BONUS

Sommaire

hangman-ascii-art

Objectives

You must follow the same [principle]https://lyon-ynov-campus.github.io/YTrack/subjects/hangman/hangman-classic/) as the first subject.

Ascii-art is a program which consists in receiving a string in the standar input and outputting the string in a graphic representation using ASCII. **Time to write big !!**.

What we mean by a graphic representation using ASCII, is to write the string received using ASCII characters, as you can see in the example below:

```
  _    _   _____   _        _        ____
 | |  | | |  ___| | |      | |      / __ \
 | |__| | | |__   | |      | |     | |  | |
 |  __  | |  __|  | |      | |     | |  | |
 | |  | | | |___  | |____  | |____ | |__| |
 |_|  |_| |_____| |______| |______|  \____/
```

Instructions

- Your project must be written in Go.
- This project should display letters in ASCII-Art from those following files
    - First you need to pass the file in argument containing the ascii letters as argument to your program (see example below).
    - You will then need to parse this file.
- You are free to design your own letter file !
- The code must respect the good practices.

Allowed packages

- Only the Standard Go packages are allowed.

Usage

```
./hangman words.txt --letterFile standard.txt

Good luck you have 10 attempts !


 _
| |
| |__                          ___
```

```
|  _ \                          / _ \
| | | |                        | (_) |
|_| |_|                         \___/

        ______   ______   ______
       |______| |______| |______|
```

Choose: E

```
 _
| |
| |__     ___                    ___
|  _ \   / _ \                  / _ \
| | | | |  __/                 | (_) |
|_| |_|  \___|                  \___/

              ______   ______
             |______| |______|
```

Choose: A

Not present in the word, 9 attempts remaining

```
 _
| |
| |__     ___                    ___
|  _ \   / _ \                  / _ \
| | | | |  __/                 | (_) |
|_| |_|  \___|                  \___/

              ______   ______
             |______| |______|
```

=========

Choose: L

```
 _             _   _
| |           | | | |
| |__     ___  | | | |   ___
|  _ \   / _ \ | | | |  / _ \
| | | | |  __/ | | | | | (_) |
|_| |_|  \___| |_| |_|  \___/
```

=========

```
Congrats !
```

This project will help you learn about :

- The Go file system(fs) API
- Ascii code manipulation
- Data manipulation

hangman-start-and-stop

Notions

- Golang Documentation: json
- Json encoding/decoding
- Golang Documentation: ioutil
- Read/Write file golang
- Go example: os Arg
- [Golang Documentation: flag] (https://pkg.go.dev/flag)

Objectives

You must follow the same principles as the first subject.

In hangman-start-and-stop have to create a way to stop the game and save the progress and restart at anytime to the point where you left the game.

You must:

- Implement a keyword `STOP` in the standard input.
    - It will stop and exit the game.
    - It will save the status of the game in a file `save.txt` . The data in the file must be encoded with `json.Marshal`
- Handle a **flag** `--startWith save.txt` in the command line, that allow you to launch the game with the file you saved with `STOP` command. The file will be decoded with `json.UnMarshal`

This project will help you learn about:

- Json format
- Encoding/Decoding structure to Json format
- Reading and Writing in files
- Argument handling in the command line

Allowed packages

- Only the standard go packages are allowed

Usage

```
 $> ./hangman words.txt
Good Luck, you have 10 attempts.
_ _ _ _ O

Choose: E
_ E _ _ O

Choose: A
Not present in the word, 9 attempts remaining









=========


Choose: STOP

Game Saved in save.txt.
```

```
 $> ./hangman --startWith save.txt
Welcome Back, you have 9 attempts remaining.
_ E _ _ O

Choose: L
_ E L L O

Choose: B
Not present in the word, 8 attempts remaining

      |
      |
      |
      |
      |
=========

Choose: H
H E L L O

Congrats !
```

Instructions

- The code must respect the good practices.
- We suggest you to search for the principles of a good website design.

Hard Mode

Objectives

You must follow the same principles as the first subject.

In hard-mode you have to create an harder mode for hangman.

You must:

- Implement a **flag** `--hard` that make the game harder :
  - Change the number of letter revealed at the beggining of the game `len(word) / 3 - 1`
  - Do not display the letter already use by the user (-1 points if a letter is reused)
  - You can submit only 3 vowels (-1 points if a vowels already submitted or the limit of 3 already reached)

Allowed packages

- Only the standard go packages are allowed