Pedro Carneiro Passos, Alvin Huynh, Ramon Asuncion, Olivia Peters
CSCI205 Final Project - WordFX
May 3rd, 2022

Design Manual

[Introduction]

WordFX begins by listing three different buttons corresponding to three game modes. Once the game mode is selected, the program will read in the appropriate file (3 letters, 4 letters, or 5 letter word files). Now along with the separate word files, each file has a corresponding file for example one is called ‘allowed3words.txt’. Once you select the mode you want to play, the words from the two files are loaded into two arrays called ‘guessSet’ which contains all possible words in English of such length, and a ‘secretWordSet’ which will include words of that length that are allowed to be the secret word. The difference between the two arrays is that ‘secretWordSet’ does not contain plural words or extremely hard words. As the user is playing the game and typing out their guess, it will be matched against the secret word. The tile will turn green if the letter is in the correction position, the tile will turn yellow if the secret word contains the letter yet the user placed it in the wrong spot, or it will remain gray indicating that such a letter is not contained in the secret word. The keys on the virtual keyboard will transition simultaneously. Another feature that the user is able to use is the drag and drop functionality that has been implemented to assist in the guessing process. At the end of six guesses, you user either won or lost which will be indicated by a pop-up on the screen which does contain a play again button. If the user decides to play again, the cycle repeats.

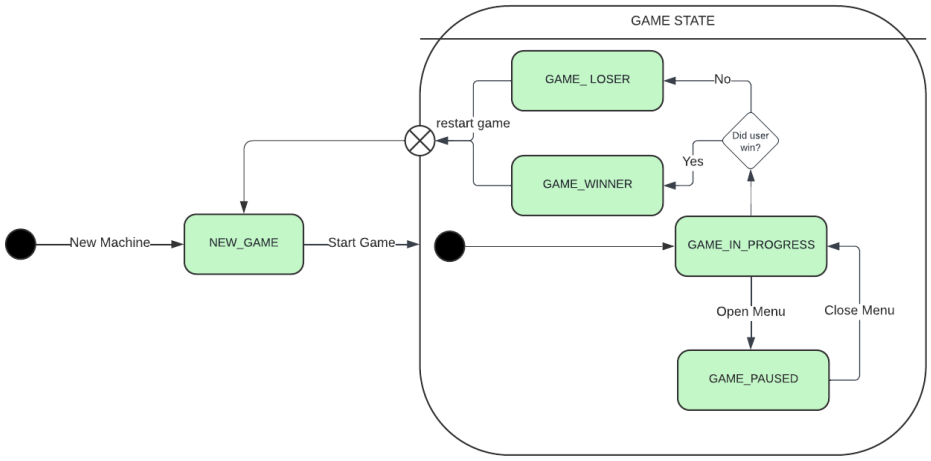
[User Personas]

1) Jean-Michel Mathieu, 32 years old, competitive player
    - “ I like a challenge.”
    - Jean-Michel is someone who thinks the game itself is very easy and wants to play hard mode, where words are more limited.
2) Jose Pastor, 17 years old, competitive player

- “ I don't keep the score, I just keep scoring. ”
- Jose likes to be challenged more and more over time to see how far they can get with the game. Someone competitive who likes to stack numbers.

3) Juliette Masson, 21 years old, social player
   - “ I lean towards games that have really awesome and unique designs. When I play games with just simple black and white GUIs I find myself not too intrigued by the game. ”
   - Juliette likes choosing a colorful design of the games they play.
4) Megan Foster, 29 years old, social player
   - “I love to play games every day that have an easy user interface.”
   - Megan wants to keep up with the trends and play a popular game but doesn’t want too much of a challenge.

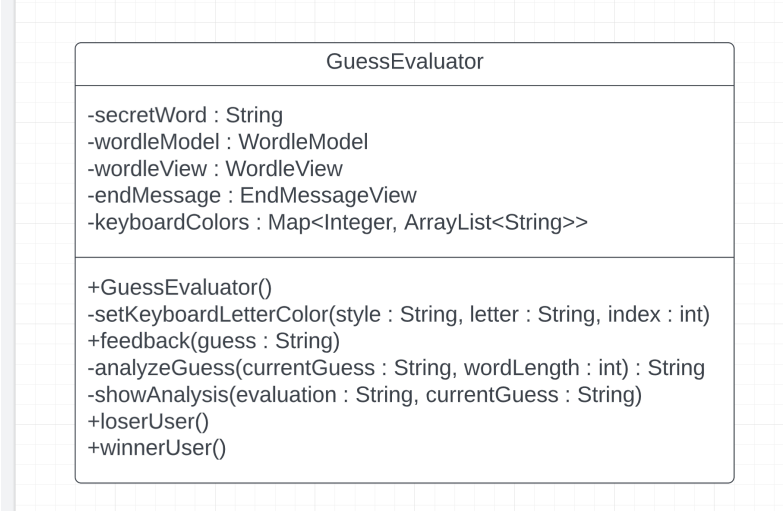
[Object-Oriented Design]



The game state diagram shows the different states our game goes through based on how the user interacts with the game. The game begins with the NEW_GAME state every time the player starts a new game. Once the game has started the game updates to GAME_IN_PROGRESS where the user is allowed to type on the keyboard and open the setting menu. Once the setting menu is open the game state is updated to GAME_PAUSED, so the user will not be able to type to the tiles while being in the menu. After they close the menu, the game state switches back to GAME_IN_PROGRESS. At that point, the user continues to play the game until they are up to

