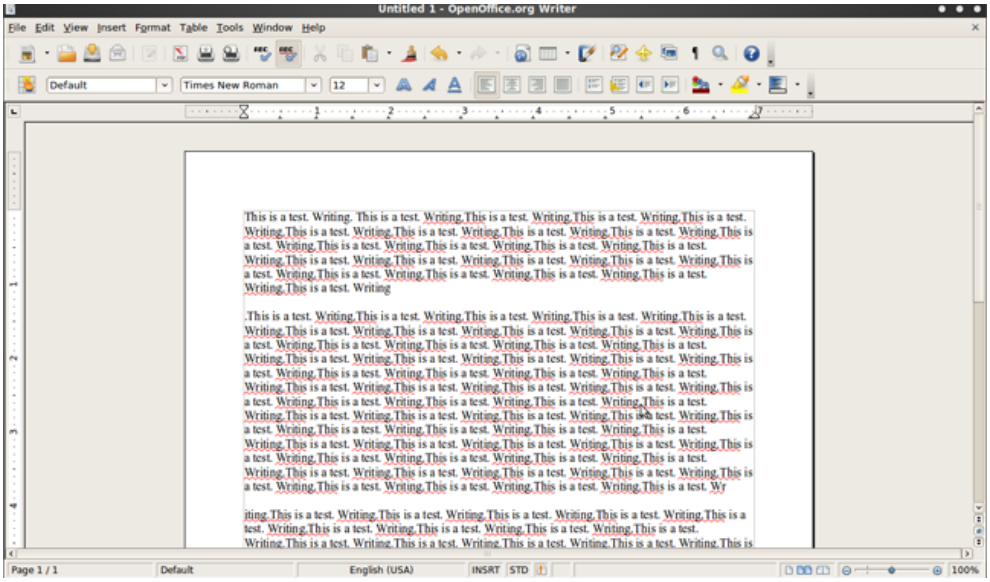

Figure 5: OpenOffice Writer 3 with text.

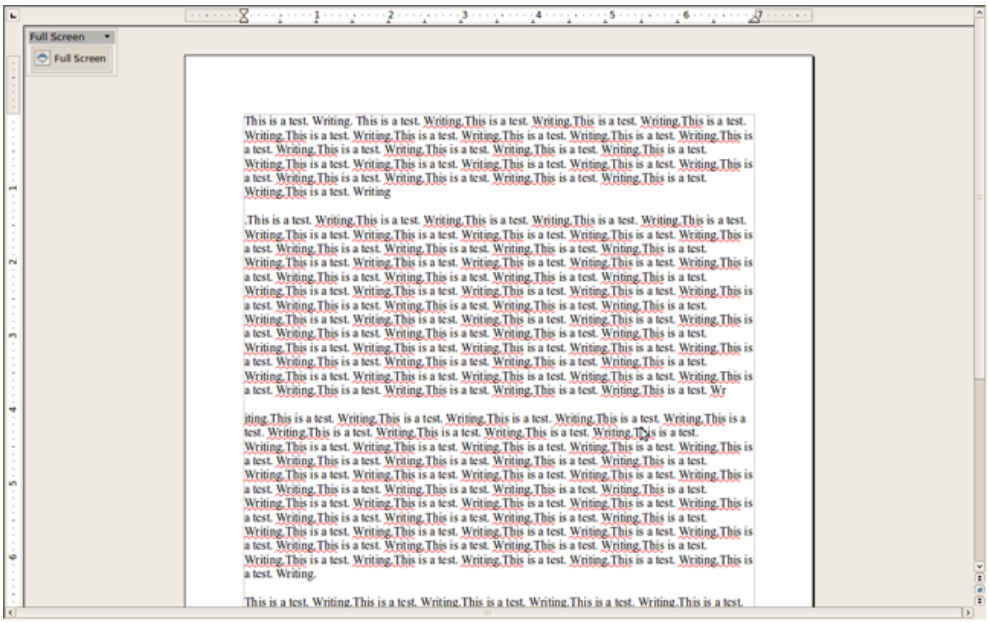

Figure 6: OpenOffice Writer 3 in full screen mode.

