Cameron Jones - Terminal App

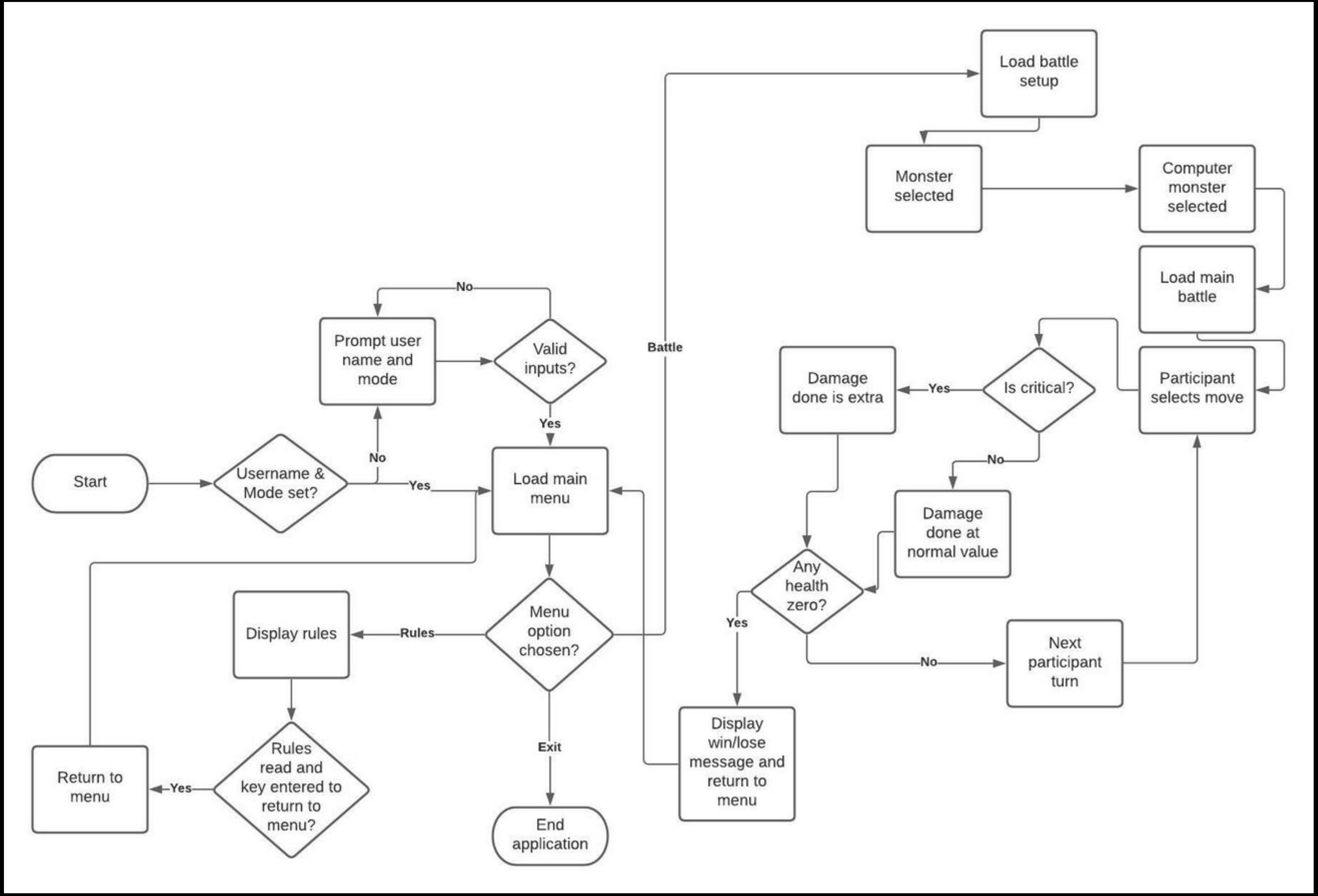
Contents

What will be presented today:

- WALK-THROUGH OF APPLICATION AND FEATURES
- CODE WALK-THROUGH
- BUILD PROCESS
- CHALLENGES / FAVOURITE PARTS

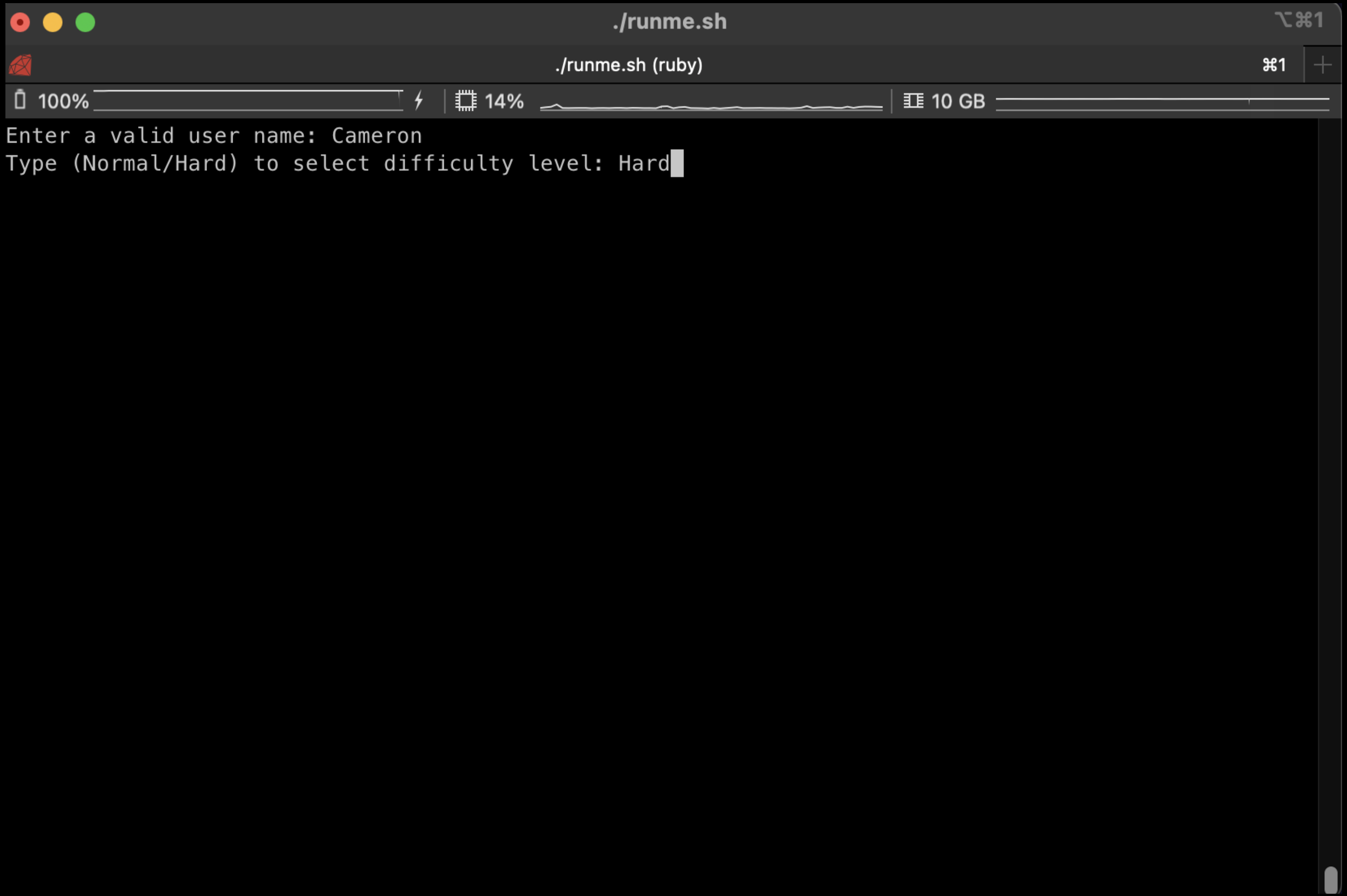
Flow Chart

- Shows the basic flow and logic of how the application will function through the use of looping and conditionals



