Jasque Saydyk

Professor Vanderberg, Swenson, and Trice

CS 126L Section 2

6 May 2017

Lab 13 - Hangman

1. Problem Statement

For this lab, I was tasked with making a Hangman game using Tkinter and Python Turtle. To do this, I created 3 separate files, Hangman, which contains the game's logic, HangmanGUI, which contains methods to manipulate the GUI, and HangmanMain, which runs the actual game. All minor details can be found in the lab specification, but the game should operate as a normal Hangman game that informs you how much of the word you have completed, how much of the man is hanged, and what letters were already used to do the game. Also, the game should be able to handle errors inputted by the user.

2. Planning

For the Hangman class, I determined that there should be a readWords() method that transfers the words from a given filehandle to a list structure. There should also be a chooseWord() method that picks a random word from the list to be the word for the game. I determined that there should be a inputLetter() method that determines if the given letter fits in the word or not, is a repeat, or is bad input. To help create this method, I also created determineLetterMatch(), determineLetterRepeat(), and updateLetterInWord() to stay true to the "One task per method" principle, and to not make the inputLetter() method to big. I also needed a determineVictory() function to make sure that the game ends when the win condition is met.

For the HangmanGUI class, I determined there there should be a setupWindow() function to initialize the screen and fill it with whatever beginning text it needs. I also determined there should be a drawGallows() function to draw the gallows for the hanged man on screen. There should be a updateWord() function that updates the word the player is trying to solve. There should be a takeInput() and updateIncorrectLetters() functions as well that do as their name implies. I also determined there should be a separate function for each limb of the hanged man, so a drawHead(), drawBody(), drawLeftArm(), drawRightArm(), drawLeftLeg(), drawRightLeg().

I would have a computer drawn UML diagram, but I don't have the time to transfer my paper one to the computer.

