

**Project Title: A PySimpleGUI Game of Wild West Themed Hangman**

Author(s): Blake Buskirk Northeastern University

EECE 2140. Computing Fundamentals for Engineers Fall 2024

Date: 22 November 2024

**Abstract**

This project used PySimpleGUI as sg to include numerous python libraries based on PySide, Tkinter, and WxPython. The goal was to create a command-line version of hangman; however, a GUI was used to create a game window with more user interactions themed from the wild west. A class was created for game design with numerous instance variables to describe the GUI framework as well as the game functions and buttons while pyfonts and string were used to upload a font to the game and read the txt file to choose from the list of themed words randomly. For and while loops were used to develop game conditions for adding limbs to the hanged man, and setting conditions for game restart, quit, and new game. Moreover, sets and lists were both used to develop game conditions, framework, and coordinates to have python draw the game components that are illustrated. Certain letters and vowels were more probable since “s” and “o” are more likely to be found in wild west themed words and the limited theme of words can allow for easier guesses and wins. The game counted wins per games played “Maybe: and stores that data for when game relaunched”, but using the selected GUI there were many limitations to size the certain frames as the command must be in a certain part of the code, so moving the letters frame over to move the hangman game over proved to be difficult. In addition, the selected GUI drew the hangman objects as opposed to uploading personal drawings for each stage of the game. Therefore, OOP and OOD were used for the project, and beyond the scope of the project would allow for a web browser be used to run the game or create other game modes to select from.

# Contents

1. [Introduction](#_bookmark0) 2
   1. [Background](#_bookmark1) 2
   2. [Problem Statement](#_bookmark2) 2
   3. [Objectives](#_bookmark3) 2
   4. [Scope](#_bookmark4) 2
2. [Technical Approaches and Code UML](#_bookmark5) 3
   1. [Development Environment](#_bookmark6) 3
   2. [Data Collection and Preparation](#_bookmark7) 3
   3. [Implementation Details](#_bookmark8) 3
3. [Project Demonstration](#_bookmark9) 4
   1. [Screenshots and Code Snippets](#_bookmark10) 4
4. [Discussion and Future Work](#_bookmark11) 5
5. [Conclusion](#_bookmark12) 6

**Chapter 1**

# Introduction

## Background

The project had options to choose from beginner-level python projects, intermediate-level, and advanced-level. The game chosen was the hangman game from the intermediate-level python projects, and the basis of this was to create a hangman game that is a command-line version of it, so the user would play in the command-prompt instead the idea of creating a game that opens in a new window was chosen to produce a more interactive hangman game.

## Problem Statement

This project is done to create a game of hangman using a class system, main function, and PySimpleGUI to integrate the game into a new window for interactive features.

## Objectives

To successfully develop a hangman game that allows players to quit the game, restart the game, and start a new game while keeping track of wins. Moreover, this project’s purpose was to use knowledge from lecture and personal research to demonstrate an understanding for object-oriented programming, object-oriented design, and the class structure with methods involved such as the instance method. The objective was to create a wild west themed hangman game with the hangman drawing being of a cowboy that gives extra guesses due to components like a hat and bandana being added to the drawing. Also, a goal was to put tumbleweed in the background to give a western theme to the post for the hangman drawing.

## Scope

The scope of the project included creating a command-line hangman game that was developed to instead be used with a GUI to create a game window with buttons and user interaction for the inputs. Moreover, the hangman drawing itself was to be developed with a coordinate system while the game needed a class to implement the game methods and drawings. Beyond the scope of the project would include using the PySimpleGUI or even converting to a pygame to produce an app or webpage for players to play online. Also, to create a mode to verse another online player, or create a chaos mode where the limbs randomly appear or disappear to create a stress mode of the classic hangman game with a time limit. There was a 2 month time limit for the project, and for the first two weeks ideas were brainstormed and the hangman game was chosen, for weeks three to six the game started off as a command-line hangman game, but as the game was being developed it was changed to be supported with a GUI to create a more user interactive experience. Moreover, the last two weeks were used to finalize errors and send everything to the github repository as well as creating the presentation. There were many deliverables for this project as iteration one the theme was chosen, for iteration two the timeline and initial development of the txt file to create the random words was created and for iteration three project tasks were created such as sending more of the code back to the github web repository, starting the class, and creating the random words with the wild west theme. The GUI was created and the presentation included a demonstration of the game to emphasize the theme of the game, and the code to make the game work.

**Chapter 2**

# Technical Approaches and Code UML

## Development Environment

Python version 3.13 was used to update pip and use it to download the PySimpleGui, String ascii, and pyfonts. At first basic functions were used to develop the game until the game development switched to using a GUI, so a class was created with many instance methods to implement the frame of the game and the functions associated with the game as well as object usage to restart the game, quit the game, start a new game, and close the game after popup windows. Moreover, google was used to understand that html web colors have to be used when choosing line colors for the GUI coordinates to draw, and math and random were used to choose random angles and calculate the x and y end lines for the tumbleweed drawings.