Setup and Command line arguments

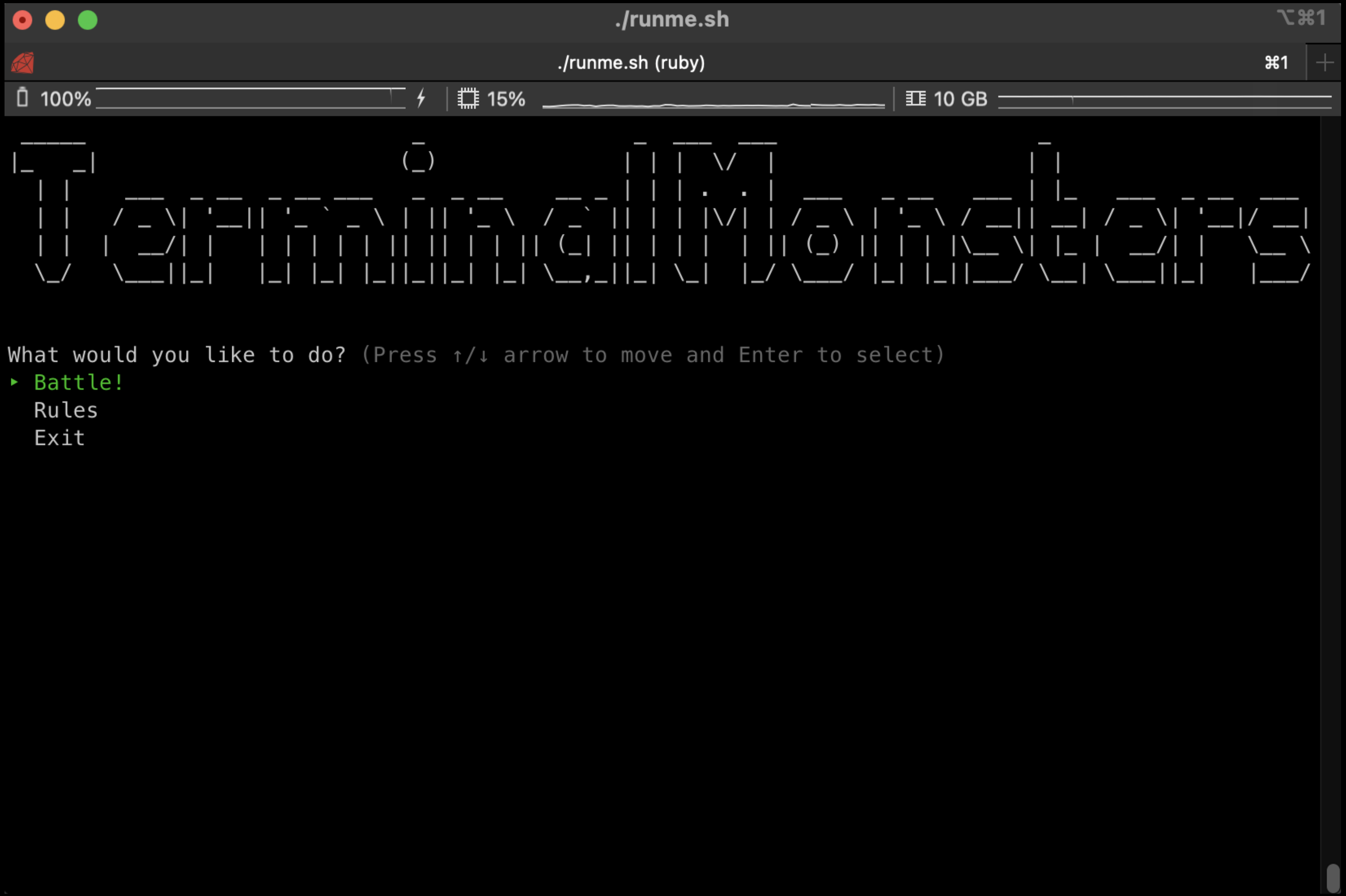
- Bash script is run which installs bundle and carries over command line arguments if they have been set
- If no arguments entered, user will be prompted to enter a username and difficulty mode
- Main menu will then commence loading



```
Enter a valid user name: Cameron
Type (Normal/Hard) to select difficulty level: Hard
```