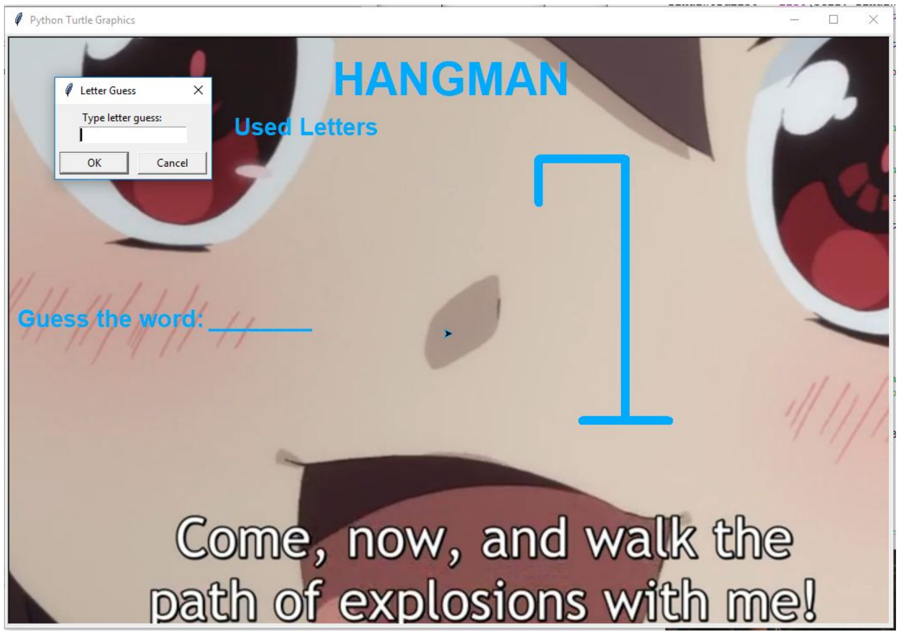
3. Implementation and Testing

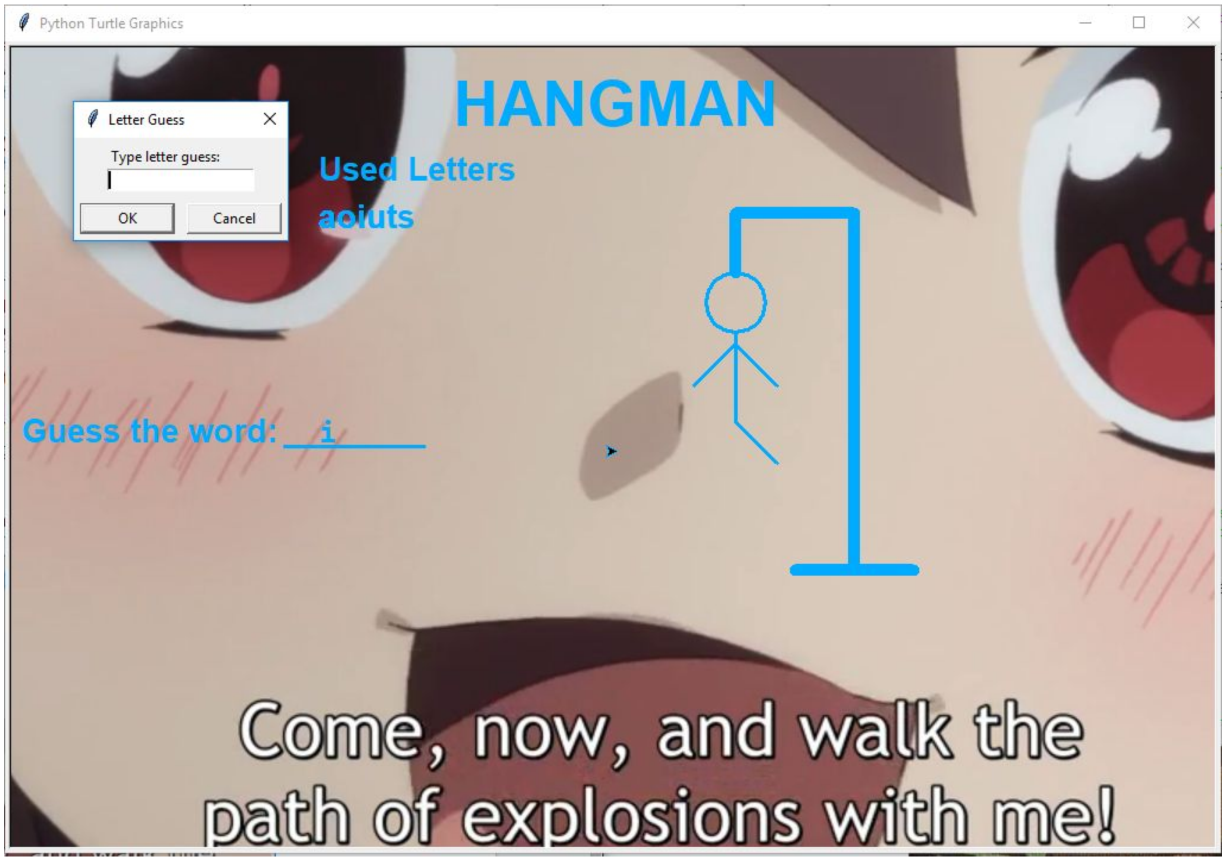
Due to how I initially designed the classes, it was pretty easy to test the methods as I created them. This saved a lot of headache later on when I finally tied the two classes together in the main class. I did add some methods as I was creating each class. In the Hangman class, I add a determineLose() class, as it does different things from the victory method and, staying true to one purpose per method, I refused to combine the two together. I also had to create a method called newGame() that cleared the data and reset-up the game for a new game.

