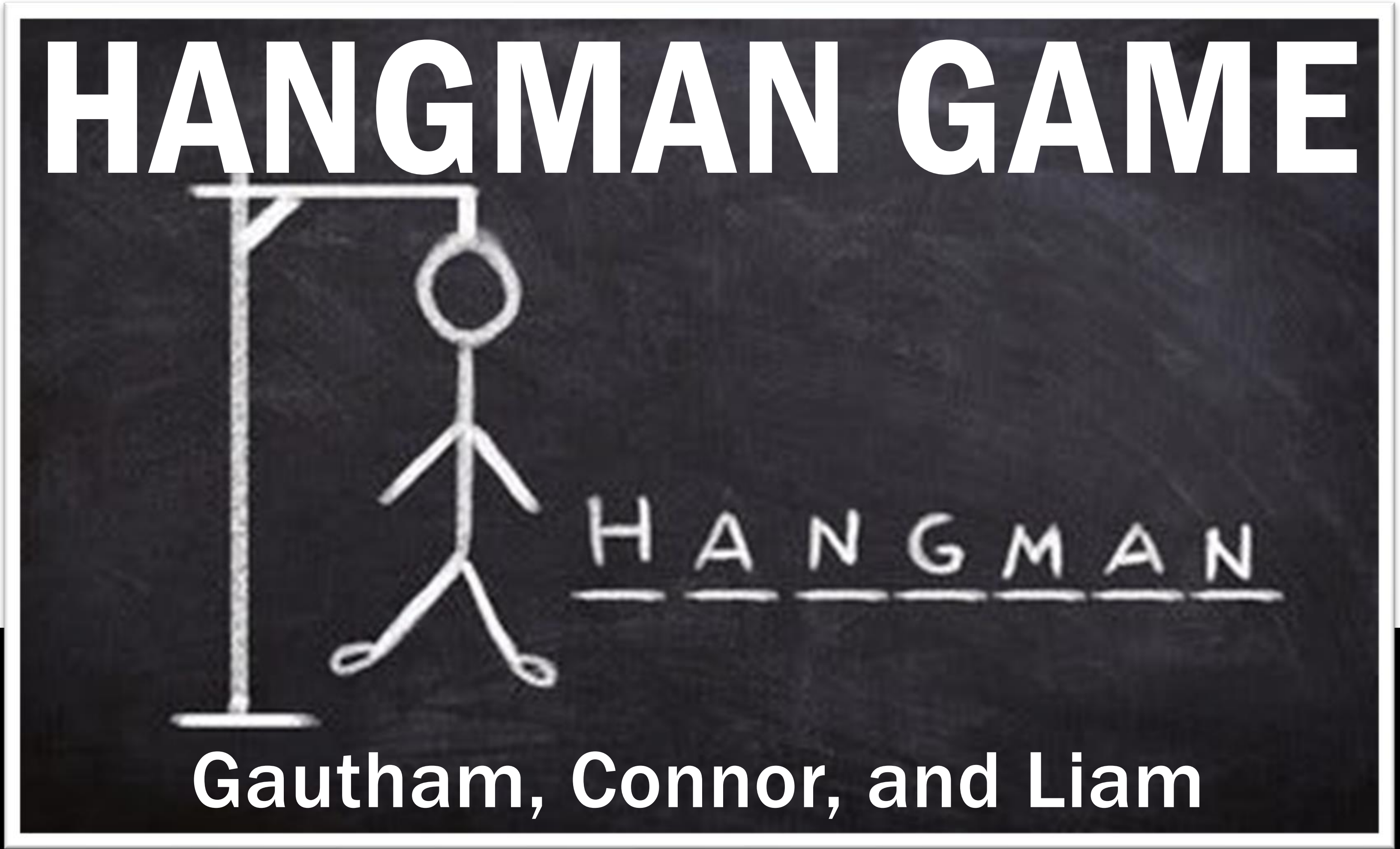
HANGMAN GAME



Gautham, Connor, and Liam

EECE 2140: Computing Fundamentals

INTRODUCTION

Objective and Goals:

- Design a word randomizer with several different categories to choose from
- Prompt the player to start guessing letters
- If the letter inputted are contained within the word, all instances of the letter will be revealed
- If the 6 attempts to the guess the word have been exhausted, the game will end

Project Scope:

- Focuses on a single-player Hangman game with predefined word banks

METHODOLOGY

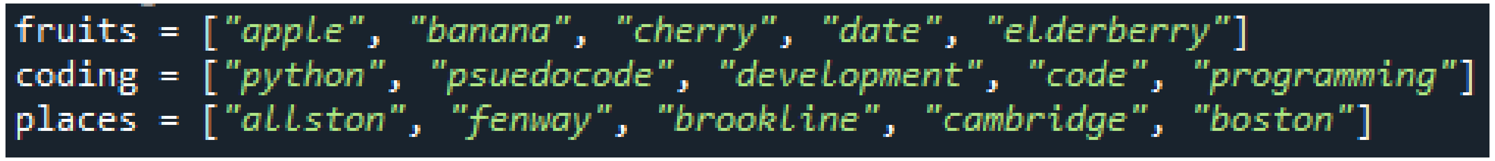
Description of Methods and Techniques Used:

- Utilizes Python's Tkinter library for UI alongside classes for game logic.
    - WordBank
    - HangmanGame
    - HangmanUI

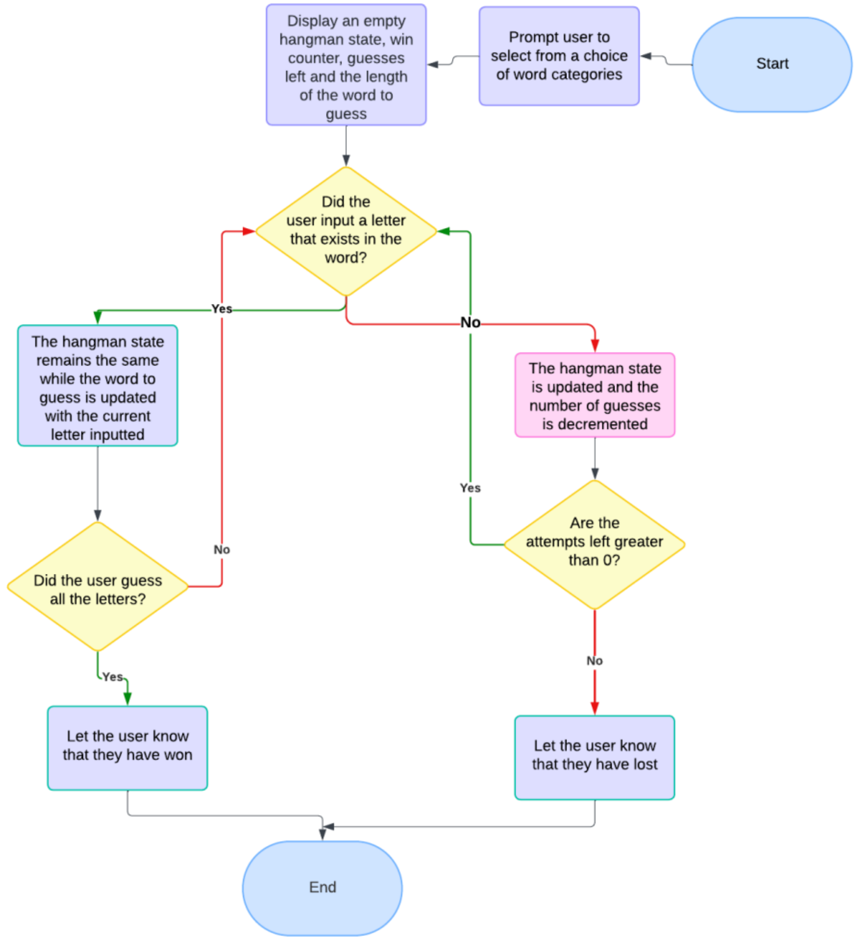
Pseudo Code for Techniques Used:

Data Structures Utilized:

- **List**: Storing words in category, randomly selected for guessing
- **Set**: Tracking guessed and correct letters: guessed_letters, correct_guesses
- **String**: Represents word to be guessed, guessed word stored
- **Dictionary**: Maps letters of alphabet to buttons in GUI: self.letter_buttons