Main menu

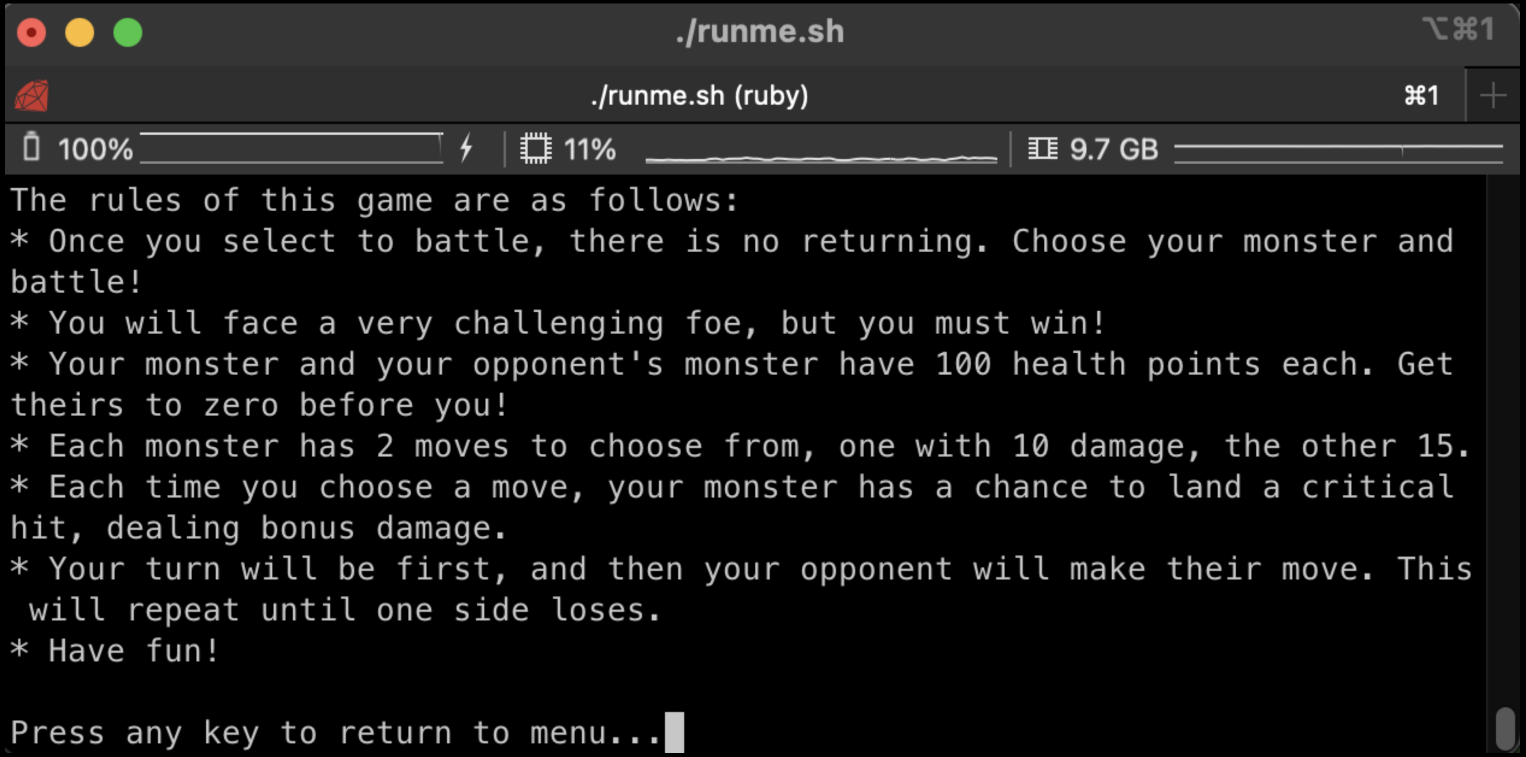
- User is presented with three options using TTY Prompt
- Battle will begin application main feature
- Rules will display how the game works
- Exit will end the application



```
 _____                         _                _ ___  ___                    _
|_   _|                       (_)              | ||  \/  |                   | |
  | |    ___  _ __  _ __ ___   _  _ __    __ _ | || .  . |  ___   _ __   ___ | |_   ___  _ __  ___
  | |   / _ \| '__|| '_ ` _ \ | || '_ \  / _` || || |\/| | / _ \ | '_ \ / __|| __| / _ \| '__|/ __|
  | |  |  __/| |   | | | | | || || | | || (_| || || |  | || (_) || | | |\__ \| |_ |  __/| |   \__ \
  \_/    \___||_|   |_| |_| |_||_||_| |_| \__,_||_|\_|  |_/ \___/ |_| |_||___/ \__| \___||_|   |___/

What would you like to do? (Press ↑/↓ arrow to move and Enter to select)
‣ Battle!
  Rules
  Exit
```

Rules

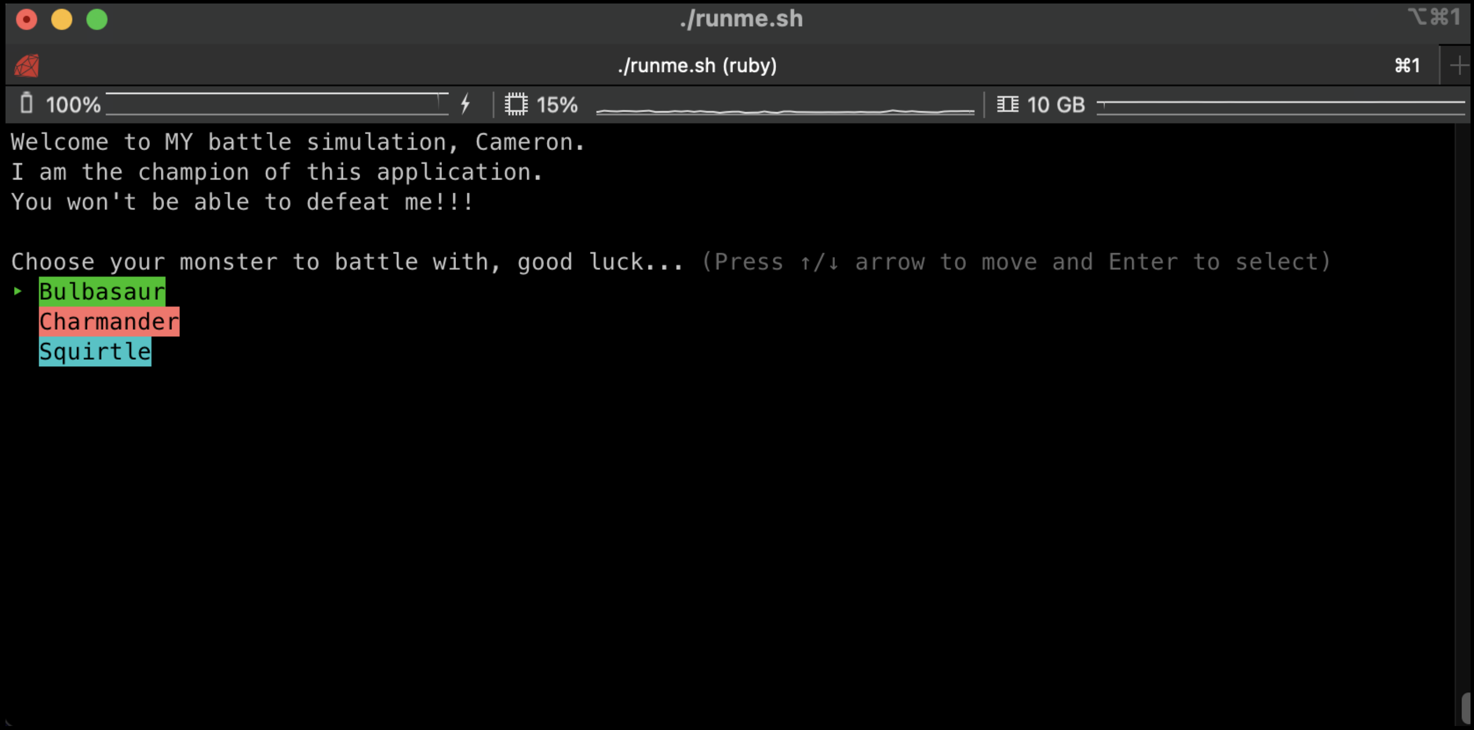
- Simple puts function with sleep timers informing the user the flow of the game
- Once all text has appeared on screen, user can press any key to return to the main menu



```
The rules of this game are as follows:
* Once you select to battle, there is no returning. Choose your monster and
battle!
* You will face a very challenging foe, but you must win!
* Your monster and your opponent's monster have 100 health points each. Get
theirs to zero before you!
* Each monster has 2 moves to choose from, one with 10 damage, the other 15.
* Each time you choose a move, your monster has a chance to land a critical
hit, dealing bonus damage.
* Your turn will be first, and then your opponent will make their move. This
 will repeat until one side loses.
* Have fun!

Press any key to return to menu...
```