## Data Collection and Preparation

A txt file was used to create the wild west themed phrases, and with this file the words were taken and made uppercase to create the randomly selected word that was to be guessed. A string was used to group alphabet letters in groups of four and then separated as one letter to create the buttons and the event associated with the button press to indicate when a letter is chosen. The hangman drawings were created with coordinates as well as a list labeled body to indicate the order of which a limb would appear when a guess was wrong, and if conditions were used to create the head at the first wrong guess and step each body part to be added as another wrong guess was made.

## Implementation Details

For the coding process, it was known that the random library would be the first one needed as a function later used as an instance method to import choice from random for the code to randomly pick a word from the list. With this a maximum of wrong guesses was created and set to six to include each limb of the stick figure, but when the wild west theme was developed more it was changed to eleven to account for the hat, bandana, vest, and boots added to the drawing. As the choice to use a GUI was implemented PySimpleGUI was imported as sg to be able to use the functions to create the frames and edit the window that would be used for the game. Moreover, with the switch a class called hangman was created with the constructor function having a layout list with frames for the gameboard, the letters, the word being guessed, and the buttons using the self.\_ to create private variables that would later be accessed. Outside of the layout list more private variables were created for the window and title, the gameboard, the new game method, quit game method equal to false so the game starts up, and a counter for games won and played. The gameboard frame was initialized as an instance method to create a title, a graph with coordinates using sg to create a key, canvas size and set where the frame that has the hangman drawing would be. An instance variable was created for the letters frame and groups for the letters were created with the string library as ascii was used to make the letters uppercase and range letters in groups of four to use len() and create buttons the user will interact with for each letter. An instance method was created for the words guessed frame to return the word with underscores and have letters appear as the letters were guessed. The buttons framed used a sizer to space the letters and keys were created for the new game, restart game, and quit game functions. The next part of coding the game was to code the post, and knowing the gameboard frame was (0,0) to (300,600) the post was drawn to be around the center of that and a for statement was used to draw the lines with a silver color. Moreover, tumbleweed was drawn and this occurred with the math library as there was a center and spokes we given for lines to be drawn around the center. In addition, the math library was used to calculate the x and y ends of the spoke, and random library was used to randomly choose the angle and length of each spoke. After this the hangman coordinates were given with the head being placed and then the hat was created using lines and triangles while the bandana was created with triangles, and this occurred to produce each limb using a list of coordinates for each limb. A list called body was created to initialize the order at which each limb would be chosen when a guess was wrong, so an if statement was used for when the wrong guess was one then the head would form, and the rest was when the guesses was greater than one a for statement was used to draw each part in the body list. This process was used to develop the window from the GUI and indicate the drawings and gameboard setup.

The next part of the coding process was to create the instance variables for the game process. Previously the random word chosen variable was created before the decision to use a class system, so that format was used for an instance method to open the txt file and create the word choices list to be from the file and return the choice which was imported from random library in uppercase. The next instance method was to update the guessed word frame that when a button is clicked the letter will update and replace the underscores. This was done via for statements and the append method to update the letters. Next the new game instance method was created that when the target word equaled the word selected the game would select a new word and reset any other game settings. Moreover, the game restart method was created and the method set was used to make the letters chosen list restart at empty and reduces the counter of wrong guesses to zero while the new target word would be created. In addition, the gameboard would erase and the drawings would reset to show the post, and two tumbleweed drawings, and a for statement was used to reset the button frames and have none disabled while the key for the word displayed was used for the window to update the word. The next instance method was the game play and this was attached to letter as well as self to increment the wrong guesses if a letter was chosen wrong, and the display would be updated. Moreover, this function calls back to the hangman drawing to update the drawing. Two functions were created called event and event read in order to wait for an event which would be button click and the event id would be chosen and returned for the event read function to keep the game playing, restart the game, or start a new game. The game over function was created to indicate that the game would end when guesses equaled max wrong guesses or when the words aligned while a game winning check was created to increment the games won and produce a statement asking if the player would play again using a popup window with the GUI library. This process occurred for when the game was lost and a popup window would produce, but if selected no then the game would quit using the quit game variable and if not a new game would begin. The last variable in the class was created to close the game and it would close the GUI window if selected. After this the main function outside of the class using the method \_\_name\_\_ == \_\_”main\_\_” was used to run the code in order to make it accessible when running directly or as a module from a download. This function called back to the hangman class to run it and loop to continue until the game quit, and to loop through until the game was over. Moreover, this function would capture the event and create a condition to make quit game true if selected. A break was then used for the loop to process the event and hold off on actions until the event was processed and if the game was not quit the winner would be checked, and lastly the game close function was called back to close the window.