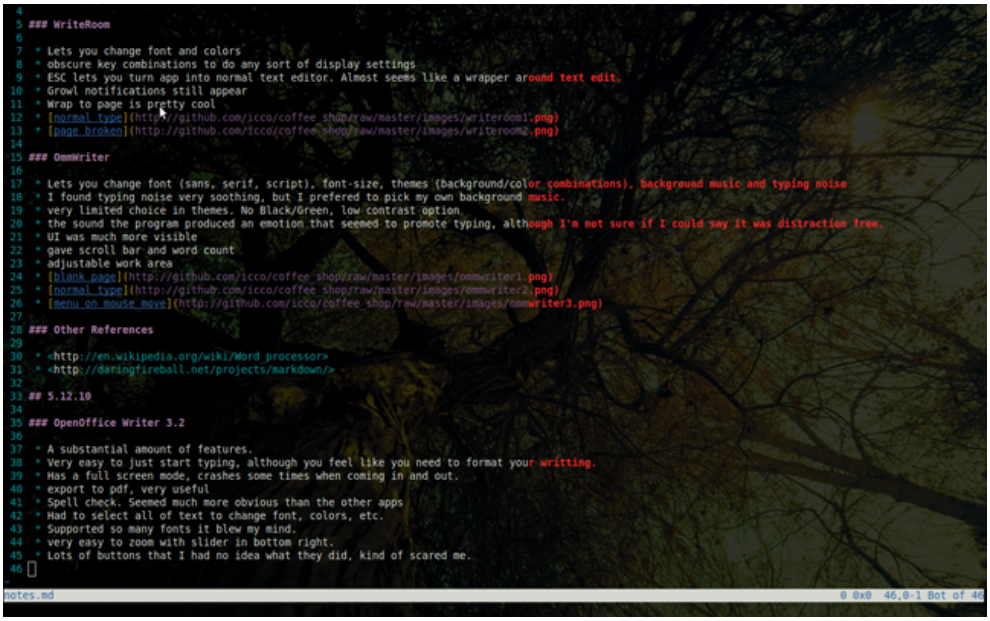

Figure 7: Vim with one file open.

3.1.5 Vim

According to the Vim website, vim.org, "Vim is a highly configurable text editor built to enable efficient text editing. It is an improved version of the vi editor distributed with most UNIX systems" [6]. I love Vim, and since starting this project, I've pretty much decided that in almost every scenario, I would use Vim to write things.

I currently am using Vim to write this paper, I used it to write almost all of the code for coffee_shop and I have been using it pretty exclusively for the last four years.

Vim has its issues though. It has a really difficult learning curve. In fact, without working for a company where almost every developer used it, I probably would have given up long ago. Most people give up quickly because they do not understand why Vim tries to make sure you never use the mouse. In their defense, this is fair, the user interface does not tell the user much at all, and expects the user to know all of the magic incantations to get things to happen. I have heard that emacs is the same way, but I have never used it.

Figure 8: WriteRoom with one page of text.

3.2 Similar implementations

These are programs I am trying to emulate while developing coffee_shop.

3.2.1 WriteRoom

WriteRoom is a piece of software for OS X by HogBay Software [7]. By default WriteRoom opens in full screen. Pressing escape though turns the app into a normal windowed text editor. This windowed text editor looks very similar to OS X's Text Edit.

WriteRoom has a lot of cool customizations. It lets the user choose colors and fonts. It also does a really nice job wrapping and splitting up pages.

The only thing I found annoying was that you needed to know obscure key combinations to do change any sort of display settings. This seemed weird, because while key bindings let you keep the

Figure 9: WriteRoom with two pages of text.

Figure 10: OmmWriter immediately after launch.

interface simple, users can't use them if they aren't taught them when the application opens.

3.2.2 OmmWriter

OmmWriter [4] is similar to WriteRoom in its feature set, except that it focuses around themes. Users select from a limited set of fonts, background and color combinations, background music, and typing noises.