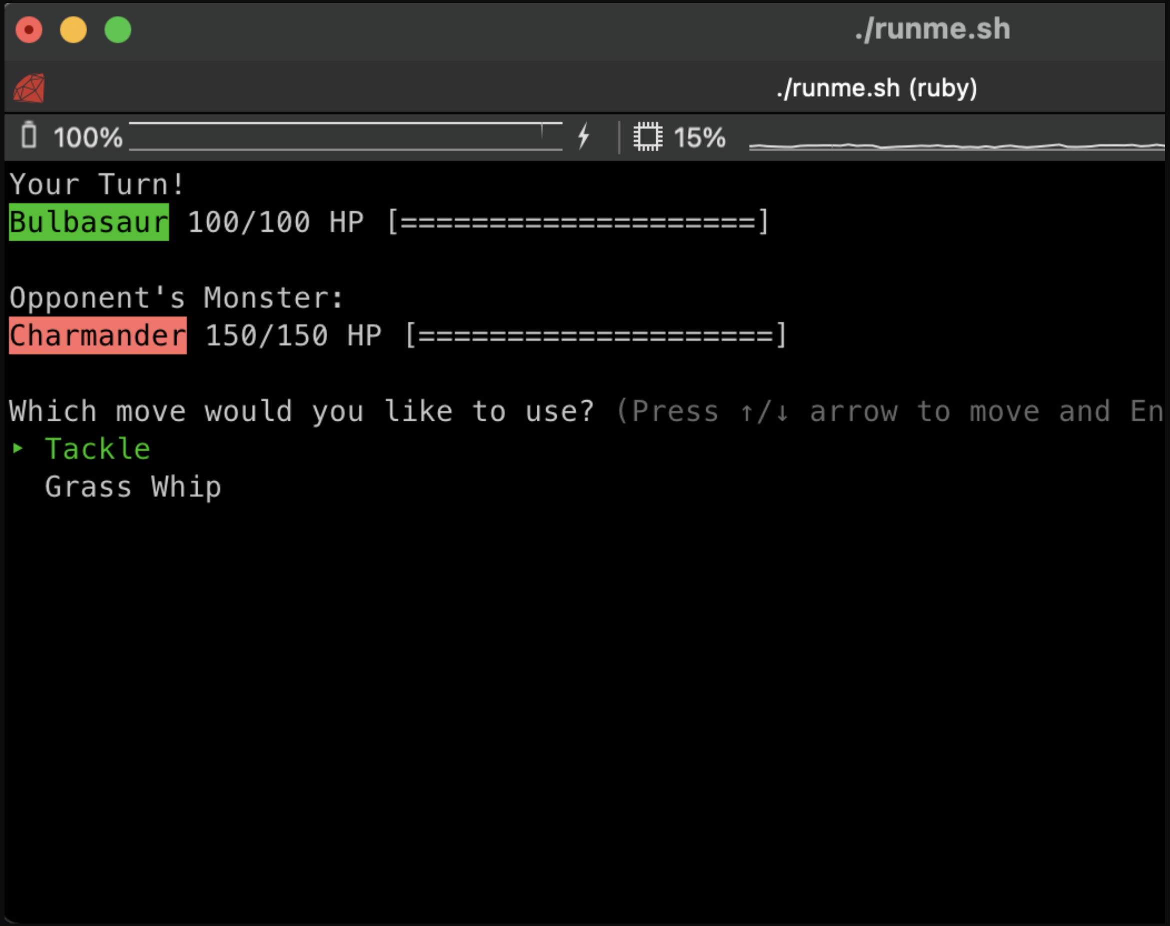
Battle Setup

- Begins with message from the computer saying they cannot be defeated
- User then chooses from the selection of monsters through TTY Prompt
- Computer is then assigned a monster at random and battle then loads



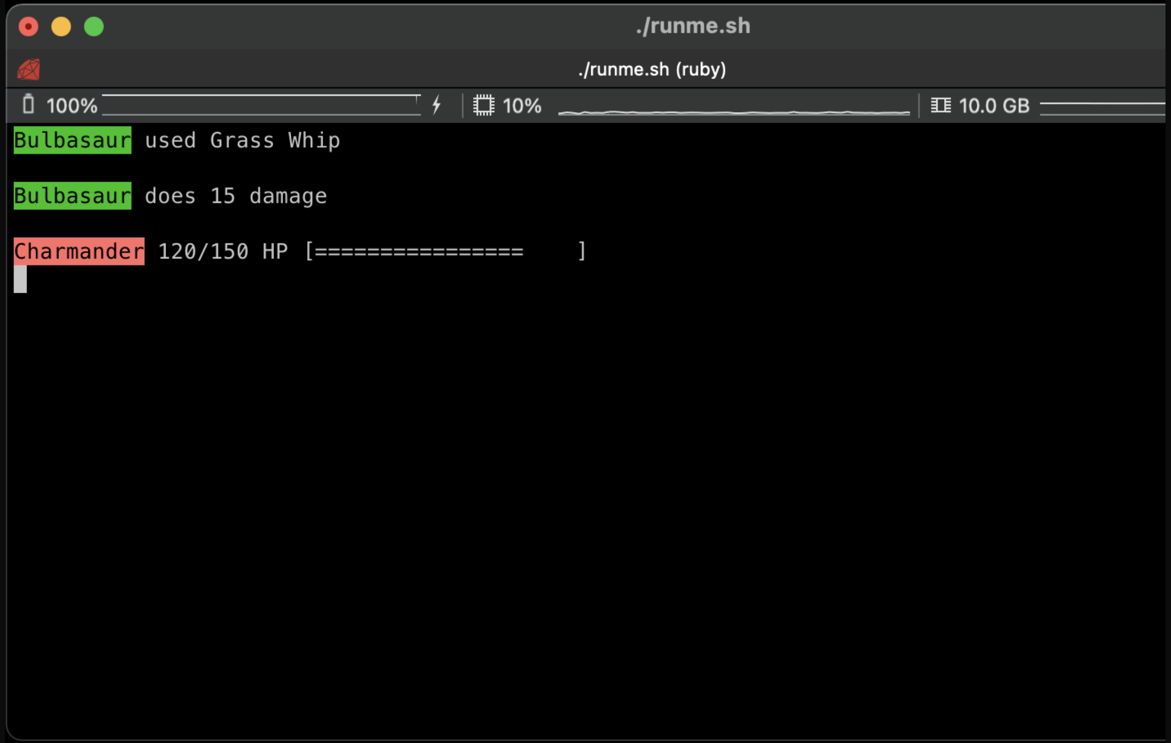
```
Welcome to MY battle simulation, Cameron.
I am the champion of this application.
You won't be able to defeat me!!!

Choose your monster to battle with, good luck... (Press ↑/↓ arrow to move and Enter to select)
‣ Bulbasaur
  Charmander
  Squirtle
```

Turns



```
Your Turn!
Bulbasaur 100/100 HP [====================]

Opponent's Monster:
Charmander 150/150 HP [====================]

Which move would you like to use? (Press ↑/↓ arrow to move and En
‣ Tackle
  Grass Whip
```

