# Dream Design of a Text Editor

Andrew Kowalczyk

December 06, 2013

## 1 Design

Text editors are some of the most important pieces of software that programmers can use for thier craft. I think a redesign of this application using multiple interaction styles would be incredibly beneficial for a coder/programmer.

The design would mainly use audio recording from the user for all main actions. The ability to type in the editor is always a possibility, but when the user simply gets tired of doing that they can revert to their spoken voice for accomplishing their goal.

Certain key combinations would denote certain types of actions. Whenever a user will hold the key combination, the application would wait for some kind of input and only after the user lets go of the keys will the application respond.

### 1.1 Record code/text

⌥+R This would allow the user to record any text they want to have written down. Certain words would be keywords, like *for loop*, *if*, *section*, etc. would all have their overall syntax saved for any particular syntax of a file. So for example if a

⌥+R+V This denotes recording verbatim. This would tell the application to ignore keywords and to simply record whatever input the user has provided. This would allow for

### 1.2 File actions

#### 1.2.1 File syntax

⌥+Y This would allow the user to to change the syntax of the file. Sample input of the user includes: "Python", "C++", "LaTeX", "Ruby on Rails", "JSON", etc.

#### 1.2.2 File encoding

ctrl+⌥+E The user could change the encoding of the file. Input from the user can be: "UTF-8", "UTF-16 Little Endian", etc.

### 1.2.3 File actions

⌥+F Complete a specific file action. Input ranges from: "save", "save as", "new file", "open most recent", "open last closed file", "close window", "new window", etc.

## 1.3 Text and cursor manipulation

## 1.4 Cursor

### 1.4.1 Selecting text

⌥+S Select a certain part of the text. "word" "line" "paragraph" "scope"

### 1.4.2 Text style for LaTeX

⌥+S+T If a user is perparing a document using LaTeX, this would allow them to change specifc style of text.

### 1.4.3 Expanding selection

⌥+E Once a user has selected their text, they can expand their selection in a number of ways.

### 1.4.4 Selection manipulation

⌥+M Once a user has selected their text, they can manipulate it in several ways. Input includes: "put into if statment", "",

### 1.4.5 Line number

⌥+L Go a certain line number. Sample input includes: "fifty-three", "first", "last", "two", etc.

### 1.4.6 Moving text

⌥+M Once a pience of text is selected, the user can move the text as the please. Sample input that the user can provide: "up one line", "to the top", "move into XXX method"

### 1.4.7 Deleting Text

⌥+D Delete a certain part of the text. Input from the user includes: "word", "line", "paragraph", "scope", etc.

## 1.5 Scolling

⌥+A The user would be able to scroll depending on their input: Sample input includes: "up", "down", "top", and "bottom". The speed at which the text is scrolled at could be changed in the settings.

## 1.6 Preferences

⌥+P The user would be able to scroll depending on their input: Sample input includes: "up", "down", "top", and "bottom". The speed at which the text is scrolled at could be changed in the settings.

## 1.7 User defined commands

## 2 Usage Scenarios

### 2.1 Handicapped persons

The first user group that comes to mind are people who normally have some difficulty typing. Due to some physical barriers, they might be unable to input or manipulate text via the keyboard. This application would allow for those people who want to code, to be able to do so.

### 2.2 The tired coder

The second user group that comes to mind is the coder who is simply just tired of typing.

## 3 Rationale

### 3.1 Priorities

### 3.2 Mental Models

### 3.3 Interaction design concepts, guidelines, principles, and theories

Hello › What's › Up

## 4 Usability metric forecast

### 4.1 Learnability

The overall learnability of the application would not be the fastest. Learning a new set of key commands is never a quick thing to lean. Also learning new phrases for certain actions for users could take a while.

### 4.2 Efficiency

### 4.3 Errors

### 4.4 Memorability

In terms of memorability, the application would fare well. The same commands would do that same things and revisiting this application after some time would not take long to remeber how the application functions. Refamiliarizing onesef with the application would be straightforward.

### 4.5 Satisfaction

Not surprisingly, this metric is largely subjective. For people who previously had a hard time coding due to a disability, this would induce large amounts of satisfaction. For people who do not have any physical diabilities impeding them from typing, teir satisfaction would rely upon how effiecient they have become at using the application. The overall learning process could be