The typing noise was quite an interesting idea. I tended to mute the background music, but noise of clicking typewriter keys ended up being quite soothing.

Probably the biggest complaint comes from the limited number of themes. I was looking for a high contrast option, such as light green on a black field. There was nothing similar to this, and most themes were very low contrast and hard to read.

The workspace was adjustable in size, and the user interface came and went based on mouse movement and typing speed, which was a nice touch. OmmWriter seemed the most polished of everything I tried out, and was the program I probably tried the hardest to emulate.

Figure 11: OmmWriter while typing.

Figure 12: OmmWriter while hovering over a menu item.

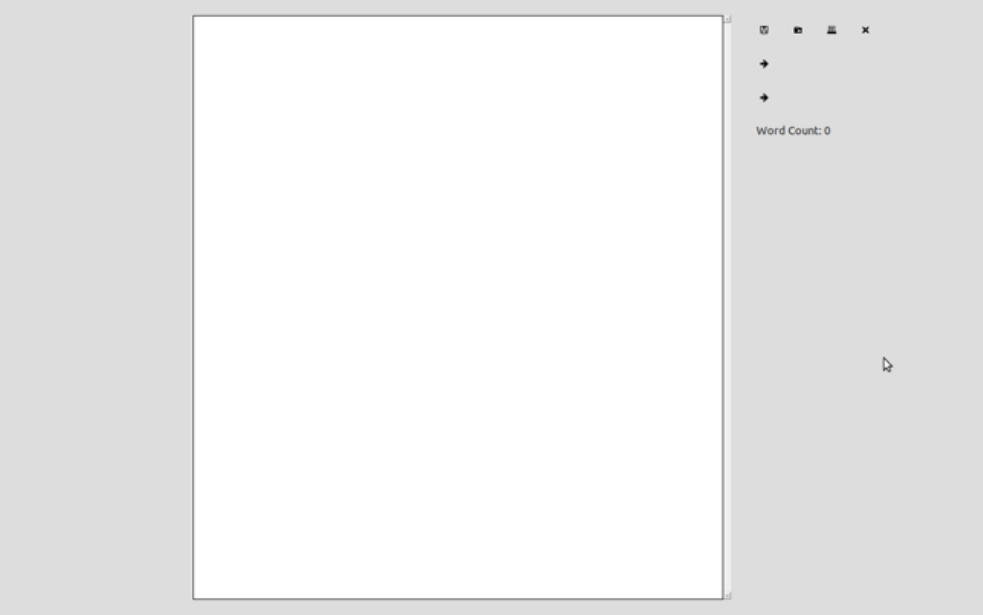

Figure 13: coffee_shop after launch.

4 coffee_shop

Talk a little about the development process. We talk about how features were implemented below, so just give an overview of what the program does.

5 Evaluation of coffee_shop

Below is a list of requirements determined at the begining of the project and whether or not coffee_shop passes them. In the following requirements, "it" refers to the application, coffee_shop.

5.1 Functional Requirements

FR1 It must allow the user to type into the application.

Passed.

FR1.1 As the user types, text will appear onto the screen.

