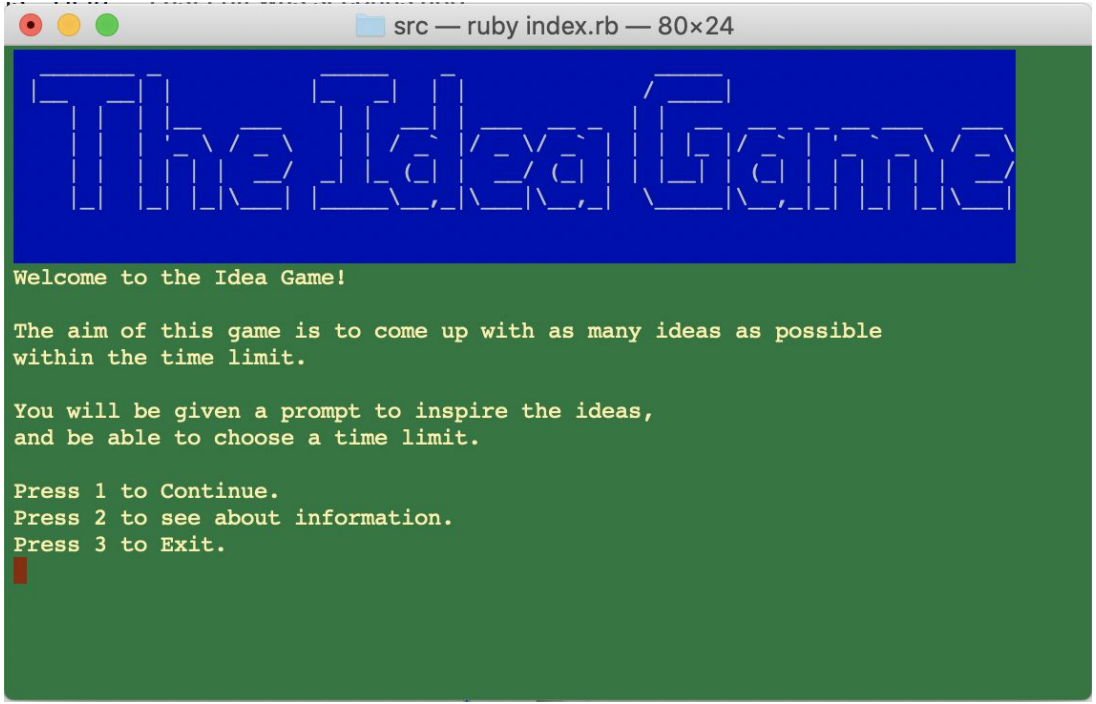
Terminal App

Tim Dunk

Overview

- The app is an idea generation game
- You select a time and a prompt category
- You are then given a random prompt, and must write as many ideas for that prompt as possible

Overview


src — ruby index.rb — 80×24
Welcome to the Idea Game!
The aim of this game is to come up with as many ideas as possible
within the time limit.
You will be given a prompt to inspire the ideas,
and be able to choose a time limit.
Press 1 to Continue.
Press 2 to see about information.
Press 3 to Exit.

Overview

```
Select a timer amount:
Press 1 for 30 seconds.
Press 2 for 1 minute.
Press 3 for 2 minutes.
Press 4 for 5 minutes.
1
You have selected 30 seconds.

Select a prompt category:
Press 1 for band names.
Press 2 for sketch ideas.
Press 3 for observations.
Press 4 for business ideas.
```

Overview

```
You have selected band names.
Come up with as many band names that contain the word Mouse as you can.
Go!
```

Overview

src — ruby index.rb — 80×24

```
You came up with 4 ideas! Good work!
This is what you came up with:
idea one
idea two
idea three
idea four

Press 1 to exit.
Press enter to play again.
```

Gems Utilised

- Colorize to colour the title screen
- Artii for the title screen
- Timers for game functionality

Issues With Timers

- The biggest problem I ran into during the creation of the app was that, because of how the ruby gets function works, Ruby will wait for the user to enter something before moving on.
- For the app, I needed to have a timer running unaffected by this

Issues With Timers - First Attempt

```
thirty_second_timer = timers.after(30) {still_time = false}
timers.wait
while still_time do
    ideas << gets.chomp
end
puts "time over!"
puts ideas
```

- Stopped here until timer finished

- By the time this was reached, it is no longer true

Issues With Timers

- I had a few other solutions that all felt very hacked together
- There were a few where I was able to get to the user input stage but the timer would not progress while there, or where the user could only input one input before the computer would stop listening for an input to push into the array

Multi-threaded Programming

- The regular flow of Ruby works sequentially down, executing pieces of code one after another, separately
- Multi-threaded programming is where you can have multiple pieces, or threads of code executing at the same time
- Think of a fork in the road, where the road is joined back together later

Timer Solution

```
def guessing(number, phrase)
    # ideas array is created
    ideas = []
    # timers is created for use with timers gem
    timers = Timers::Group.new
    # this will allow loop to be broken after timer expires
    still_time = true
    # puts the prompt from the last section
    puts phrase
    puts "Go!"
    # timer is created, when this expires it will turn still_time to false
    game_timer = timers.after(number) {still_time = false}
    # another timer is created to tell user that time's up
    game_timer2 = timers.after(number) {puts "", "Time's up! Press enter to continue."}
    # new thread is created so timers can run independently of gets
    timer_thread = Thread.new {timers.wait}
    # loop that will continue for more times than needed
    for num in 1..100 do
        # action conditional on the timer not having expired
        if still_time
            # gets is pushed into ideas array
            ideas << gets.chomp
        else
        end
    end
    # threads are joined back together
    timer_thread.join
```

Other Features

- Prompt randomizing

```
# random number for use in randomizing prompt given to user
prompt_num = Random.rand(5)
# user input to decide prompt category
prompt_input = gets.chomp.to_i
case prompt_input
when 1
    # array of prompts to be chosen from
    band_names = ["Snail", "Mouse", "Cold", "Valley", "Steep"]
    puts "You have selected band names."
    # sets prompt
    prompt = "Come up with as many band names that contain the word #{band_names[prompt_num]} as you can."
```

- I was proud of this working like I thought it would first try, I just had to look up syntax for generating a random number

What I Learned

- A big lesson in not procrastinating, the project took me longer than I originally thought it would because of the timer issue. Because I didn’t procrastinate, this wasn’t an issue
- I now have a good, very basic grasp of multi-threaded programming

What I Learned

- A lot of my general Ruby knowledge got cemented
- Shell scripting
- Programming is fun!

The End

Any questions?