# Design and prototyping – 18:53 14/06/14

## Discussion

Today’s work was about designing the interface of the app. First I drew some drawings in paint with what I wanted the interface to look like. The general theme I was going for is a minimalistic and clean interface, similar to Windows 8 (I thought they did an excellent job with the aesthetics of it). The form factor of the application is designed so that it can be a kind of desktop buddy style program, which you use for reference when doing work. Using this form factor means I can easily transfer to a mobile app later on. Secondly I linked the different screens of the application together using a navigation diagram. (I kind of knew the hierarchy when I was designing the screens.) Then, I opened Livecode and worked on a few prototypes of the screens I drew in paint. Finally that, I began the design documentation, and started the journal and put an entry in.

## Learning

I knew Livecode had some kind of rich text renderer, so I spent about an hour figuring out how to make that work, and managed to get it to load html code from a file and put it in a field. Turns out, you need to set the HTMLText property of something to some HTML code and the field will render it as HTML code. An important note is that it’s only a Pseudo-HTML in that there are only about 7 or 8 tags. They basically serve the purpose of providing a way for the user to style text in Livecode easily.

# Evolving the prototypes – 22:00 14/06/14

## Discussion

Tonight I fleshed out the prototypes into actual working interfaces, and began to develop a structure to load the html source for the content pages. Navigation is easy, it’s just changing cards, but loading the HTML content could be simplified if I code it the right way.

## Learning

I figured out that I could make the HTML loading a lot easier if I simply loaded the HTML source from a file based on the name of the card. Maybe something like:

get the name of this card  
set the HTMLText of field “ContentArea” to the HTMLText of URL it

But I’ll have to wait and see.

# Halfway there – 9: 31 15/06/14

## Discussion

Turns out the idea for loading the HTML through the card name doesn’t work as intended, since windows doesn’t like spaces at the start of a file name. Also, the go to dropdown was a bit of a hassle as well, since you can’t go to a card via a variable name, even though it should work theoretically. On the bright side, I’ve implemented all the code and interface for about half the app now, and the other half isn’t too far behind.

## Learning

As previously mentioned, code-wise, there were several hiccups, but I patched them through some ugly but working code. While it does mean more work, it also means that I can rest at night knowing it works.

The main time-saver for me this project has been developing a good looking template that can easily be duplicated many times to work well. For example, when creating new content pages, all I have to do is copy one of the other pages and change the title and html source file. Another time saver is a quick html editor I made so I can write html files in Livecode and preview the look of them instantly.

I also abandoned the loading html files thing when I realised that I need to distribute all the source files along with the application if I do that, and it’s just too much hassle! That said, I’ll need to distribute the images and videos with the program somehow, but I may just host them on the web so people only need to have a copy of the app, not all its resources.

# Interface and code done – 12:27 15/06/14

## Discussion

The interface is now completely done, and I’ve been experimenting some more with loading pictures off of the internet. It seems like this is going to be a way easier method of doing this than actually including the images in a folder and messing with that crap. Next up is videos! Maybe…

## Learning

With Livecode, I’ve found that it’s very complicated to display images stored on disk, but it’s very easy to use images off the internet. But, because this software is being accessed by many people, I should host them myself. Because taking advantage of other people’s hosting is bad.

# Web hosted file solutions! – 14:07 15/6/14

## Discussion

I have finally arrived on a method of displaying content in the app. I’m going to use google drive to host the HTML code for each content page, and use their live previewing feature to edit and obtain formatted text. As for pictures, they can simply be linked through an img tag, and are being hosted by imgur. I should probably add some sort of special thanks to both services for that.

Now that I think about it, I should make the app more colourful. It’s currently only 2 colours, and even with images, it looks quite bland.

## Learning

As mentioned before, the best solution for Livecode files seems to be finding a host that provides direct links to the content you would like to see.

# NOT Web hosted file solutions! – 13:12 11/07/14

## Discussion

After some testing on the actual platform of deployment (school computers), I found that having the files hosted on the internet and being streamed onto the application wasn’t going to work. Ultimately, the school computers don’t have an internet connection when it comes to LiveCode applications, and the problem there lies that content can’t be streamed off the internet if the device can’t connect to the internet. However, not all is lost. It turns out that developing the content on “the cloud” and then streaming it to *the source of the application* while I’m designing the app means I can design and host the files on the web, and then just update the content in the source to see what it looks like in the application.

That all probably sounded a bit convoluted, so here is a step by step walkthrough of the new workflow:

1. Design the content of the application, including text and pictures. When the content is done…
2. Stream it back to the Livecode editor. I do this by opening the message box (LiveCode’s version of an interactive prompt) and manually set the content of the page to text from a URL (in this case, a google docs hosted page)
3. When happy with the content and how it looks in the app, compile the app.
4. When you want to distribute updated content, simply repeat steps 1-3

## Learning

It’s always important to check that your application will work on the intended platform early in the process, as well as every so often, because if it works on your development device that’s all fine and good, until your consumers want to sue it on their devices and your product doesn’t work in practice. The moral of the story, test it on your target platform when you make any major changes to the app.

# 50% content complete – 15:43 11/07/14

## Discussion

I’ve now finished all of the compulsory content for the application, now all that’s left is the optional content. Of course, there’s still stuff to do after that, such as optimization, spellcheck, bug testing and more. But 50% of the content is done. I also got an Icon going. But I don’t think it’s going to be the final application Icon. It looks a little clunky. For now, I’m going to go test out the application on the target platform and see how it runs.

## Learning

I’ve gotten into a nice pattern of creating the content pages, and I found an online PNG-ICO converter to make the application icon with. The current workflow seems to be pretty efficient.