PLAYER TURN WITH HEALTH BARS AND PROMPT FOR SELECTING MOVES

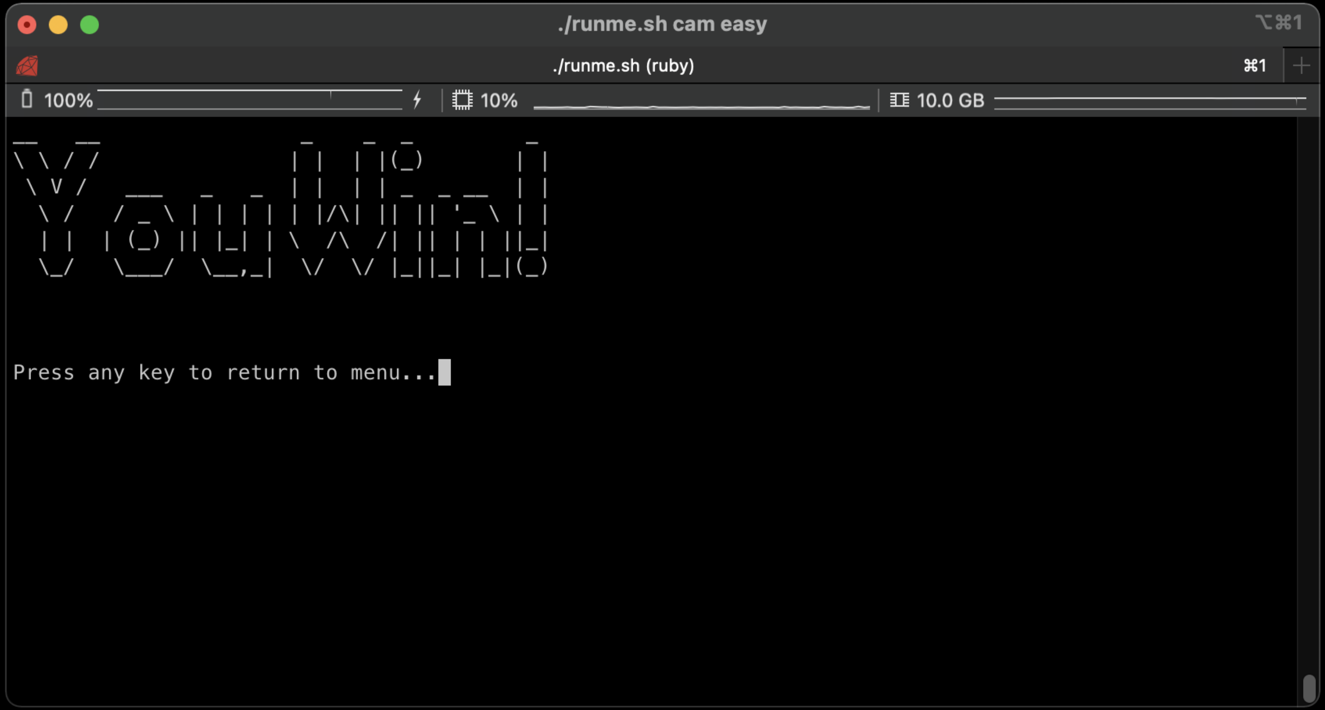


```
Bulbasaur used Grass Whip

Bulbasaur does 15 damage

Charmander 120/150 HP [================    ]
```

SHOWS MOVE SELECTED, DAMAGE DEALT, AND OPPONENT'S NEW HEALTH VALUE

Win / Lose Screen

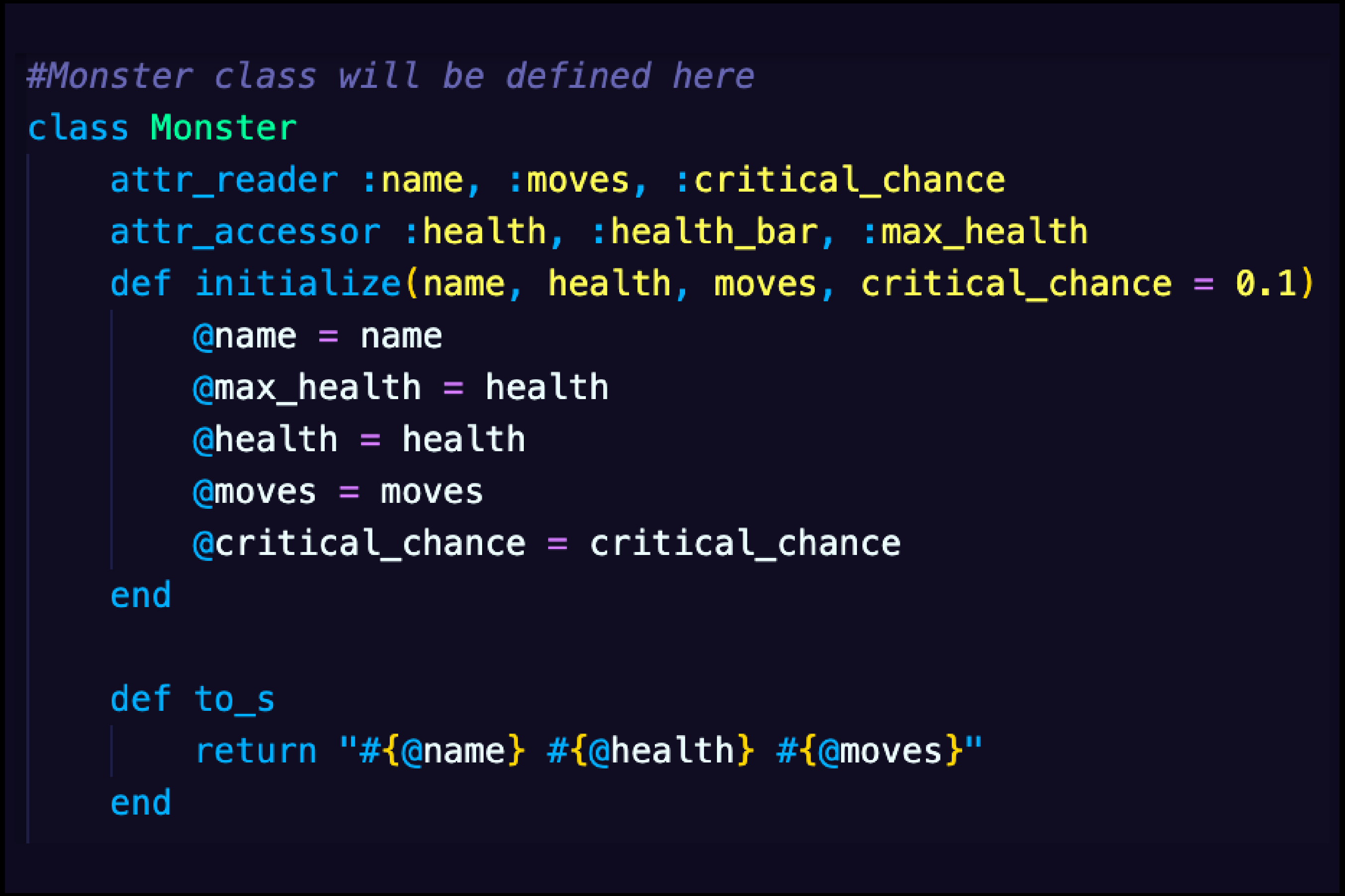
- Win or Lose message displayed depending on which health bar reaches zero first
- User then presses any key to return to main menu



```
__   __                _    _  _         _
\ \ / /               | |  | |(_)       | |
 \ V /   ___   _   _  | |  | | _  _ __  | |
  \ /   / _ \ | | | | | |/\| || || '_ \ | |
  | |  | (_) || |_| | \  /\  /| || | | ||_|
  \_/   \___/  \__,_|  \/  \/ |_||_| |_|(_)


Press any key to return to menu...
```

