Introduction to Computer Science

Project 4: Hangman

Students will implement a Snap! version of the class word game "Hangman."

Overview

Hangman is a popular word game in which one player (the "chooser") chooses a secret word and another player (the "guesser") attempts to guess the word one letter at a time. If a guessed letter appears in the word, all instances of it are revealed. If not, the guesser loses a chance. If the guesser figures out the secret word before he or she runs out of chances, he or she wins. If not, the player who chose the word wins. Traditionally, chances are tracked using a stick figure drawing of a person being hanged from a gallows. The figure is drawn one body part at a time, and the guesser loses when the entire figure has been drawn. This game is also the basis for the TV game show Wheel of Fortune.

Details

1. **Behavior**
   1. **Gameplay**
      In our implementation of Hangman, the computer will take on the role of the "chooser" and the human player will be the "guesser." The computer will secretly choose a word from a list (see below) and show the player how many letters are in the word by displaying a sequence of blanks (underscores). Then, the computer will begin asking for guesses. If the player guesses a letter that is in the secret word, all blanks representing an instance of that letter should be replaced by the letter. If the guessed letter is not in the word at all, the player should lose a chance and a new part of the Hangman figure should appear. If the player guesses a letter he or she has already guessed, he or she should not lose a chance, even if that letter is not in the word. If the player guesses all letters in the word, he or she wins. If the Hangman figure is completed, the player loses. In either case, the secret word should be revealed after the game is over.

   2. **Sprites**
      Your game will need to include at least three sprites: the Hangman itself, a "host" sprite that asks the player for a guess and informs him or her whether or not the guess is correct, and an "assistant" sprite that tells the player the status of the secret word. You may use more sprites if you think they are appropriate. The host and assistant should have clear roles and should never do each other's job.

   3. **Word Status**
      As the game is played, the player should be shown the current guessed status of the secret word. Letters that have been correctly guessed should be shown in the correct locations. Unguessed letters will appear as blanks. At the beginning of the game, no letters will have been guessed, and the only information shown to the player will be a sequence of blanks, with one blank for each letter in the secret word. As the player guesses letters correctly, blanks

representing guessed letters should be replaced by those letters. So, for example, if the secret word is "screwdriver" and the player has guessed 'e,' 's', 'r', and 'd,' the current word status would be "s *r e* d r e r".

iv. **Chances/The Hangman**

The player will have six "chances" to guess the word. Guessing a correct letter does not cost a chance. Each missed chance will cause a new piece of the Hangman to appear. The six pieces of the Hangman are: head, body, left arm, right arm, left leg, right leg. You may use a stick figure for your Hangman, but if you would like to be more creative with the appearance, feel free to do so. No matter what your hangman looks like, though, it should include these six pieces and no more.

v. **Game End**

The game can end in one of two ways:

```
* If the player has guessed the complete secret word, he or she wins. * Otherwise, if the player has run out of chances and the complete Hangman has been drawn, the player loses.<br/> <br/>
>
```

In either case, when the game ends the host should stop asking for guesses. The host should inform the player whether he or she won or lost, and the assistant should reveal the entire secret word.

2. **Implementation Details**

i. **Word List/Secret Word**

You will be provided with a list of words from which the secret word should be chosen for each game. You will be shown in class how to import this list into your program. At the start of each game, a word should be randomly chosen from this list to be used as the secret word. The secret word must be chosen randomly, and must be a word in the list.

ii. **Documentation**

In addition to functioning well, your program must be well-documented and readable. This includes, but is not limited to, things such as:

```
* organizing your scripts so that they can be read and comprehended easily * giving  your sprite
s meaningful names * naming and using your variables, lists, and custom blocks well * including
comments to describe the structure of your program and any particularly complex or unintutiv
e pieces of code
```

iii. **Required Snap! Elements**

Your program must include, at a minimum, the following Snap! code elements:

```
* At least two lists, once of which must be used to track guessed letters * Custom blocks as app
ropriate, including arguments and reporters
```

3. **Required Checkpoints**

    i. Be able to select a secret word, keep track of which letters have been guessed, determine if each letter guessed is in the secret word or not
    ii. Be able to announce the current status of the word, showing letters that have been guessed and blanks for other letters.
    iii. Be able to play a full game of Hangman, identify correct and incorrect letters, display the hangman figure, and inform the player whether they have won or lost.

Grading Scheme/Rubric

Functional Correctness (Behavior)	
Computer randomly chooses a secret word	1 point
Host repeatedly asks for a letter and announces whether that letter is in the secret word	2 points
Assistant displays the correct secret word status after each guess	4 points
Player loses a chance and a piece of the Hangman appears when a guess is incorrect	3 points
Host informs player when he or she guesses a letter that has already been guessed; player does not lose a chance	2 points
Game ends with player victory if the entire secret word is guessed	2 point
Game ends with player defeat if the player runs out of chances	2 point
Secret word is revealed when game ends	1 points
Total	**17 points**
Technical Correctness (Implementation)	
Program is well-designed visually and has a consistent theme	2 point
Program is well-documented and exhibits good style	2 points

Functional Correctness (Behavior)	
Program shows good creativity and effort	3 points
Program includes at least two lists	2 points
Program uses custom blocks with arguments and reporters appropriately	2 points
Program tracks guessed letters using a list	2 points
Obtain and respond to playtest feedback from a parent or guardian	2 points
Checkpoint 1 (4/30)	4 points
Checkpoint 2 (4/30)	4 points
Total	*19 points*
Total	***40 points***