**Chapter 3**

# Project Demonstration

## Screenshots and Code Snippets

(Include visual evidence of the project’s outputs here. Add screenshots and code snippets as needed.) For the win statement popup window, the code is as follows as well as the output provided.

A screenshot of a computer program

Description automatically generated

*Figure 1: Conditions for Game Win/Loss for Popup Window*

A screenshot of a computer game

Description automatically generated

*Figure 2: Game Output when Win Conditions Satisfied*

A screenshot of a computer game

Description automatically generated

*Figure 3: Game Output with Lose Conditions Satisfied*

When the restart or new game process is chosen the game resets. With a before and after picture and code snippets.

A screenshot of a computer game

Description automatically generated

*Figure 1: Game Before Reset/New Game Chosen*

*A screenshot of a computer game

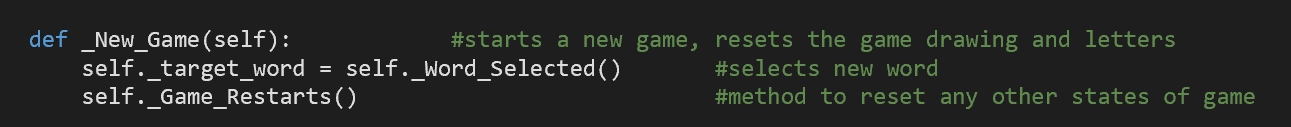
Description automatically generated*

*Figure 2: Game After Reset/New Game Chosen*

A computer screen shot of a computer program

Description automatically generated

*Figure 3: Reset Game Code*



*Figure 4: New Game Code*

A screen shot of a computer code

Description automatically generated

*Figure 5: Event Read for Functions*

**Chapter 4**

# Discussion and Future Work

There was a limited list of specific words that could lead to deviations with the level of difficulty for guessing words since there is a theme of words. In addition, game.close() was used to close the game when the player hit quit, but at first there was some error with a claim that there is no attribute to close in the class, but that was fixed since there was a capitalization error between the close() instance variable and the use of game.close() in the main function. Some limitations would be the need to download PySimpleGUI as well as downloading the venv file and activating it to use the GUI. Consequently, the GUI used requires an activation code so an account would be needed if someone were to download the code to play. In addition, the game data would be reset after the game was closed, so if the game was relaunched then the number of games played, and games won would also reset for the player. Moreover, it was learned that python needs a file, list, or defined set of words to be able to choose a word at random for the game. The GUI used was imported as sg to the game code, but the function sg.Sizer() to size the frame of the drawing, letters, and game commands to exit, and restart had a lot of limitations, and the letter box would not move over anymore due to the limitations. A code editing app was used for some of it instead of spyder, so some syntax errors were only seen after the code was ran in gitbash. With the GUI and setup chosen, the colors for the drawings built from coordinates there were limitations as there could only be html web colors chosen, so for the tumbleweed the color brown or tan could not be chosen, so maroon was inputted for the tumbleweed color.

**Chapter 5**

# Conclusion

With a theme of wild west words being used, a pattern developed which indicated that guessing a “c”, “l”, or “o” would produce a higher chance of a right letter being chosen. The only game data that was collected was a win counter to output the number of games won compared to the number of games played. The projects objectives were obtained as the game was completed and it functioned well with the cowboy theme being followed, and the drawing completed by the hangman that looked like a cowboy. The project produced a greater understanding of classes and object-oriented programming as the class ran the game with the use of the GUI and executed all functions as intended without any errors during the game running and after. Moreover, this project also gave knowledge with event reads and keys as those were researched and used in order to allow for buttons to be clicked the change the game to a new one or quit the game.

# Bibliography

[1] Clough, David E, and Steven C Chapra. *Introduction to Engineering and Scientific Computing with Python*. CRC Press, 2022.

[2] Deitel, Paul J., and Harvey M. Deitel. *Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and the Cloud*. Pearson Education Limited, 2020.

[3] GeeksforGeeks. “Getting Started with Python Programming.” *GeeksforGeeks*, 5 Sept. 2024, www.geeksforgeeks.org/getting-started-with-python-programming/?ref=shm. Accessed 10 Oct. 2024.

[4] *Google*, www.google.com/. Accessed 26 Nov. 2024.

[5] Lutz, Mark. *Learning Python*. 5th ed., O’Reilly Media, Inc.

[6] McKinney, Wes. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. 3rd ed., O’Reilly Media, Inc.

[7] “Wanted Poster Creator.” *Wanted Poster Creator - Make a Wanted Poster | Create Funny Wanted Posters with Your Own Picture and Text*, www.tuxpi.com/photo-effects/wanted-poster. Accessed 18 Nov. 2024.

[8] “Web Colors.” *Wikipedia*, Wikimedia Foundation, 19 Oct. 2024, en.wikipedia.org/wiki/Web\_colors. Accessed 10 Nov. 2024.