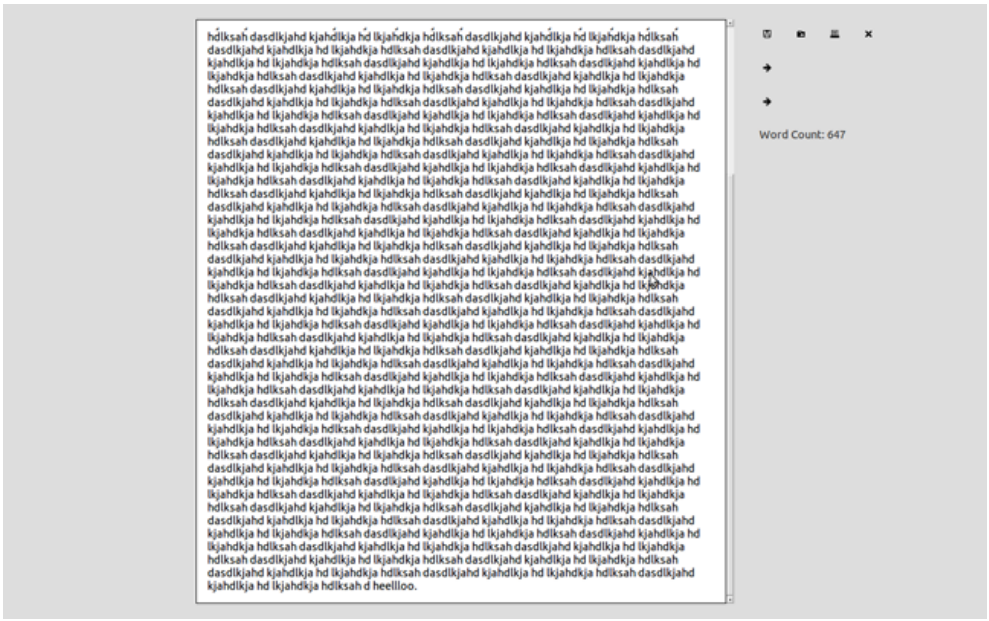

Figure 14: coffee_shop with text.

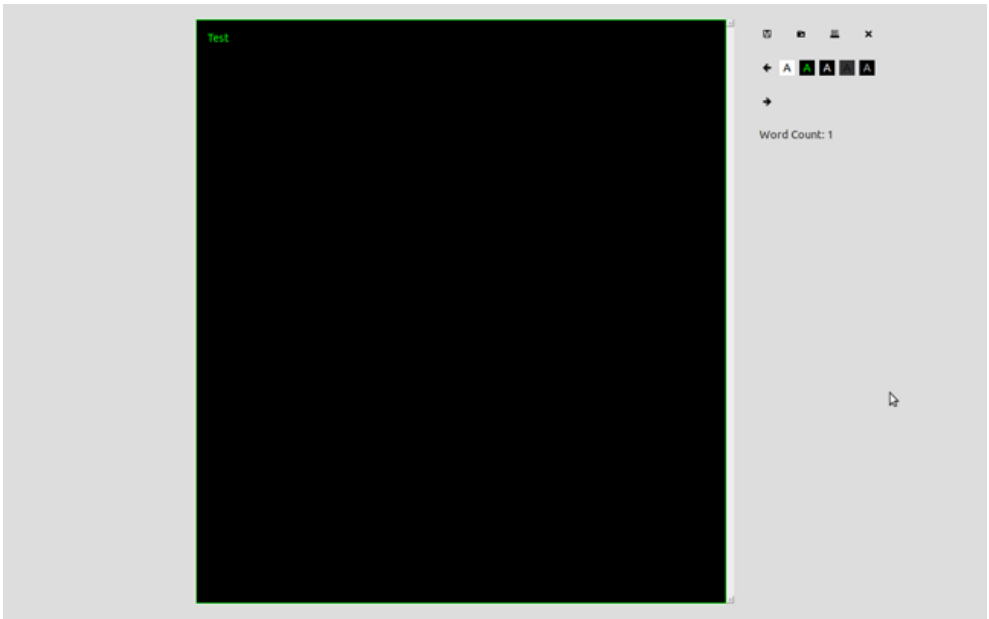

Figure 15: coffee_shop using a different color pallet.

Figure 16: The coffee_shop splash screen designed by Jeannie Nguyen[2].