Monster Class

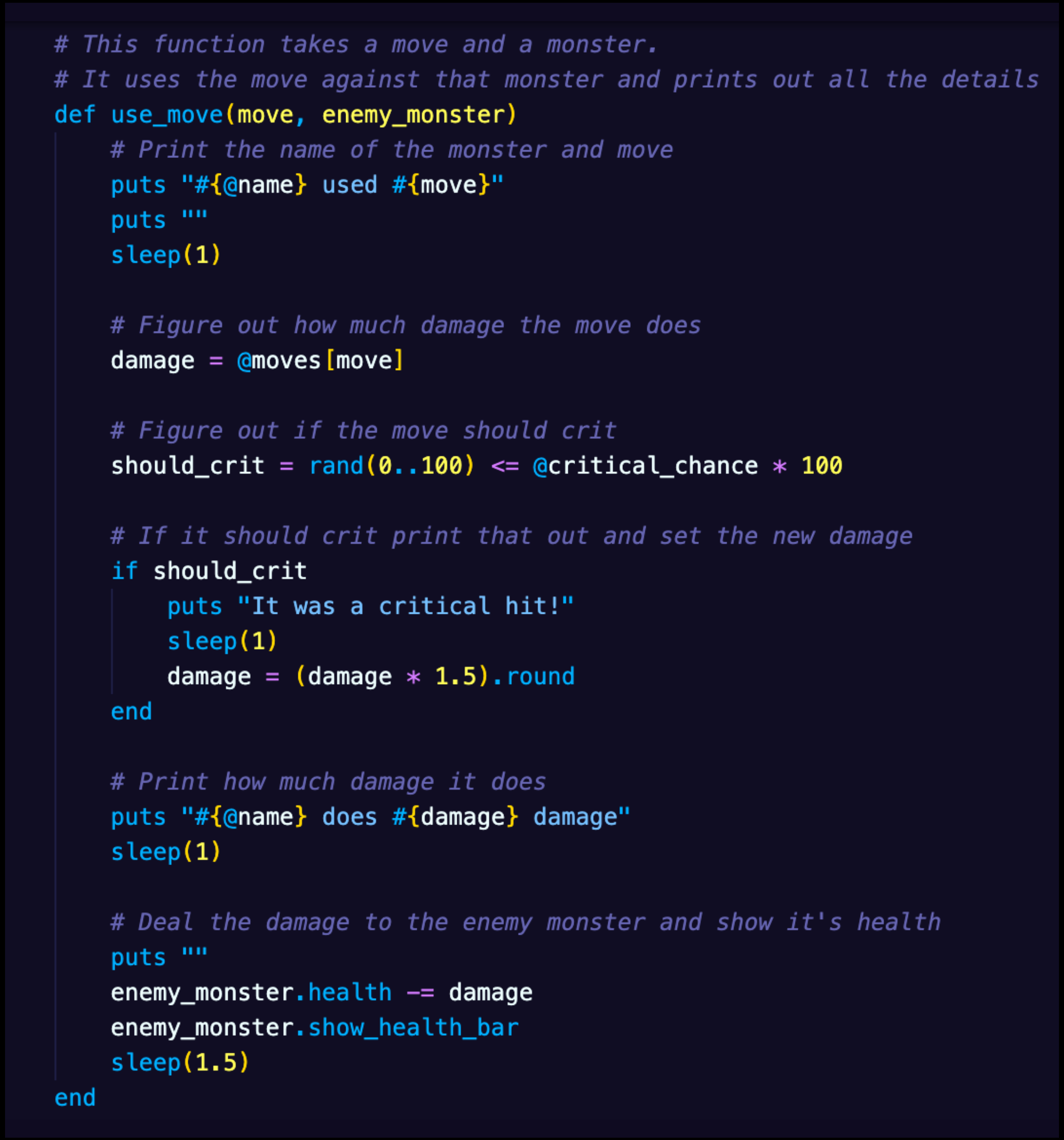
- Name, health, moves, crit chance attributes
- Max health attribute for hard mode health bars



```ruby
#Monster class will be defined here
class Monster
    attr_reader :name, :moves, :critical_chance
    attr_accessor :health, :health_bar, :max_health
    def initialize(name, health, moves, critical_chance = 0.1)
        @name = name
        @max_health = health
        @health = health
        @moves = moves
        @critical_chance = critical_chance
    end

    def to_s
        return "#{@name} #{@health} #{@moves}"
    end
```

