Game of Hangman

In other computer versions of this game, the computer asks the user to guess a word by picking letters. Since there is no easy way to store a large number of words in a Scratch program, we will create a 2-player game in which the first user enters the word and the second user guesses it. The flow of the game should go as follows:

- At the start of the game, ask the first player to enter the word.
- Let the second player know how many letters the word contains.
- Ask the player to supply one guess (i.e. letter) per round.
- The player should receive feedback immediately after each guess about whether their guess appears in the computer's word.
- After each round, the player sees the partially guessed word so far.

Some additional rules of the game:

- A user is allowed 5 guesses. The hangman figure reminds the user of how many guesses s/he has left after each round.
- A user loses a guess only when s/he guesses incorrectly.
- If the user guesses the same letter twice, no penalty is charged.
- The game ends when the user constructs the full word or runs out of guesses. If the player runs out of guesses (s/he "loses"), the word is revealed to the user.

CS and Scratch Concepts:

- Algorithms
- Boolean operators (and, or, not)
- Conditions: YES/NO questions
- Conditionals (If-Else)
- Conditionals (IF)
- Conditionals (Wait until)
- Conditionals (nested IF)
- Costumes
- Data structures – list
- Events
- Looping - simple (repeat, forever)
- Looping - conditional (repeat until)
- Procedures
- Procedures with inputs
- Procedures with return value
- Random numbers
- Random numbers - mapping to a set of things
- Recursion

- Relational operators (==, <, >, !=, <=, >=)
- Sequence
- Stopping scripts
- String operations (join, letter, length of)
- Synchronization using broadcasting
- User events (keyboard)
- User events (keyboard - polling)
- User text input
- User input (buttons)
- User input validation
- Variables - numbers
- Variables - strings
- Variables - properties (built-in)
- Variables - local/global scope

Data Structures:

We will store letters guessed by the user in a list called "lettersGuessed".

We will store the actual word also in a list called "secretWord". The reason to use a list instead of a string is because we can quickly check if a given letter is part of the word by using the "contains" operator.

Control flow:

This is how the overall program will be:

- The main script (when a new word is started) will perform the following tasks:
    - Get the secret word from player 1 and save it in a list using "Save Secret Word"
    - Read a key from the user:
        - Add keyed letter to "lettersGuessed" unless it is already there
        - Show all letters guessed so far using "Show Letters Guessed".
        - If keyed letter is a "good" guess, (i.e. it is part of "secretWord"):
            - Build the "guessed word" from all letters so far using "Construct Guessed Word".
            - Check if all letters of the secret word have been guessed using "IsWordGuessed":
                - If True, declare win and stop the game.
        - Else (i.e. if keyed letter is a "bad" guess), change hangman's costume to the next one. If no more attempts left, declare failure and stop the game.
        - Read a key again …

Sprites:

"Hangman": This sprite will indicate the current state of the game. Using costumes the game will complete the stick figure.

"Word" and "Attempted" will respectively show the current state of the guessed word and list of letters entered thus far.

"New word" button will allow to start a new session.

A "Logic" sprite will interact with the user and contain all the logic of the game.

Algorithms:

Note: In Scratch, the list parameters passed will actually be global variables. The return value will also be a global variable.

Algorithm to read keys:

One way is to use the ASK command for every new key. This approach is certainly workable but boring because the user has to press ENTER in addition to the key.

The other way is to allow the user to only press a key. We can achieve this using the "key pressed" condition. We will need a long list of IF commands as shown below:

```
Repeat until game over:
      If key "a" pressed:
            Check Key "a"
      If key "b" pressed:
            Check Key "b"
      If key "c" pressed:
            Check Key "c"
      And so on …
End repeat
```

Algorithm Check Key

Process every key entered by the user.

```
Input:
Character c: key entered by user
Procedure:
This is already explained in the control flow described above.
```

Algorithm Is Word Guessed

Determine if the word has been completely guessed.

```
Input:
List secretWord: the original secret word the user is trying to guess
```

```
List lettersGuessed: which letters have been attempted so far (both
right and wrong)
Output:
True if lettersGuessed contains all the letters of secretWord;
False otherwise
Procedure:
For every character c in secretWord:
      If c is not in lettersGuessed:
            Return False
      End if
End for
Return True
```

Algorithm Get Letters Guessed

Converts the list "lettersGuessed" to string format.

```
Input:
List lettersGuessed: which letters have been attempted so far
Output:
String "guessed"
Procedure:
guessed = ""
For every character c in (lettersGuessed):
      Append c to guessed
      Return guessed
```

Algorithm Save Secret Word

Convert the secret word to list format.

```
Input:
String word: the word to be guessed
Output:
List secretWord
secretWord = []
For every character c in word:
      Append c to secretWord
```

Algorithm Construct Guessed Word

Construct, in string format, the current state of the word as it has been guessed. Use "-" where the letter has not yet been guessed.

```
Input:
List secretWord: the word the user is trying to guess
List lettersGuessed: what letters have been guessed so far
Output:
String comprised of letters and underscores that represents the
current state of the word as it has been guessed thus far.
Procedure:
guess = ""
For every character c in (secretWord):
      If c in lettersGuessed:
```

```
            Append c to guess
      Else:
            Append "_" to guess
      End if
Return guess
```

Solutions:

hangman-final.sb2
https://scratch.mit.edu/projects/323678799/

Author: Abhay B. Joshi (abjoshi@yahoo.com)
Last updated: 9 August 2019