WORD HUNTER

PROJECT REPORT

Ömer Faruk Satık

June 4, 2023

Word Hunter: A combination game of the official games Wordle and Quordle

by
Ömer Faruk Satık
ID: 150210330

to
Ahmet Cüneyd Tantuğ
Istanbul Technical University
Faculty of Computer and Informatics
Artificial Intelligence and Data Engineering Dept.

Istanbul, Turkey
June 4, 2023

SUMMARY

This report presents the development and analysis of the "Word Hunter" game, which combines the mechanics of the popular games Wordle and Quordle. The aim of the project was to create a user-friendly game experience that challenges players to guess hidden words within a limited number of tries. The game was developed as a desktop application using the Python programming language, specifically utilizing the PyQt5 library for the graphical user interface. The report discusses the planning and development process of the game, including the decision to develop it as a desktop application instead of a web-based game. The game's interface was designed using PyQt5 and the Qt Designer program, allowing for a visually appealing and intuitive gameplay experience. A dataset of English words was collected from the OPTED (The Online Plain Text English Dictionary) and used as the word pool for the game. The game logic of Word Hunter was explained, highlighting the various game modes and the algorithm for handling user input and updating the game state. The performance of the game was evaluated, confirming its usability and functionality. Additionally, the potential impact of Word Hunter on vocabulary and word-finding skills was assessed, noting that the game encourages engagement with the English language and has the potential to improve players' vocabulary and word-finding abilities. Overall, the Word Hunter game successfully combines elements from Wordle and Quordle to offer an enjoyable and challenging gameplay experience. It serves as a valuable resource for those interested in word puzzle game design and development, and showcases the possibilities of the Python programming language for implementing game logic. Suggestions for future development include adding features such as difficulty levels, player registration, leaderboards, and player statistics.

Contents

Word Hunter: A Combination Games of the Official Games Wordle and Quordle

***Abstract*—Word games are not only enjoyable by players of all ages, but also very important in terms of improving their vocabulary. Wordle and Quordle games, on the other hand, give players brain training as they contain strategy elements. In this paper, the development stages, methods and general performance of the game Word Hunter, which was developed in Python language by combining the logic of these two games under one roof, will be discussed.**

Keywords—Python, game development, PyQt5, design

I. INTRODUCTION (*HEADING 1*)

The aim of this project is to examine the Wordle and Quordle games in detail, understand the mechanics of the games and then develop the “Word Hunter” game, offering a user-friendly game experience, that combines the logic of the games Wordle and Quordle.

This project report will cover the design and operation of the Word Hunter game, explain in detail the game's key components such as the user interface, word selection algorithm, main algorithm, and evaluate the game's performance.

In the rest of the report, we will describe the development process of the Word Hunter game, discuss the algorithms and data structures used in the Python language, and test the usability of the game. We will also assess the impact of Word Hunter on vocabulary and its potential to improve players' word-finding skills.

This project aims to be an important resource for those interested in the design and development of word puzzle games. It will also be a useful study for those who want to develop games in the Python programming language and apply the logic of word finding using algorithms.

II. METHODS

A. Introduction

Word Hunter game was developed using the Python programming language. The game aims to offer players the challenge of guessing one or four hidden words within a limited number of tries. The words are chosen to be four, five or six letters according to the player's request.

B. Planning the Procedure

This game was originally planned to be developed on the web as it was declared in project proposal. For this, one of the Django and Flask libraries would be decided on. Research has been done on both. As a result of these researches, I gave up on the web-based style for 2 reasons:

- Django and Flask libraries required a lot of HTML, CSS and JavaScript knowledge and my knowledge was insufficient.
- Since the use of these libraries gave too much space to the languages mentioned in the previous article in the project, the use of the Python language was reduced and the project was moving away from being a Python project.

For above reasons, I decided to develop the game as a desktop application and followed the following plan in order:

1. Python library research for game interface design
2. Deciding on PyQt5 because of the extensive possibilities and ease of use it provides
3. Watching PyQt5 tutorial videos
4. Draft interface designs and simultaneous draft algorithm writing with Qt Designer program
5. Dataset search for English words
6. Saving words in a set inside a class structure in "words.py"
7. Logo design and addition to the design with Adobe Photoshop
8. Finding button icons from the internet and adding them to the design
9. Adding the icons and logo in the design to the source file and converting the file to "py" file
10. Finalizing the interface design
11. Finalization of the algorithm
12. Creation of the game's exe file
13. Preparation of requirements.txt and readme.md files

C. Development Environment

The game was developed using the Python 3.10 programming language. PyCharm IDE (Integrated Development Environment) has been preferred because of the useful interface it offers as a development environment. PyQt5 library was used to visualize the written algorithm.