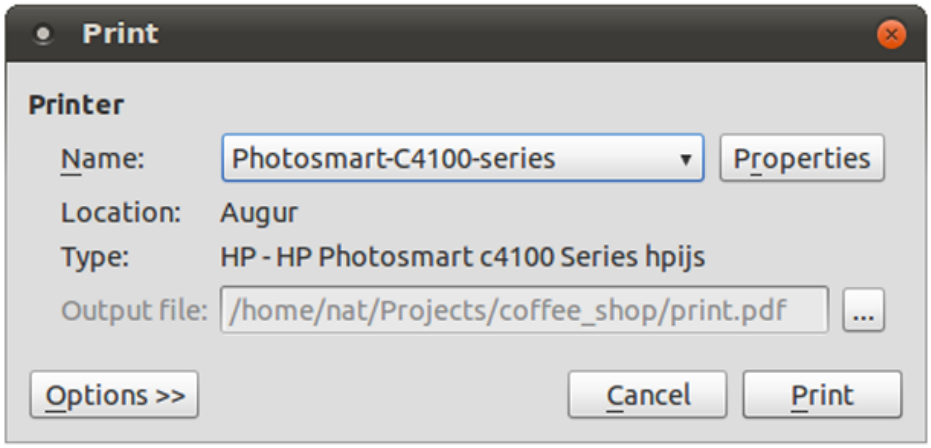

Figure 17: coffee_shop's print dialog.

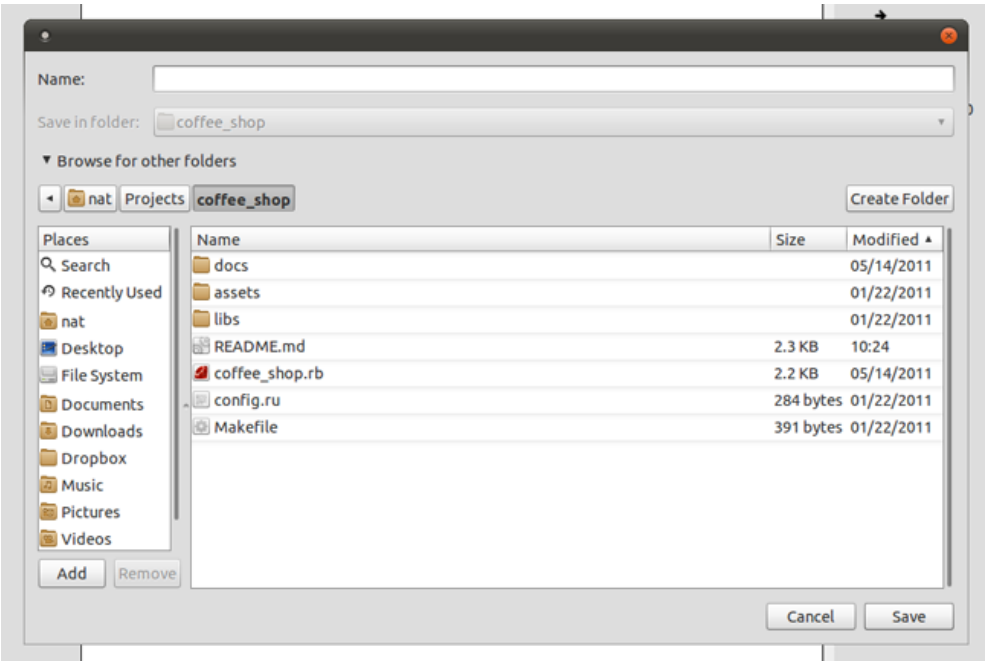

Figure 18: coffee_shop‘s save dialog.