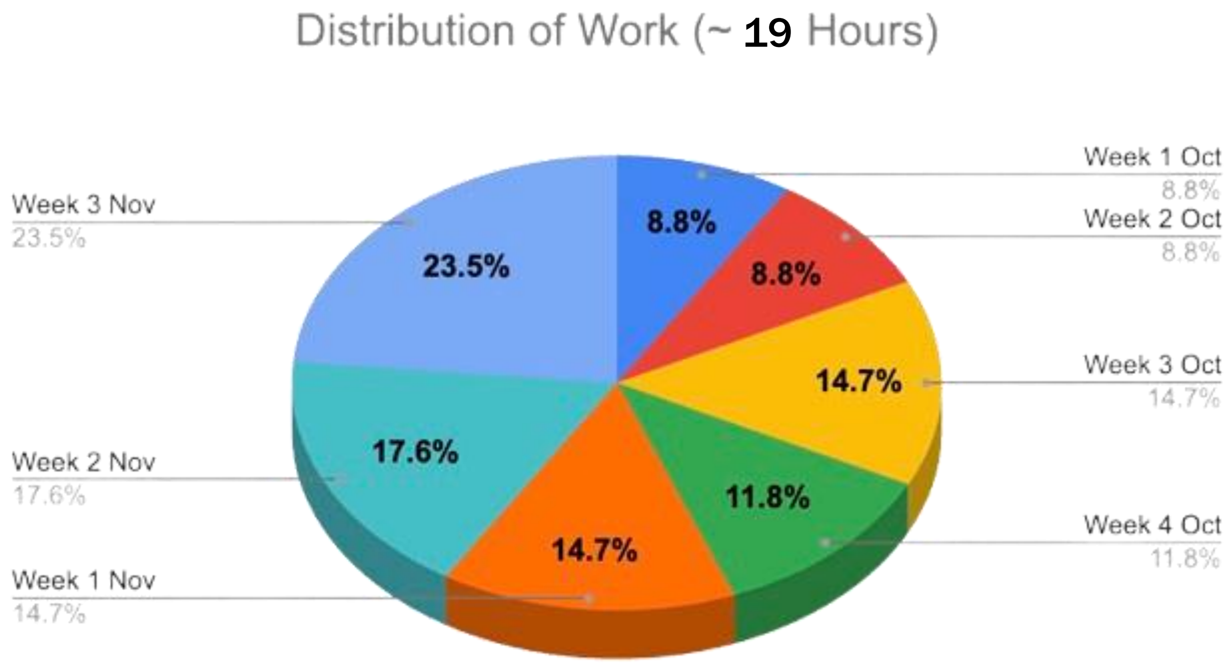
```
fruits = ["apple", "banana", "cherry", "date", "elderberry"]
coding = ["python", "psuedocode", "development", "code", "programming"]
places = ["allston", "fenway", "brookline", "cambridge", "boston"]
```



Start
Prompt user to select from a choice of word categories
Display an empty hangman state, win counter, guesses left and the length of the word to guess
Did the user input a letter that exists in the word?
Yes
No
The hangman state remains the same while the word to guess is updated with the current letter inputted
The hangman state is updated and the number of guesses is decremented
Did the user guess all the letters?
No
Yes
Are the attempts left greater than 0?
Yes
No
Let the user know that they have won
Let the user know that they have lost
End

WORK DISTRIBUTION

Project Stages:
- Planning
- Development
    - Word Randomizer
    - Game Logic
    - UI
- Integration
- Testing
- Final Changes



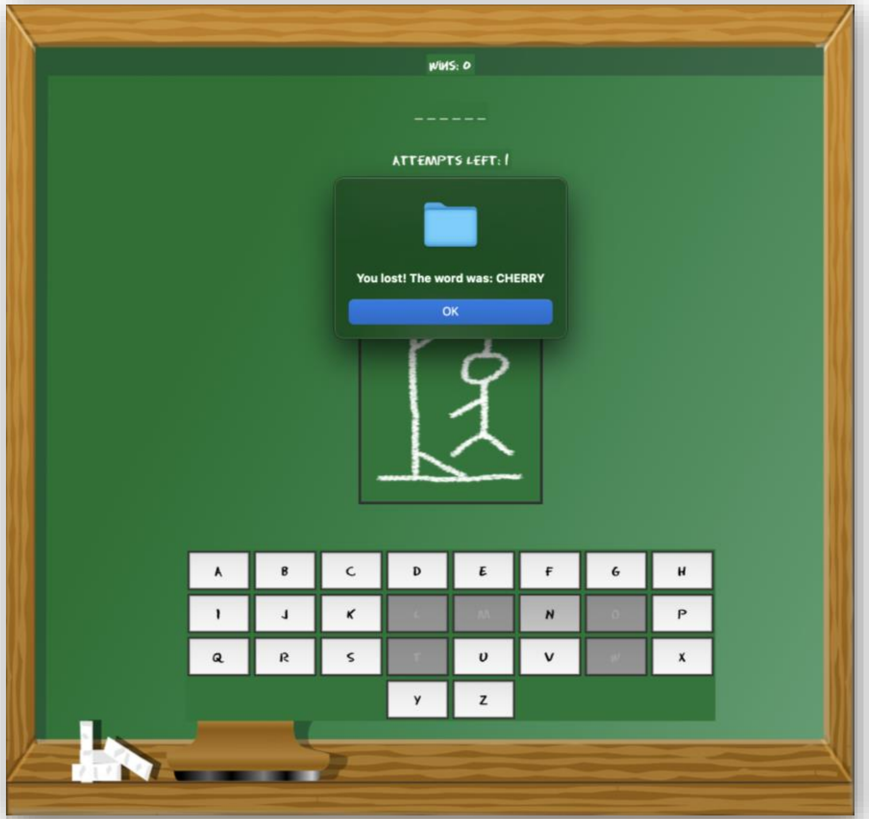
DISCUSSION

Implications of Findings:

- Using Pygame instead of Tkinter would have been more intuitive
- Tkinter has less flexibility in positioning game elements
- Button customizability in Tkinter is limited
- Create a list of libraries that can be used and assess each one

Project Limitations:

- A more expansive word bank could have been used with more categories
- Streamline the UI to correct minor imperfections
- Add additional buttons to customize the gameplay


WINS: 0

ATTEMPTS LEFT: 1

You lost! The word was: CHERRY

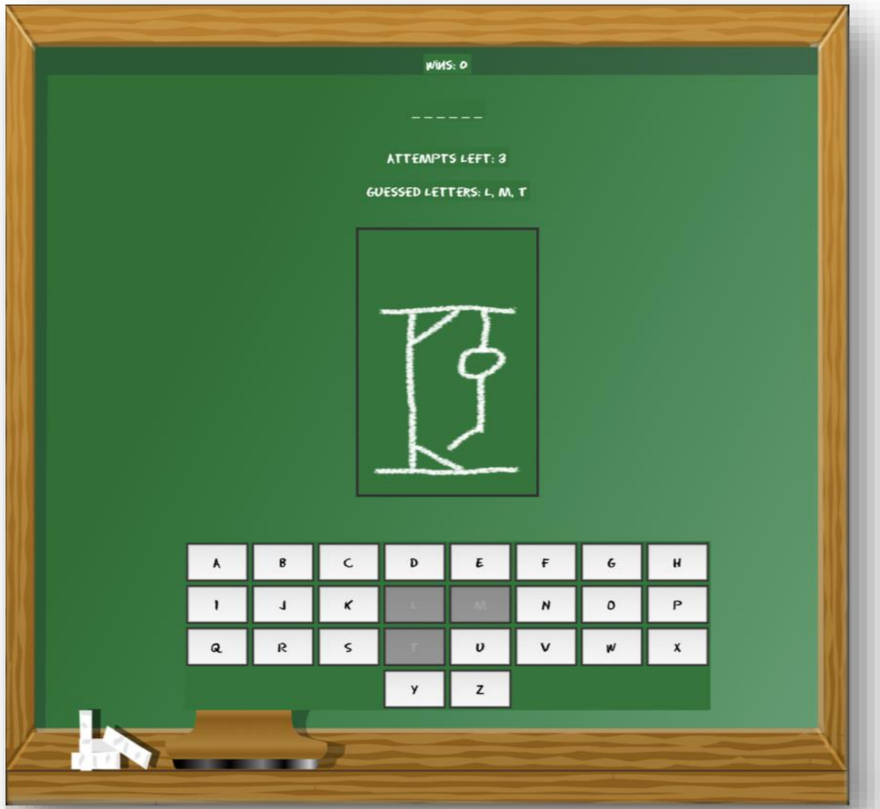
CONCLUSION

Conclusions from the Project:

- Implemented a fully functional category-based hangman game
- Made use of images to animate stages of the hangman
- Tinkered with custom fonts to follow the chalk board theme
- Followed a class-based design for seperately handling the UI and game logic

Recommendations for Future Work:

- Use a dictionary API to implement a dynamic word bank
- Additional gameplay modes (e.g., Timed challenges, Difficulty Slider)
- Add UI with animations for better engagement



RESOURCES

Summary of Relevant Existing Work:

[1]

Python, “Graphical User Interfaces with Tk — Python 3.7.4 documentation,” *Python.org*, 2019. https://docs.python.org/3/library/tk.html

[2]
“Pillow: Python Imaging Library (Fork),” *PyPI*, Oct. 15, 2024. https://pypi.org/project/Pillow/

[3]
W3Schools, “Python Classes,” *W3schools.com*, 2019. https://www.w3schools.com/python/python_classes.asp

[4]
R. Bansal, “Python GUI - tkinter - GeeksforGeeks,” *GeeksforGeeks*, Jun. 17, 2017. https://www.geeksforgeeks.org/python-gui-tkinter/