Use move method

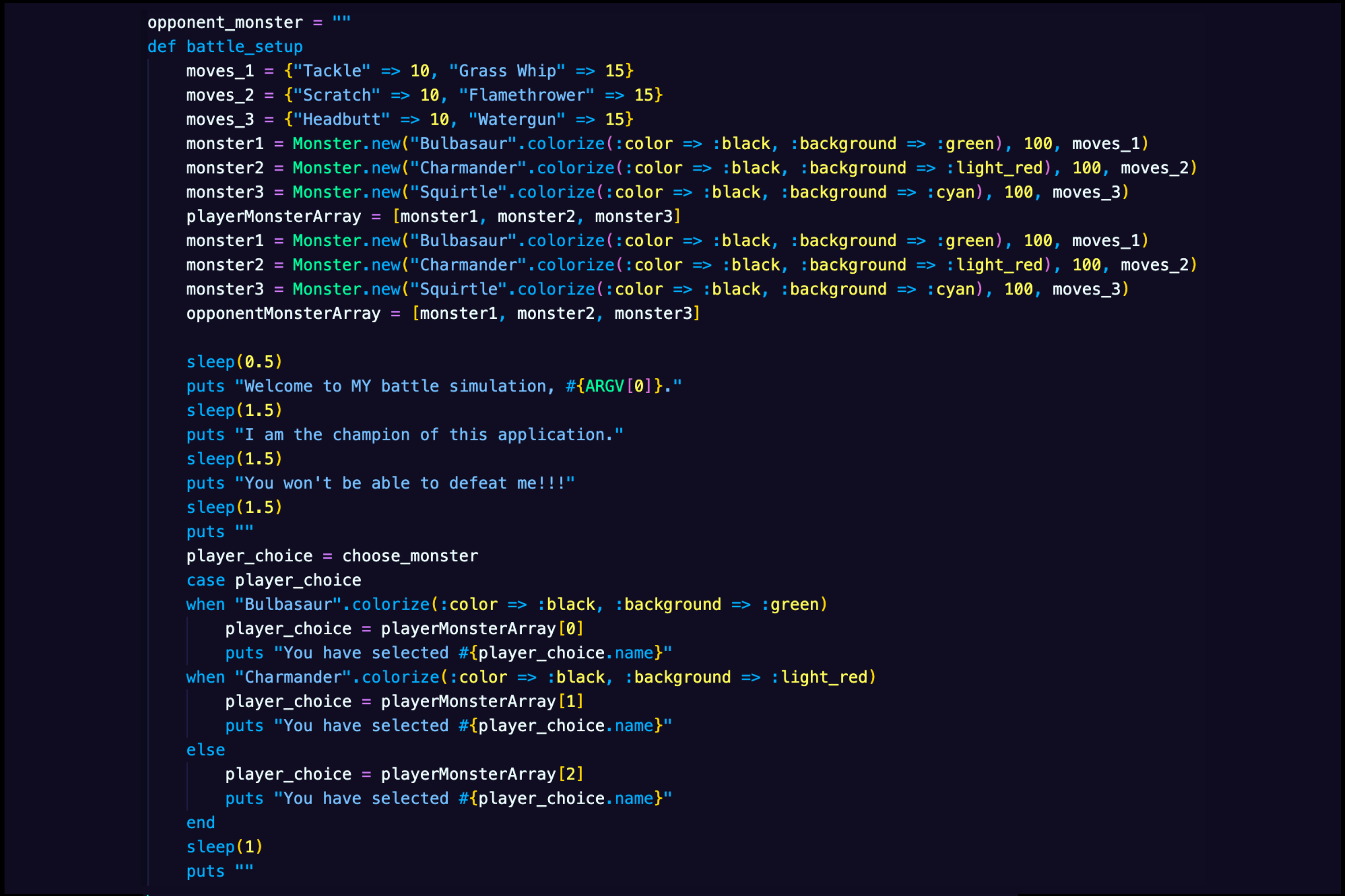
- Takes move selected from tty prompt by user, or randomly selected by computer during opponent turn and opposing monster depending on whose turn it is
- Random number generator checks is move will deal critical damage
- App then prints Move that was used, if critical it will display so
- Damage done is displayed
- Health of opposing side is updated and new health bar is displayed



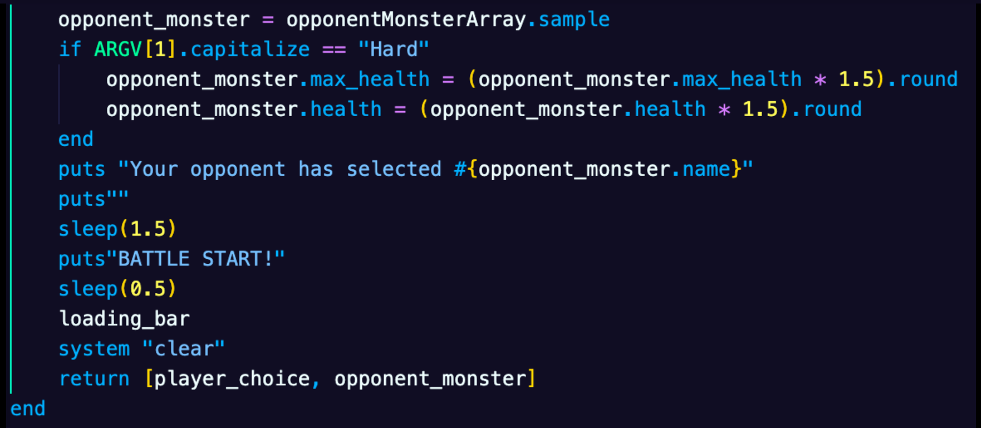
```ruby
# This function takes a move and a monster.
# It uses the move against that monster and prints out all the details
def use_move(move, enemy_monster)
    # Print the name of the monster and move
    puts "#{@name} used #{move}"
    puts ""
    sleep(1)

    # Figure out how much damage the move does
    damage = @moves[move]

    # Figure out if the move should crit
    should_crit = rand(0..100) <= @critical_chance * 100

    # If it should crit print that out and set the new damage
    if should_crit
        puts "It was a critical hit!"
        sleep(1)
        damage = (damage * 1.5).round
    end

    # Print how much damage it does
    puts "#{@name} does #{damage} damage"
    sleep(1)

    # Deal the damage to the enemy monster and show it's health
    puts ""
    enemy_monster.health -= damage
    enemy_monster.show_health_bar
    sleep(1.5)
end
```

Battle Setup

- Moves and Monster objects are defined
- Second monster array to deal with self damage if computer is assigned same monster
- TTY Prompt with colorize for monster options
- If hard difficulty selected, opponent monster health and max_health multiplied
- Player monster and opponent monster then returned to be called in main battle phase