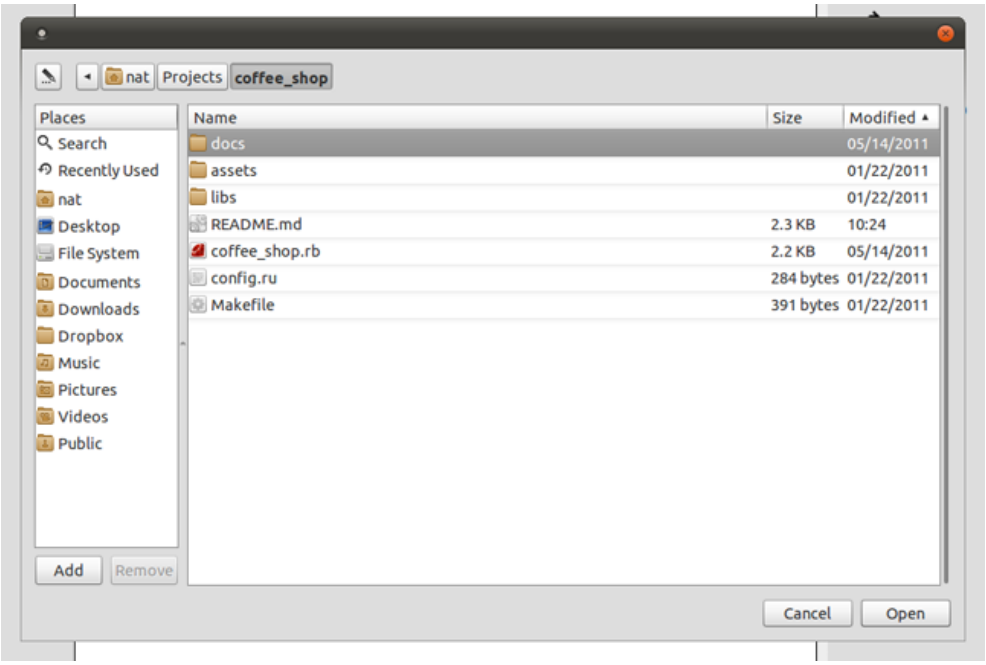

Figure 19: coffee_shop‘s open dialog.

Passed.

FR1.2 Certain keyboard shortcuts will cause the system to act in a predictable way. Such as CTRL+s will force the program to save the current working document.

Passed, although as find and replace is not supported, CTRL+f does not work.

FR2 It must be able to save and open the users work.

Passed.

FR3 The work environment must be customisable.

Passed.

FR3.1 The user should be able to select a background color.

Passed.

FR3.2 The user should be able to select a font color.

Passed.

FR3.3 The user should be able to select a font, from a limited selection of fonts.

Passed.

FR3.4 These setting should be saved between sessions.

Passed.

FR4 It must support find and replace.

Failed. I ran out of time and never implemented this.

FR5 It must be able to display a word count.

Passed.

FR6 When the user clicks the print menu item, the dialog to print must be displayed.

Passed. It prints as well.

5.2 Non-Functional Requirements

NFR1 It must be Open Source.

Passed. The application is available for download at `https://github.com/icco/coffee_shop` under the MIT License.

NFR2 The language it is written in must be Open Source.

Passed. The program is written in Ruby, which is available under the GPL v2 at `http://www.ruby-lang.org/en/`.

NFR3 It must run on Linux.

Passed.

NFR4 It must run full-screen.

Passed.

NFR5 It should promote a distraction free work environment.

Passed.

NFR6 It should be usable by people who work in various lighting environments (writing in a coffee shop, dark room, corporate environment).

Passed. I was able to use it in all three of these places.

NFR7 The format the work is saved in must be standards compliant format.

Passed. Saves in plain text.

5.3 Development decisions

I made some decisions while developing and designing coffee_shop, which may be considered controversial or unintelligent. The two big decisions I made were to use Ruby as my programming language and Qt as my graphics framework.

5.3.1 Ruby

Ruby is an interesting beast. I selected it because I had little experience with the language and it had a very large community online. Since starting the project I have learned a lot about the language.

Ruby is a really fun language to write in, but it is not made to develop systems whose state changes during its run. Ruby works far better as a shell script or responder for a website because it can execute in entirety and doesn't need to jump around to a ton of objects already allocated. It's possible that a graphics framework could solve this, but I was unable to find one that did.

5.3.2 Qt

Qt is incredibly powerful, and RubyQt [5] is probably the best UI toolkit available for Ruby. But Qt is written for Java and C++. Qt expects to have references to everything, and wants everything to be passed by reference. Ruby doesn't really work this way and because of it, hindered Qt's power.

Interestingly enough, a lot of the things I saw other applications doing that I thought were cool, were actually areas where people were hacking directly on the Qt source, or rewriting entire sections of the library. [3].

In hindsight, if I were to use Qt, I would use it with C++, instead of one of it's many ports to other languages.

5.4 Features and their implementations

5.4.1 File writing

Pretty simple, although lots of discussion.

