Design Diary – Milestone 1

Despite the many frustrations that come with learning a new library entirely, I’ve been enjoying working on the TextEditor project with pdcurses. This has been great because it’s forced me to step back and consider exactly how each aspect of a text editor works and exactly what features are needed to make it appealing to users. One of the main things I enjoy is how it’s helped change how I think of programming assignments in general.

For instance, this project has made me aware of how useful it is to break down a big project into smaller, less intimidating portions that can be worked on slowly. So rather than think to myself “this will be really hard to do because I’ve never made anything like this before”, I started listing the features a text editor usually has and found that the majority of them are straightforward to make. Then once I started working on one thing at a time, the next thing I knew I had a semi-functional program. I think this is a great skill to develop because who knows what else I’ll be asked to make and knowing where to begin is the first step.

Another thing I’ve been really enjoying is the exposure to github. I’ve seen github before, but I’ve always been intimidated by how complicated it always seemed, so I really appreciate the hands on experience with pulling requests, committing all changes to my code, tracking issues, and regularly making updates to document my work. When I look at github now, it’s nowhere near as confusing as it used to be and I can navigate it with little to no problems.

As of right now, the most difficult aspect that I’m facing is making my code more robust. All of my code is current in int main, which I know isn’t a problem for milestone 1, but it means that int main has become long, unwieldy and maybe even a little difficult to read. So, sometime soon I need to go back and see what I can do to make abstraction happen with any code that might end up being reused a lot. I’m dreading this because it’ll like playing Jenga – the code works NOW, but the minute I start removing pieces to make functions and header files is when the whole tower topples and I have to do damage control to figure out what I broke. This is made so much worse considering how much time has gone into getting the code to work in the first place.

It won’t be fun but I do think it’s necessary because it’ll be expected for good coding. That and I think it’d be good practice for abstraction, debugging and getting a better understanding about what my program is doing.

A preview into future features I’d like to implement for milestone 2 as well as some thoughts regarding these features (so I can look back and compare later on):

* An interactive file menu header
  + This seems like a lot of work and I’m wondering if it’s the right path to take considering this text editor doesn’t really have a lot of features that could be hidden in the menu
    - The features that are present can be listed below in a layout similar to nano and might be the best route to take for a simple program
    - Question to self: do I need a working file menu or do I want to add one because of the time it took to make the file menu and now I’m reluctant to get rid of it?
  + If it is implemented, I’d like to figure out how to add more options, like:
    - undo (ctrl+z),
    - redo (ctrl+?),
    - word search,
    - open file
    - save file
    - word replace
    - There might not be enough spare time for this, if it isn’t scrapped.
* Get special keys working
  + Live text input is currently functioning, but the arrow keys, ctrl, esc and enter either don’t do anything or return weird results (i.e.: the left and right arrows both type the letter A) – these need to be fixed
* A working word wrap
  + Pseudo word wrap is in effect but it looks terrible (it stops the text mid-screen and was mainly so I could test to see if I could get it working at all)
* Vertical scroll bar
* Save feature for files