D. Data Collection

A detailed dataset research was conducted on the Kaggle website for the word pool needed in the game, and OPTED (The Online Plain Text English Dictionary) was decided upon. The words in the dictionary data that comes in a csv file are stored in a set in the "Words" class in the "words.py" file.

E. Graphical User Interface (GUI)

The graphical user interface (GUI) for the game was developed using the PyQt5 library. The design of the pages

was done through the Qt Designer program. The design consists of 7 pages in total. The first page is the main menu page and contains game mode selection buttons. Each of the other 6 pages represents a mode of the game. On these pages, there are word blocks to make predictions, on-screen keyboard, main menu key, replay button and close game button. When the buttons are clicked, the necessary functions are executed thanks to the "clicked.connect()" method of the PyQt5 library. This design, which was produced as a "ui" file, was converted into a "gui.py" file with a command line and became importable in the main algorithm. The game logo, which appears as a window icon on the pages, was designed with Adobe Photoshop.

F. Game Logic

This project is a game where you try to guess 4, 5 and 6-letter words only one word at a time or 4 different words simultaneously. These make up six selectable game modes in total. The words in the game are taken from the English language. In this game, the player is given a guess right in accordance with the selected game mode and is expected to find the correct word or words before these rights are filled. As the player makes a guess, it is examined whether the letters in the word entered are in the winning word. If the entered letter is in the correct position, that button turns green. If the entered letter is in the winning word but its position is wrong, the button turns yellow. If the entered letter is not in the winning word, the button turns gray. In order to improve the user's gaming experience and provide convenience, the on-screen keyboard keys are colored with the same logic according to the situation of the letter in the word. This coloring is simple for the single word mode while a mess is possible for the four-word mode. Because while the letter on the keyboard key is present in one word, it may not be present in the other. In order to avoid this complexity, the prediction blocks are placed in the form of “+” sign in the four-word mode, and the relevant edge of the key on the keyboard is painted in the relevant color according to the position of the block. In this way, the player is provided with a hint. If the player guesses the word(s) correctly before the guessing rights expire, he wins the game. If the guesses run out before the player find the word, the player loses the game. When a word that is not in the English dictionary is entered, the warning message is returned and the next row is not passed.

G. Algorithm

The algorithm is written in a class named “MyMainWindow” in the “app.py” file. First of all, GUI design which is imported from “gui.py” is loaded with the "setupUi()" method. Words consisting of four, five and six letters in the set created in the “Words” class imported from the “words.py” file were filtered and saved in different lists.

The "texter()" function defined in the class is called inside the initialize function. This function detects clicks on the on-screen keyboard and sends data to the "on_button_clicked" function, where the game algorithm is processed. In the line after the “texter()” method, the “open_page1()” method is executed and the main menu of the game is opened and the game starts. Each time the “Back to main menu” button is clicked, this function is called again and the main menu is returned. In total, 7 “open_page” functions have been defined. Each of them opens a page of the design with the "stackedWidget.setCurrentIndex(<index>)" method and arranges the parameters of the game to fit that page.

While in the main menu, the player is expected to choose one of the 6 modes. When the game mode is selected, random hidden words are selected from the pre-prepared lists according to the number of words to be guessed in that mode and the number of letters the words contain, thanks to the "random" module. And other necessary parameters (game mode name, number of letters, list of words to choose from, number of lines) are formatted according to the mode. Also, the "determine_row_dict()" function is called. This function creates a dictionary that stores prediction lines according to the selected mode. In this dictionary, the keys are the row numbers and the values are the lists in which the boxes that make up the rows are recorded in the prediction blocks. This dictionary is created to access each box while writing the basic algorithm of the game. In this way, letters can be written and deleted on the boxes and the colors of the boxes can be changed.

When the “Play again” button is clicked, the “refresh()” function is called and all parameters on the page are reset, the word block and all colorings are cleared with the “clean_buttons()” function. Printing, deleting and coloring in the game are all done with looping and conditional blocks in the "on_button_clicked()" function. Clicking the "Quit Game" button closes the application.

