Ruby 101

Intermediate Ruby: User Input

Intermediate Ruby:

When 'gets' isn't enough.

IO has several ways to read input from files or the $stdin stream.

`.getc` -> retrieves the next character and waits for a new one

`.getbyte` -> retrieves the next byte and waits for a new one

-> getc and getbyte return nil at the end of a file (eof), they wait for user input if called on $stdin

`.read` -> read until eof, with blocking

`.read_nonblock` *`max_length`* -> read max_length without blocking

`.readchar` -> read single char with blocking until eof, raises error and blocks

`.readbyte` -> as above but with bytes

`.readline` -> like 'gets' but reaises an error at eof

Intermediate Ruby:

But these require a carriage return if used with $stdin

io/console includes .getch
    returns the first char, blocks when no chars are available

Intermediate Ruby:

Let's work this into our program loop:

controller.rb

```
    def run
        while user_input = $stdin.getch do
            #process the input
            begin
               while next_chars = $stdin.read_nonblock(10) do
                  user_input = "#{user_input}#{next_chars}"
               end
            rescue
            end
            if @current_view.quittable? && user_input == 'q'
                  break
            else
                    parse_input user_input
             end
        end
    end
```

Intermediate Ruby:

.display not in the input processing loop.

Instead in the initializer method:

```
def initialize
      @log_file = LogFile.new
      @current_view = FileDialogView.new
      @current_view.clear_display
      @current_view.set_cursor
      @current_view.display
end
```

Intermediate Ruby:

Quit with "q"?

User types 'q':
a) They are typing the letter into an input field
b) They intend to quit

How to tell this apart?
.quittable? method in the current_view
true -> 'q' quits
false -> 'q' is just a letter in a field

```
class FileDialogView
    def quittable?
        true
    end
end
```

Intermediate Ruby:

Rough Sketch of Input parsing:

```
def parse_input user_input
   case user_input
      when "\n"
         #change controller likely
         #check the View's current interaction
         #index to see what's next
      when "\e[A"
         #up button ... update the view with an
         #up action
      when "\e[B"
         #down
...
```

Intermediate Ruby:

Rough Sketch of Input parsing cont'd:

...

```
            when "\e[C"
                #right
            when "\e[D"
                #left
            else
                #send other input to a selected input field
            end
        end
```