5.4.2 Pagination

So, this is the one feature I really wanted to get work, but I just couldn't figure it out. It seems to be that Qt only wants to do pagination when the data is read only. So, for example, you can send Qt a block of text to print preview, and it will return you a read only set of paginated data. All

of the actual text document objects though, do not support pagination while typing, which makes them all pretty useless. Posting on the Qt forums and on StackOverflow, the general opinion was that I should fork the Qt source and modify the "TextDocument" object myself in C++.

This made me furious, in what world is it good API design to make your library need to be modified to get a desired effect? I thought the internet was crazy, but I looked at the source of some of the major Qt users, such as KWrite, and was blown away. They reimplemented about half of the Qt framework because the framework didn't do what they wanted to. On top of that, this reimplementation was close to two thirds of their code base.

Frustrated, I tried to do pagination myself. I tried a variety of methods to split the data between text boxes. The problem with this was that whenever I did this, focus was put back on the top of first text box. So depending on how often I recalculated the pagination data, users would get incredibly incoherent data. The worst was when I recalculated pages after every key press. This made it so when users typed data went into the document backwards (as if they were pushing their characters onto a stack), but when they pasted, the text went in the right way.

After a while, I just moved on and abandoned the feature. I've written the Qt maintainers, and their response is not to use language wrappers, such as RubyQt, which was probably the most unhelpful response I have ever seen.

5.4.3 Printing

This was surprisingly easy thanks to Qt. All I really had to do was open a "Qt::PrintDialog", which prompted the user to select a printer, and then send a block of text to the object representation of the printer they selected.

5.4.4 Customization and preference storing

Storing customizations was a pretty easy technical challenge. It required creating a singleton class, which let me store key value pairs to it. Then we would write out the changes to disk on occasion. We would also make sure all of the data got written out when the program exited.

The biggest problem was trying to support configuration changes while the program was running.

Often Qt objects didn't want to be redrawn. Unlike an element in the DOM tree which can be styled dynamically with ease, Qt often made strange choices when to redraw with the new settings it was given. It seems like these redrawing actions are a common problem in the Qt world. When I asked around Nokia and Qt's forums, I was told I should just modify the core Qt code to get the desirable effect I was looking for.

6 Future Work

I think there is lots of work to be done here, but I would argue against it. Instead of continuing to develop this application, I would recommend looking into building a web application that simulates the same experience. CSS makes it easier to iterate quickly on the design and layout, plus you can use the HTML5 local storage API to let the user work offline.

If you were to continue working on this project, I would focus on making the application easier to deploy across a wide variety of operating systems. I would also try and find a layout that works well on all screen sizes.

7 Conclusion

Overall, I am proud of the work I did, but in the future I would probably just make this on the web.

References

[1] "Ieeeannot: A latex ieee annotated bibliography style template." [Online]. Available: http://www.barik.net/sw/ieee/

the web page where you can download the IEEE annotate style to allow annotations in a bibliography. Put it in the same folder as the .bib and .tex files.

[2] "Jeannie nguyen." [Online]. Available: http://www.jeannienguyen.com/

Homepage of the designer of coffee_shop's splash screen and icon

[3] "Kate / kwrite." [Online]. Available: http://kate-editor.org/about-kate/

Provides a nice overview of the KWrite project.

[4] "Ommwriter." [Online]. Available: http://www.ommwriter.com/

OmmWriter's product page.

[5] "Rubyforge: Korundum/qtruby - ruby-kde/qt bindings: Project info." [Online]. Available: http://rubyforge.org/projects/korundum/

Provides an overview of the Ruby Qt project.

[6] "Vim.org: About." [Online]. Available: http://www.vim.org/about.php

Provides a nice overview and history of the Vim project.

[7] "Writeroom – distraction free writing software for mac." [Online]. Available: http://www.hogbaysoftware.com/products/writeroom

WriteRoom's product page.

[8] R. Needleman, "Microsoft Word 5.5: Should You Fight or Switch?" *InfoWorld*, vol. 12, November 1990.

An article talking about Word 5.5's release.