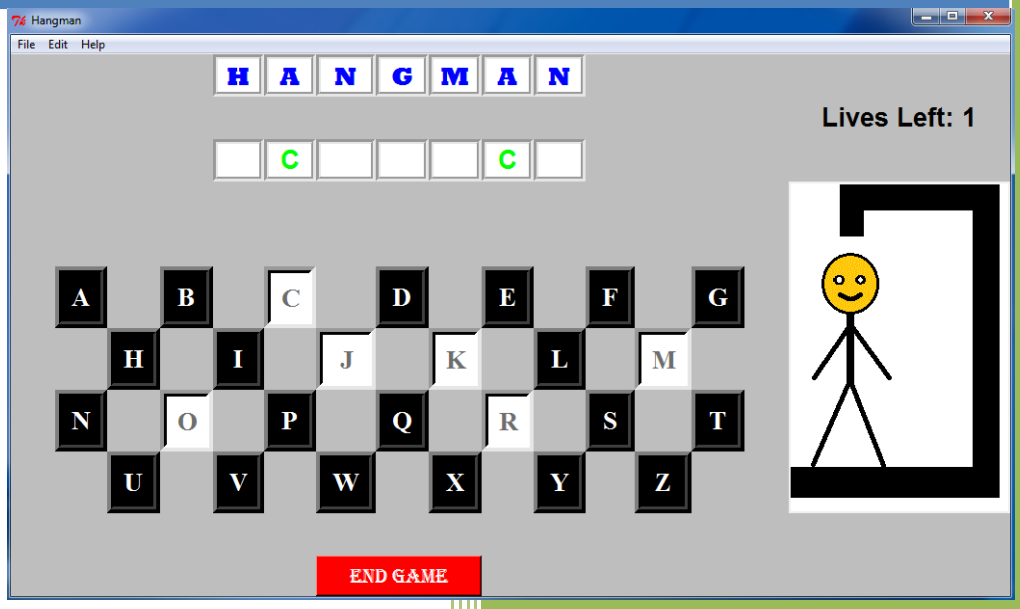
Fall 2012
CSCI 230 – Computing I

# Final Project Report




Naeem Tai

Indiana University Purdue University Indianapolis

12/11/2012

**Program Name:** Hangman
**Program Category:** Graphic User Interface (GUI) based Game
**Programming Language Used:** Python
**Libraries Used:** Tkinter, Random, tkMessageBox
**Concepts Used:** Object Oriented Programming, Arrays, String Manipulation, ASCII, "lambda" Function, File Manipulation, Window Configuration
**Files:** hangman.py, hangmanExtra.py, word.txt, hangman1.gif, hangman2.gif, hangman3.gif, hangman4.gif, hangman5.gif, hangman6.gif

**Documentation**
This program is a standard hangman game that involves evaluating a word through guessing letters. Besides normal behavior of this single player game, additional features of the game include: ability to add a new word to existing file, see about the game, and help on how to play. Some sort of pointing device (i.e. mouse, touchpad, etc.) and an input device (i.e. keyboard) is required in order to play this game to its fully extent.
Menu system on the top of the window allows users to access different features of the game. Whether it is on a big screen or a small screen, the font sizes and the widgets in the game window automatically adjust themselves with the environment in order to preserve visual consistency. Blanks of the word are always centered with rest of the widgets. Cursor icon changes while hovering over alphabet letters to indicate that they are a source of user input. Users will always be aware of the progress, whether they are playing a game or adding a new word to the file.

**Data Plan**
This program deals with data in couple of different ways. Getting a new word or adding a new word involves a file system. Most of the widgets are stored in arrays. String slicing is used to separate each letter from a word, and then stored as an array of letters.
On every new game, the game is reset. A random word selected from the existing file and converted to upper case for ease of manipulation with ASCII character codes. Only alphabets are manipulated as blanks; spaces, numbers, and special characters are displayed as they are. Upon selection of a random word, each character of the word is manipulated separately using ASCII character codes, and stored in an array. An array for answer contains all the characters of the word, while an array for an answer only contains non-alphabetical characters. After this manipulation, blanks are displayed on the screen.
Alphabets are displayed on the screen in a tile layout using mathematical algorithm. Each alphabet is a part of an array, and has the same method as other alphabets but with different parameter. When clicking on any alphabet, the letter clicked is checked with the word. If matched, it is put in appropriates blank(s); if no match found, it results in decrement of remaining lives and canvas is updated with an appropriate image. On every match found, the answer array is updated with the matched letter in appropriate index. This process is repeated on every letter guessed until the question array matches the answer array, resulting in a win, or remaining lives decrements to zero, resulting in a loss. Clicking on the end game button is also the same as bringing the game to a loss. Losing a game will also display the correct answer under the question blanks.
Accessing any additional feature from the menu will open a top window that has a scope limited to itself. When adding a new word, the user input is converted to upper case and validated using

several conditions: must contain at least one alphabet, must not exceed fifteen characters in length, and be different then the words already exists in the file. Users will be prompted through error or warning message upon failure to pass any of these validation tests. If the user input passes validation test, it will be appended to the file and the user will be notified with a message. About and How to Play are read only top windows that allows users to learn more about the game itself.
All these procedures are contained within one python file. hangman.py contains all the features of the game except for font resizing upon window resizing. Font resizing is a part of hangmanExtra.py along with all other features. Two python files, text file, and images are saved in the same root directory.

**Visual Layout**
Visual layout of this game very consistent throughout the game. Grid is configured with each row given a weight equal to number of rows and each column is given a weight equal to number of columns. Every widget on the screen has sticky attribute in all four directions in order to maintain visual layout upon resizing the game window. Font sizes are also changed according to change in the size of the window.
Game title is centered on the top of the screen, and alphabets are displayed in a tile layout. At the beginning heading, alphabets, and end game button are displayed on the screen. There are no blanks displayed and visible widgets are disabled, indicating that there is currently no game that is running. When user starts a new game File menu, appropriate number of blanks are displayed on the screen. Depending on the length of the words, blanks are always centered on the screen. Alphabets are also enabled with change in colors. Hovering over alphabets also causes a change in cursor icon to indicate source of input. Game progress is displayed on the right end of the file with remaining lives, loss or win indicator, and canvas to display image on each stage of the game. When clicking on any alphabet, that letter will be put in an appropriate blank if it matches. Game progress is updated upon winning, losing, or guessing of a wrong letter. Also clicking on an alphabet disables that alphabet with change in color so that it cannot be clicked again. Upon winning or losing, game progress will display appropriate text and images, and alphabets and end game button will be disabled. Also upon losing, correct word will be displayed underneath the question blanks.
Menu system on top of the window will allow users to access additional features of the game. Selection of each menu item will open a top window with an exception of Exit that will end the program.

**Acknowledgements**
- Andy Harris for help with lambda function
- http://www.tutorialspoint.com/python/python_gui_programming.htm for help with Python Tkinter