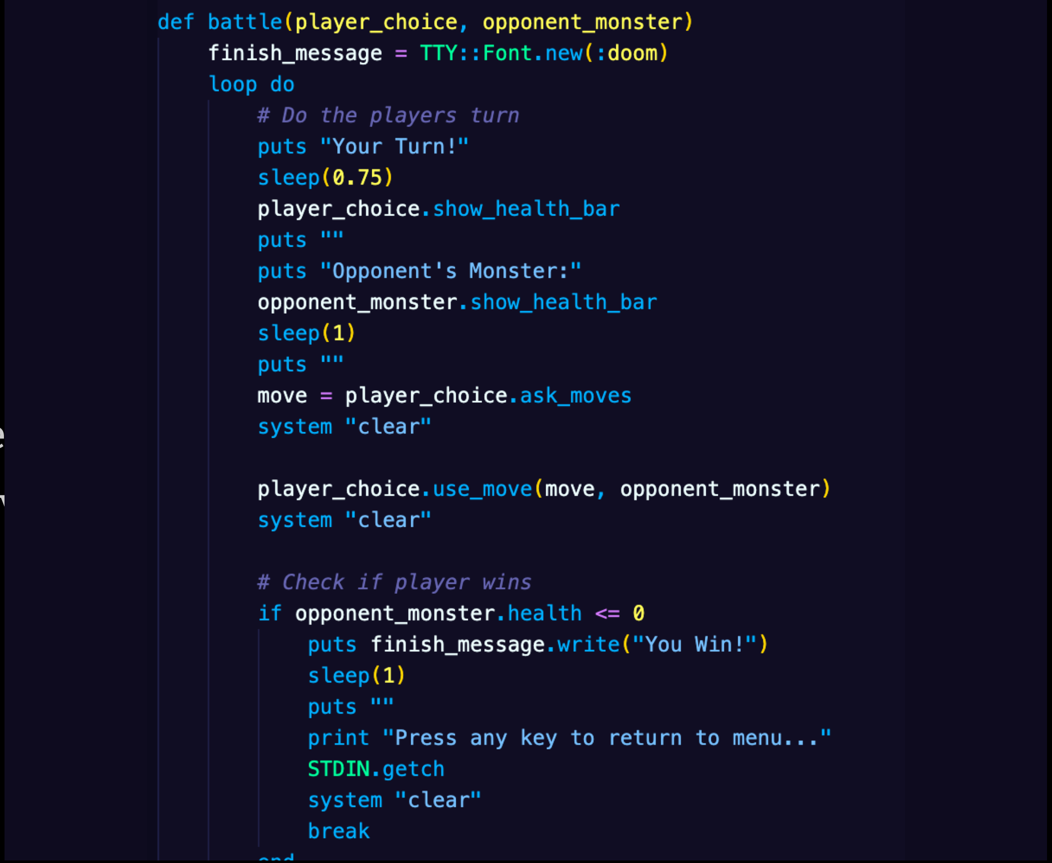
```ruby
opponent_monster = ""
def battle_setup
    moves_1 = {"Tackle" => 10, "Grass Whip" => 15}
    moves_2 = {"Scratch" => 10, "Flamethrower" => 15}
    moves_3 = {"Headbutt" => 10, "Watergun" => 15}
    monster1 = Monster.new("Bulbasaur".colorize(:color => :black, :background => :green), 100, moves_1)
    monster2 = Monster.new("Charmander".colorize(:color => :black, :background => :light_red), 100, moves_2)
    monster3 = Monster.new("Squirtle".colorize(:color => :black, :background => :cyan), 100, moves_3)
    playerMonsterArray = [monster1, monster2, monster3]
    monster1 = Monster.new("Bulbasaur".colorize(:color => :black, :background => :green), 100, moves_1)
    monster2 = Monster.new("Charmander".colorize(:color => :black, :background => :light_red), 100, moves_2)
    monster3 = Monster.new("Squirtle".colorize(:color => :black, :background => :cyan), 100, moves_3)
    opponentMonsterArray = [monster1, monster2, monster3]

    sleep(0.5)
    puts "Welcome to MY battle simulation, #{ARGV[0]}."
    sleep(1.5)
    puts "I am the champion of this application."
    sleep(1.5)
    puts "You won't be able to defeat me!!!"
    sleep(1.5)
    puts ""
    player_choice = choose_monster
    case player_choice
    when "Bulbasaur".colorize(:color => :black, :background => :green)
        player_choice = playerMonsterArray[0]
        puts "You have selected #{player_choice.name}"
    when "Charmander".colorize(:color => :black, :background => :light_red)
        player_choice = playerMonsterArray[1]
        puts "You have selected #{player_choice.name}"
    else
        player_choice = playerMonsterArray[2]
        puts "You have selected #{player_choice.name}"
    end
    sleep(1)
    puts ""
```



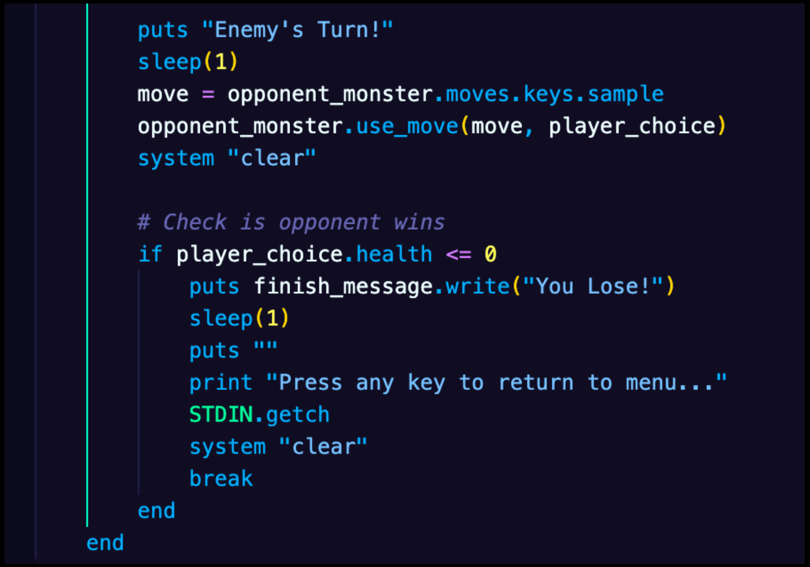
```ruby
    opponent_monster = opponentMonsterArray.sample
    if ARGV[1].capitalize == "Hard"
        opponent_monster.max_health = (opponent_monster.max_health * 1.5).round
        opponent_monster.health = (opponent_monster.health * 1.5).round
    end
    puts "Your opponent has selected #{opponent_monster.name}"
    puts""
    sleep(1.5)
    puts"BATTLE START!"
    sleep(0.5)
    loading_bar
    system "clear"
    return [player_choice, opponent_monster]
end
```

Main Battle

- Player turn first, utilises prompt to select a move
- use_move method then called with selected mo
  and opponent monster arguments
- After each participant turn, app checks if any
  monster health is zero or less and then displays
  or lose message



```ruby
def battle(player_choice, opponent_monster)
  finish_message = TTY::Font.new(:doom)
  loop do
    # Do the players turn
    puts "Your Turn!"
    sleep(0.75)
    player_choice.show_health_bar
    puts ""
    puts "Opponent's Monster:"
    opponent_monster.show_health_bar
    sleep(1)
    puts ""
    move = player_choice.ask_moves
    system "clear"

    player_choice.use_move(move, opponent_monster)
    system "clear"

    # Check if player wins
    if opponent_monster.health <= 0
      puts finish_message.write("You Win!")
      sleep(1)
      puts ""
      print "Press any key to return to menu..."
      STDIN.getch
      system "clear"
      break
```



```ruby
    puts "Enemy's Turn!"
    sleep(1)
    move = opponent_monster.moves.keys.sample
    opponent_monster.use_move(move, player_choice)
    system "clear"

    # Check is opponent wins
    if player_choice.health <= 0
      puts finish_message.write("You Lose!")
      sleep(1)
      puts ""
      print "Press any key to return to menu..."
      STDIN.getch
      system "clear"
      break
    end
  end
```

Build Process

1. Application Idea conception and figuring out general flow
2. Trello board and initial git repository made
3. Chose which gems I wanted to use for my application
4. Actual coding begins, starting with bash script and getting the command line arguments working how I designed was a challenge and I was getting errors but eventually fixed with conditional statements
5. Initialising process and main menu were then built
6. Monster class and initial methods created
7. Began coding battle function, after creating the first version I quickly realised it needed to be restructured due to variable scope being incorrect.
8. Built move selection methods and critical hit methods and then called them into main battle method to create the flow of battle and how it cycles between turns
9. Adding sprinkles with colour and debugging issues found after different functions had been built

Challenges and Favourite parts

Challenges

- Rebuilding monster selector to get battle working
- Command line arguments had issues with how they were being assigned depending on if entered in terminal or through error handling prompts
- Health bars were time consuming to understand

Favourite Parts

- The feeling of code not crashing
- Slowly seeing what I had been making piece together
- Figuring out a way to get through the challenges that were blocking my progress
- When the application worked without issue from start to finish for the first time

Thank you!