Linguistic gaming with Python
WS 23/24

Artur Schäfer
Donatien Leray

WORTEX

Introduction

Leaning new words can be pretty boring, so we have created a game to make it more exciting. Wortex is a linguistic word game. It is designed to challenge players' linguistic skills by letting them write all the words they can form out of a given set of letters. In this report, we will discuss the objectives, mechanics, and features of the game. Linguistic games offer a fun way to improve vocabulary and spelling skills. Wortex is implemented in Python and uses the Pygame library for its graphical user interface.

Objectives

The primary objective of Wortex is to form as many words as possible from a given set of seven letters. Players must create words of at least three letters, up to the maximum length of seven.

The more words a player forms, the higher their score. To achieve a top score, players should aim to construct not only words, that are as long as possible but also rare. The game ends if the timer runs out or if the player finds all possible words, that can be formed from the set of given letters.

Mechanics

Menu

Upon launching Wortex, players are presented with a menu. Here, they can choose a language (English or German) they prefer to play with. There is also a button to view a scoreboard, which is showcasing the highest scores ever achieved. The player also can choose a difficulty level where the time is increased or decreased depending on the level.

1. **Easy:** 120 seconds
2. **Medium:** 90 seconds
3. **Hard:** 30 seconds
4. **Extreme:** 15 seconds

Figure 1: The main menu of Wortex

Gameplay

After selecting a language and the play button is pressed, the game begins. It provides the player with a set of seven letters. On the screen you can see a timer, the score, and a list of words that appears if the player types a correct word. In the middle there is a circle that represents the time and in that circle there are the letters that can be typed to form words. Also there is information about how many words can be found in the given set. The player can form the words by typing the letters on the keyboard. With the escape key, the player can reset his input or by deleting the last letter with backspace. The same works with enter/return too. After guessing a correct word, the input gets cleared and the found word is added to the list of found words. If a seven letter word is found, the player gets bonus time (10 seconds) to find more words. Also if the player guessed a word that is rare, the player gets more points than for a common word. There are no penalties for guessing wrong. The player is just wasting time that could have been used to find more words.

Features

After the timer runs out or the player achieves to find all possible words, the game ends and the player is presented with a screen that shows the score and all the words that could be found and also were found by the player marked

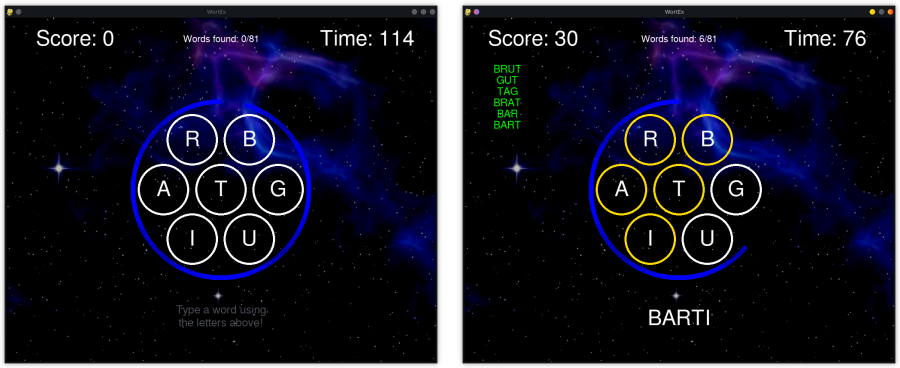

Figure 2: Gameplay of Wortex

as green. The words can now be clicked on and it opens the browser to the definition of the word. In German we open the online Duden dictionary and for English the online Oxford dictionary. The player can then repeat the process by pressing the play again button or go back to the main menu. From the main menu the player can view a leaderboard, which shows the highest scores ever achieved. If the game is started again, the player gets a new set of seven letters to guess words from. We have also implemented a scoreboard for all the different difficulties. If you want you can also add your own language. Just follow the steps in the README file.

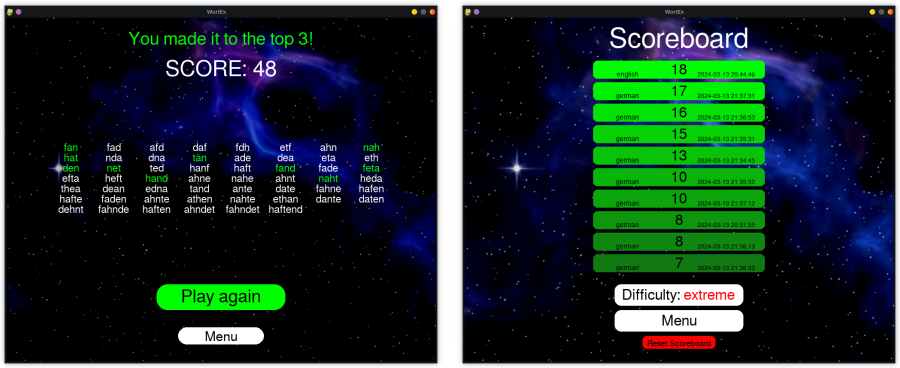

Figure 3: The end screen of Wortex

Scoring

The score is calculated by the length the frequency of the word of that given language. Frequency describes how often a word appears in a large amount of text. The more frequent a word is, the less points the player gets for guessing

that word. We used a frequency list from a github repository that contains about 60000 words and their frequency in the German language. We filtered all the words that are out of range of 3 to 7 letters. Afterwards we calculated the relative frequency of each word and used that to calculate the points for each word. Storing this information in a database file allows us to just look up the words points and saving us to calculate the points on every startup. We also saved a lot if space by storing the data in a binary file instead of just plain text. The frequency list we started at was about 116.5 MiB big and the db file is around 6 MiB big. This is our mathematical model to calculate the points for a word:

$$P = 1 + \frac{(L-2) \cdot (1 + 10 \cdot (1 - F))}{15} \tag{1}$$

P are the points of a word, L is the length of the word and F is the relative frequency of the word. The relative frequency is calculated by dividing the frequency of the word by the maximum frequency of all words in the list. The maximum frequency is the frequency of the most frequent word in the list. The points are then multiplied by a factor that is calculated by the average points of all words. This factor is used to make the points more accurate.

$$f = \frac{\sum_{i=1}^{n} P_i}{n} \tag{2}$$

We finally round the points and return them as an integer.

$$P_i = \lfloor P_i \cdot f \rfloor \tag{3}$$

Easter Eggs

If the player manages to form the words "Wortex", "Dodo" or "Artur" they receive a bonus of 42 points and gain extra time. Also the look of the game changes a bit.