Hangman Project

Professor Don Colton

Brigham Young University Hawaii

1 Overview

We will create a working online hangman game. You can see a description on Wikipedia:

`http://en.wikipedia.org/wiki/Hangman_(game)`

An example game, written by Don Colton, can be played here:

`http://dc.is2.byuh.edu/hangman/`

2 Requirements

The resulting game should be fun to play. This aspect is not explicitly graded, but it should be kept in mind as you design, develop, and implement your game.

Your program must be written in Perl.

Your program must run as a CGI program on the web.

The URL for your program may be designated for you. If you, you must make your program work from that URL.

Every time your program runs, it should create a web page that clearly identifies you as the author and Hangman as the game.

When your program is first started, and after each game ends, it should attempt to start a new game. The player should have at least these two options: (1) let the program pick a word to be guessed, and (2) let the user enter his own word to be guessed.

When the user lets the program pick a word, the word should be randomly selected from a list within your program. The list must have at least ten different words to choose from.

When the game is under way, your program must display a graphic illustration (for example, a hangman's scaffold) showing how many times the player has made a mistake. The game starts with zero mistakes. When six mistakes have happened, the game is lost.

When the game is under way, your program must also display letters and dashes to indicate the letters that have been correctly guessed and the letters that have not yet been guessed.

When the game is under way, your program must also display the letters that were incorrectly guessed, in the order those guesses were made.

When six mistakes have been made, the game is lost, and your program must announce that the game is lost and reveal the hidden word. It must also attempt to start a new game as indicated above.

When all letters have been correctly guessed, your program must announce that the game is won. It must also attempt to start a new game as indicated above.

3 Suggestions

It is recommended but not required that there be a way for the player to end the game early. If the player takes this option, the hidden word whould be revealed and your program must attempt to start a new game.

It is recommended but not required that your program allow spaces to appear in the hidden word, thus making it a hidden phrase.

It is recommended but not required that your program have a list of words that are challenging to guess.

It is recommended but not required that your program allow the player to enter more than one letter at once. For example, the player could be allowed to enter the entire word or phrase.

It is recommended but not required that your program ignore guesses that have already been considered. Thus, if the player already guessed “e”, whether it was right or not, if they guess “e” again it should be ignored.

You may want to provide hints in the form of a list of the most common letters, maybe in order of popularity.

You may want to provide an alphabetical list of letters that have not already been guesses.

You may want to provide random taunts, telling the player how close they are to losing, or making fun of wrong choices. Similarly you may want to provide random statements of encouragement, telling the player how wise and intelligent they are for the good choices they have made.

You may want to use another graphic instead of a hangman’s gallows. Perhaps apples on a tree, as mentioned in Wikipedia. Perhaps time through an hour glass. Perhaps a fuse on a bomb. Think of something unique and fun.

4 Design

Your program must comply with indentation and other style requirements specified by the instructor.

The standard for indentation is that each level of nesting will have two or more spaces at the front of each line, and that the indentation will be uniform for all lines at that level of nesting.

Subroutines are to be placed after the main program. They should not be nested inside the main program or inside each other.

“`use strict`” must be invoked to eliminate accidentally global variables.

There should be a subroutine that will return a word to be guessed. This subroutine should be called when the user requests the program to provide such a word.

There should be a subroutine that will accept as input the target word and the list of letters that have been tried. It should return letters and dashes to indicate what the user should be shown.

There should be a subroutine that will accept as input the target word and the list of letters that have been tried. It should return a list of wrong letters, in the proper order.

There should be a subroutine that will accept input of your choice and will return a count of the number of errors that have been made.

All “state” information needed to continue the game should be included in the web page that your program puts up. Thus, many people should be able to simultaneously play your game without interfering with each other. It is not required that it be encrypted, but it must be hidden to the extent that a casual player cannot see it. If “view page source” reveals it, that is not a problem.