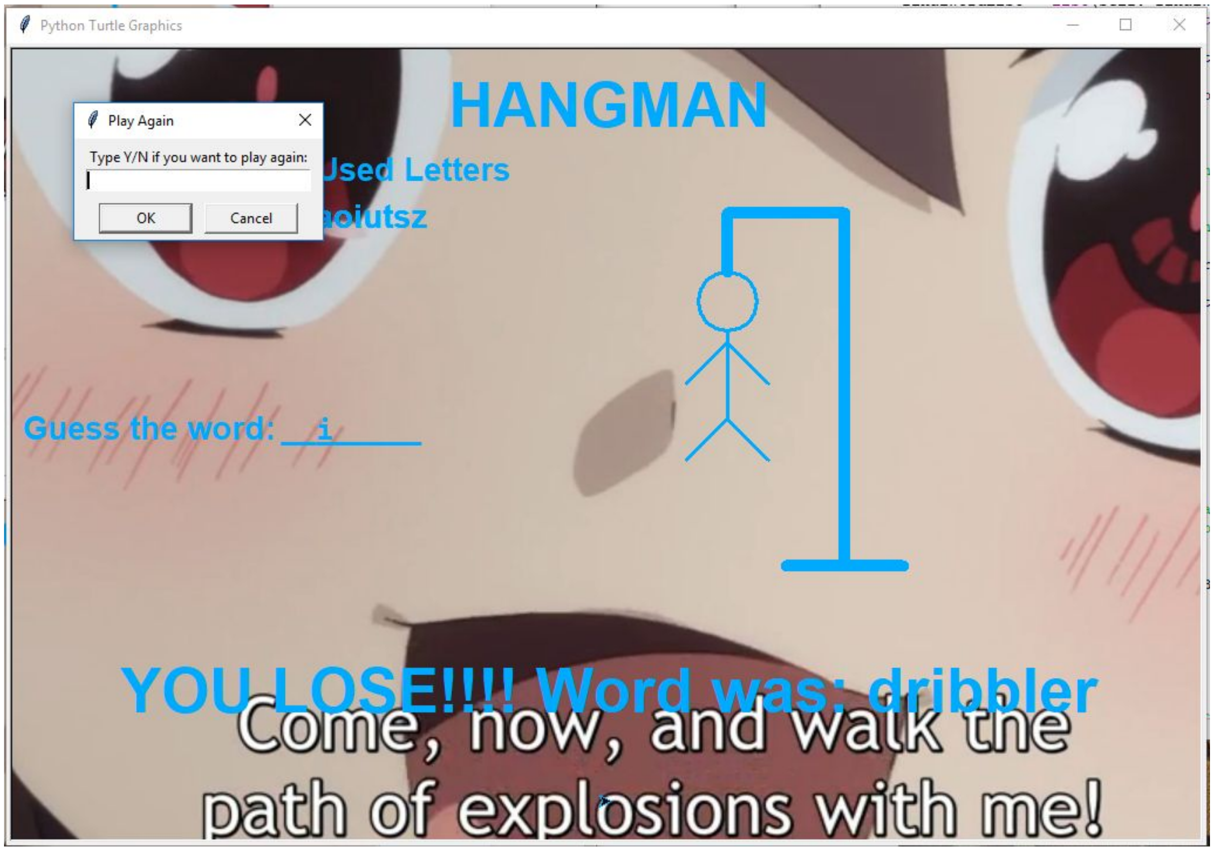
For the HangedManGUI, I had to create a victoryScreen() and loseScreen() to handle the input and messages shown when the game is over. These functions had a bug where I converted the input to uppercase for comparison, but I nevered saved it, so it was constantly giving a bad output till I discovered the bug, which took a while. I also created a newGame() function that cleared the GUI and reset it up for a new game.

With those two classes fully created, it was a simple task to combine the two together to get the game to work in the main class. The only issue that I ran into that took me a while to solve was a bug I had in the Hangman class, in the determineVictory() function where I compared the user created word and the correct answer to determine whether the game was won or not. The weird thing was that even though the code was correct as far as I could determine, it was giving a bad output, which confused me to no end. The code was “if self.currentWord == self.finalWord:” and that would always give me a False even when the two were the same. Being unable to solve this, I then changed it to look any “_” in the currentWord variable, and if there weren’t any, it would return True for victory being achieved.

Below is the output of the program in various states.




Python Turtle Graphics
Letter Guess
Type letter guess:
OK
Cancel
HANGMAN
Used Letters
aoiuts
Guess the word: __i______
Come, now, and walk the path of explosions with me!


Python Turtle Graphics
Play Again
Type Y/N if you want to play again:
OK
Cancel
HANGMAN
Used Letters
aoiutsz
Guess the word: __i______
YOU LOSE!!!! Word was: dribbler
Come, now, and walk the path of explosions with me!

4. Reflection

This was the most interesting project all year, as it tested the skill of thinking in an object oriented way to complete the assignment, and every step of the project was not scripted. Had I not been able to think in an object oriented fashion, this would have been a difficult lab, but I feel like I am adequate at thinking in that fashion. There is one flaw with this implementation, and that is that I didn't bother to error-handle the input on the try again input, so it will close the game if you enter anything but a "Y" or "y". Beyond that, this was an engaging lab that I enjoyed completing.