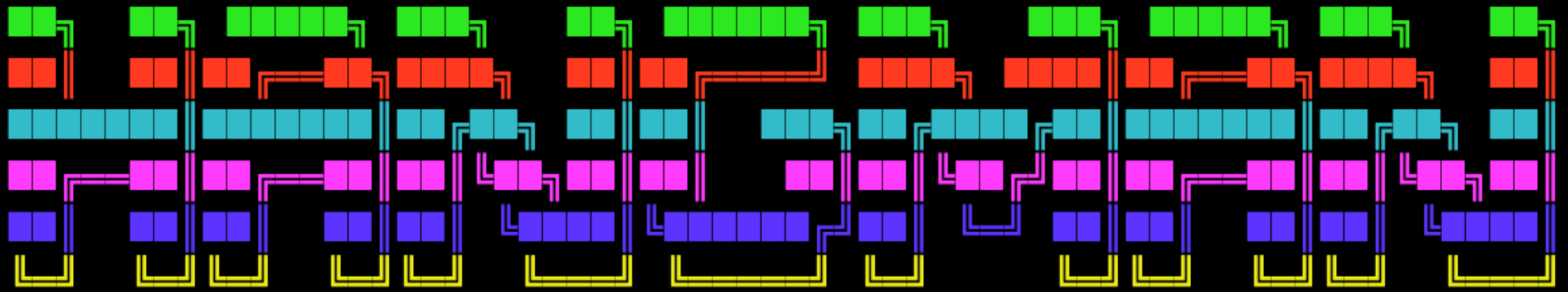
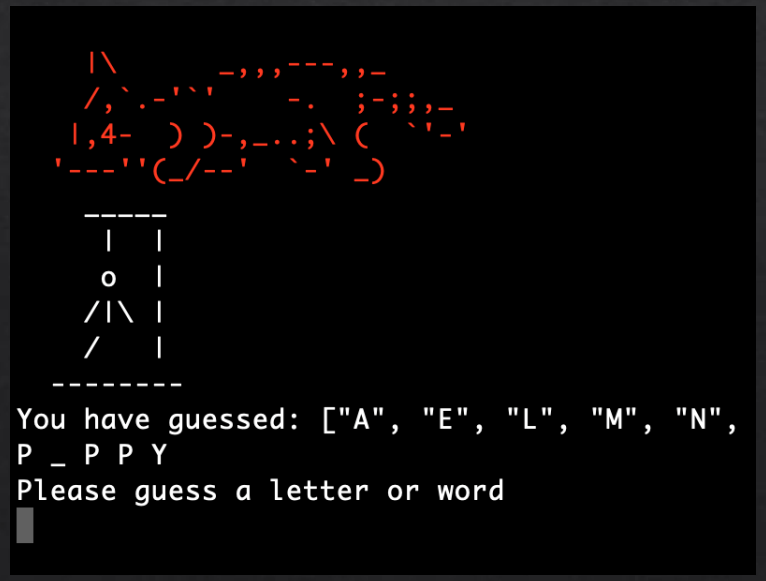
Welcome to...



...with cats



Walkthrough

- From the main menu the user can select single-player, multi-player or to exit.
- Upon selecting a game mode an instruction screen will appear relevant to the mode chosen.
- The word to be guessed is then either generated randomly using random-word-generator for single-player or chosen by one person in multi-player.
- The user then guesses letters or a full word until they run out of guesses or guess the word.
- A game over screen is displayed showing win or loss and user can choose to play again or exit.

Error Handling

Due to the amount of user input I had to develop some error handling. The below code checks the user input and ensures only letters are entered and the user is prompted to retry their entry until this condition is met.

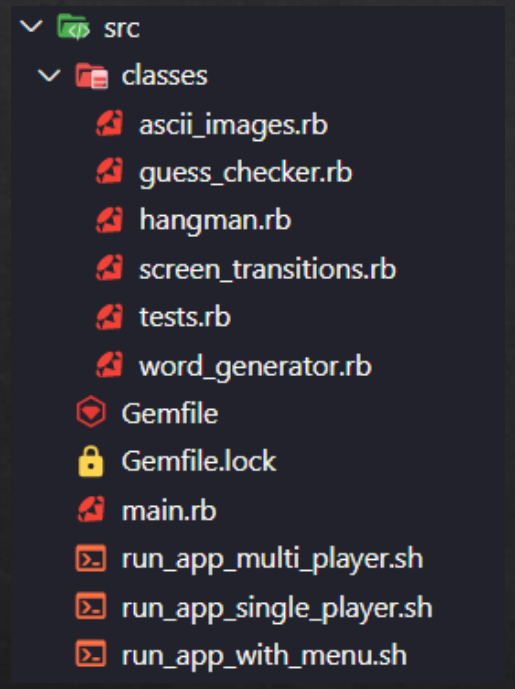
```ruby
begin
  system 'clear'
  puts "Hello Player 1!\nPlease enter a word:"
  @word = STDIN.gets.upcase.strip
  # If the word inputted above does not include numbers or special characters
  # then the word is updated and displayed back to the user.
  puts "You have selected '#{word}'."
  puts Images.menu_cat.colorize(:cyan)
  sleep 2
  # If the input contains any characters other than A-Z or a-z. Raises an
  # argument error if user input contains characters that are not allowed.
  if !@word.scan(/[^a-zA-Z]/).empty? || @word.empty?
    raise ArgumentError, 'Word has special characters or numbers'
  end
  # Displays a message advising entry is invalid and prompts the user to retry.
rescue ArgumentError
  puts "No funny business buddy! Let's avoid special characters and numbers please.\n"
  retry
end
```

Command Line Arguments

To cover the command line argument section of the rubric I decided to have three different game modes. One for the full game including menu if no arguments are entered, one to run a game with only single-player mode available if “single” is entered on the command line and one to run only multi-player mode if “multi” is entered on the command line.

```ruby
mode = ARGV[0]

# Displays the welcome screen and hides the cursor.
CURSOR.invisible do···
end

# Runs the game until the user chooses to exit.
loop do
  system 'clear'

  # Initialises the WordGenerator class so that a word can be either
  # randomly created in single-player mode or can be entered by
  # a user in multi-player mode.
  word = WordGenerator.new

  # Displays the menu and hides the cursor if no command line arguments
  # are entered.
  selection = CURSOR.invisible { Screens.display_menu } if ARGV.empty?

  # Launches the game mode depending on the selection made in the menu
  # or if command line arguments are entered when the game is launched.
  # Also functions to exit the game from the menu.
  if mode.to_s == 'single' || selection == 'Single-player'
    CURSOR.invisible { word.generate_word(1) }
  elsif mode.to_s == 'multi' || selection == 'Multi-player'
    word.generate_word('multi')
```

File Structure



- The classes folder includes all classes and modules called to run the game.
- This simplified adding in the amount of ASCII art that I used and also helped me keep related files together.
- The main application runs from the main.rb file and all gems, classes and modules are required here.
- I have also included my three executable files alongside this to install gems on running the game for a smoother user experience.

Executable Files

I created 3 executable files to demonstrate the command line arguments available in the code. These were made executable using the chmod command in the terminal and a variation of the below script in the .sh file.

```
#!/bin/bash

clear

echo "Welcome to Hangman"

echo "Ensuring all required gems are installed"

bundle install
gem update --system

echo "All gems have been installed, running application for the first time now"

ruby main.rb
```

Questions?

Thank you!!