III. DISCUSSION

The purpose of this project was to examine the Wordle and Quordle games in detail and develop a new game called "Word Hunter" that combines the logic of these games while offering a user-friendly gameplay experience. In this section, we will discuss the key components of the Word Hunter game, evaluate its performance, and assess its potential impact on vocabulary and word-finding skills.

Firstly, the development process of the Word Hunter game was described, highlighting the decision to develop it as a desktop application instead of a web-based game. This decision was made due to the complexity and requirements of web-based development, as well as the desire to keep the project focused on the Python programming language.

The development environment for the game consisted of Python 3.10 and PyCharm IDE, providing a convenient interface for development. The PyQt5 library was used to visualize the game's interface, and the Qt Designer program was utilized to design the graphical user interface (GUI). The GUI design involved creating multiple pages, including a main menu page and different game mode pages. The design process also included the creation of a game logo using Adobe Photoshop.

The game logic of Word Hunter was discussed, explaining that it challenges players to guess hidden words of different lengths within a limited number of tries. The game modes allow players to guess either one word at a time or four different words simultaneously. The player's guesses are evaluated based on the correctness of the letters and their positions in the hidden word(s), and the on-screen keyboard provides visual cues to assist the player. The algorithm for handling user input and updating the game state was implemented in the "app.py" file, utilizing a class named "MyMainWindow".

In terms of data collection, a detailed dataset research was conducted on the Kaggle website, and the OPTED (The Online Plain Text English Dictionary) dataset was selected to provide a word pool for the game. The words from the dictionary data were stored in a set within the "Words" class in the "words.py" file.

The performance of the Word Hunter game was evaluated based on its usability and functionality. The PyQt5 library provided extensive possibilities and ease of use for designing the game interface, allowing for a visually appealing and intuitive gameplay experience. The game's algorithm successfully handled user input, checked the correctness of guesses, and provided appropriate feedback to the player.

Each mode of the developed game has been tested both during development and after completion. The tests carried out during the development phase played a critical role in detecting errors and deficiencies. Tests made after the game was finalized confirmed that the game was at the target point. Two example tests are indicated in Fig. 1 and Fig. 2.

Fig. 1

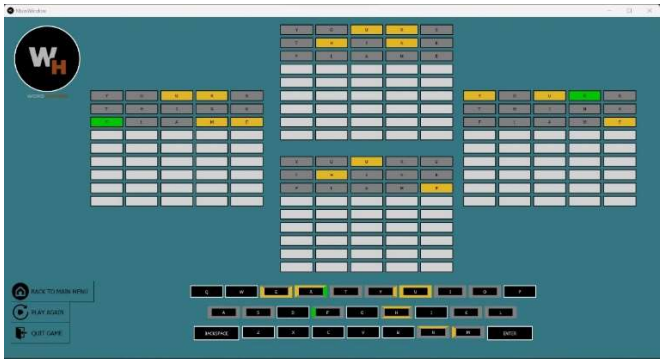

Fig. 2

Furthermore, the potential impact of Word Hunter on vocabulary and word-finding skills was assessed. By challenging players to guess hidden words within a limited number of tries, the game encourages engagement with the English language and can potentially improve players' vocabulary and word-finding abilities. It is possible to add a new word to your vocabulary with each round. Therefore, this game is a game that makes you win even if you lose.

Although the game can be played smoothly, it is open to development. Features such as difficulty level, player registration and login, leaderboard, player statistics can be added.

IV. CONCLUSIONS

1) The Word Hunter game has successfully combined elements from Wordle and Quordle to offer an enjoyable and challenging game experience.

2) It is a very useful game for those who want to improve their English vocabulary in addition to the pleasant experience it provides, because all the words in the English dictionary are included in the game.

3) This game, which can be quite challenging from time to time, pushes players to produce different strategies. Because when you play without developing a strategy, you can lose the game for hours without finding the right words. The game is also a brain training application as it pushes to produce strategy.

4) The project provides an important resource for those interested in word puzzle game design and development. It can also be considered as a valuable study for those who want to apply game logic using the Python programming language.

5) The possibilities offered by the PyQt5 library and the Qt Designer application have proven how useful tools they are for developing such games.