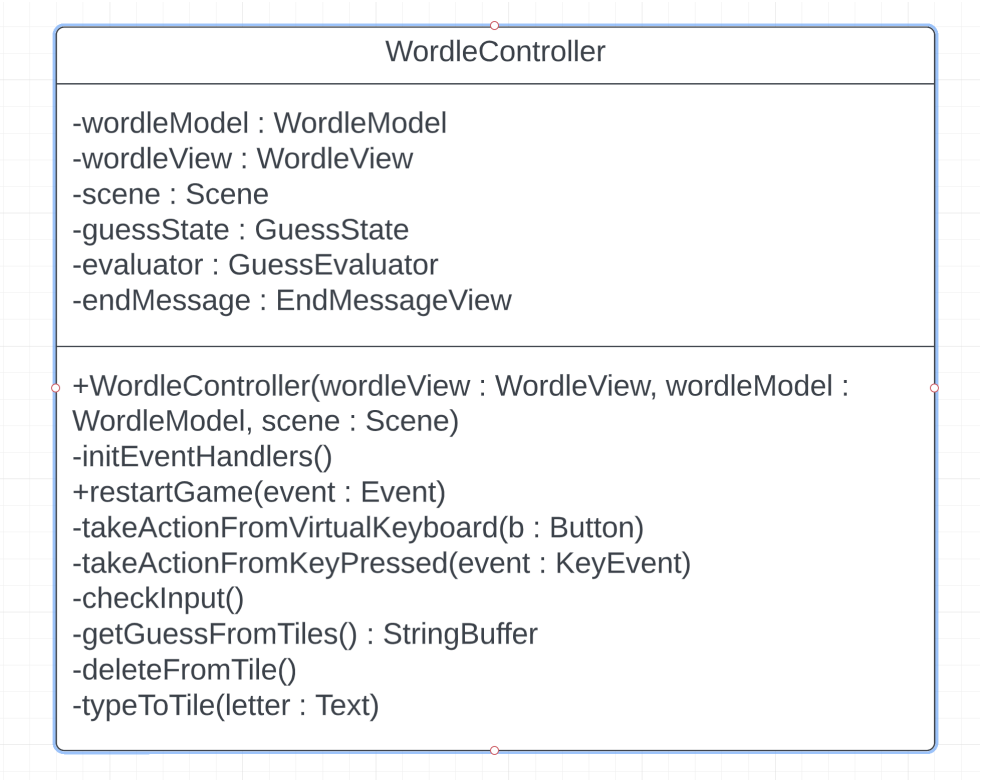
their last guess or if they win the game before the last guess or they end up loing. If they win, the state alters to GAME_WINNER or GAME_LOSER if they lose. Once the game ends, the user is given a pop-up which announces if they have lost or won, then they are prompted to play again. If the user chooses to play again, the game state switches back to NEW_GAME and the process repeats. If the user decides to exit out of the game, the last state remains based on the result of their last game, GAME_LOSER or GAME_WINNER. However, if the game is reopened, the default state will be NEW_GAME.

To give quick intuition about the project, a BorderPane is used to create the main screen where we set the top to be the header, the tiles to be the center, and the virtual keyboard to be the bottom. The header is simple, just a label with some buttons on the left and right. The tiles are styled but in essence they are all labels that can be typed to. Lastly, the virtual keyboard is all made of buttons that can be clicked and text from them is displayed on tiles. That makes the design very simple, and users can play as much as they wish. Making the interface user friendly takes care of the fourth user persona.

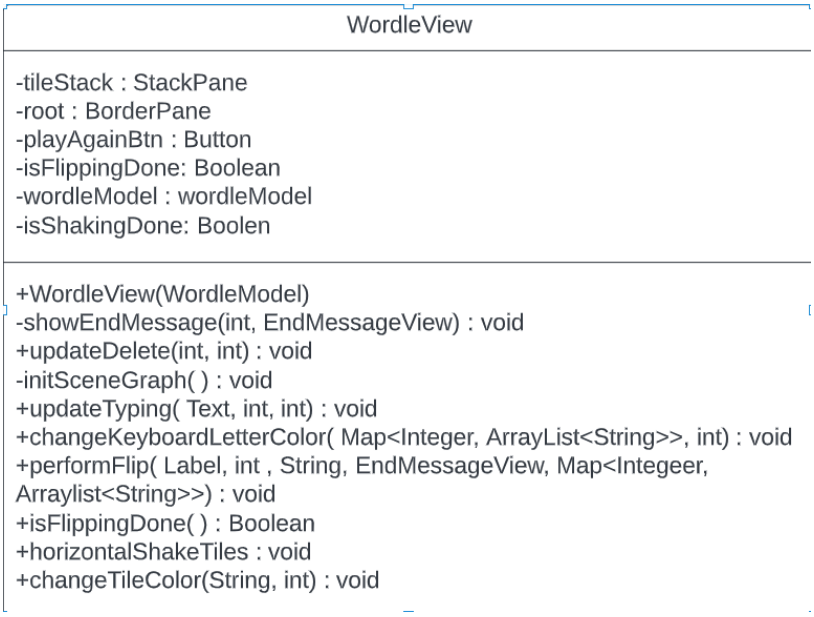


This class is particularly important for analyzing the guess given by the user. More specifically, the method analyzeGuess. The method starts out with a StringBuffer of all dashes based on the word length chosen by the user. Next, map the currentGuess to the secretWord using arrays. If a letter is placed in the correct position, set the dash at the letter index to be *, which will later on

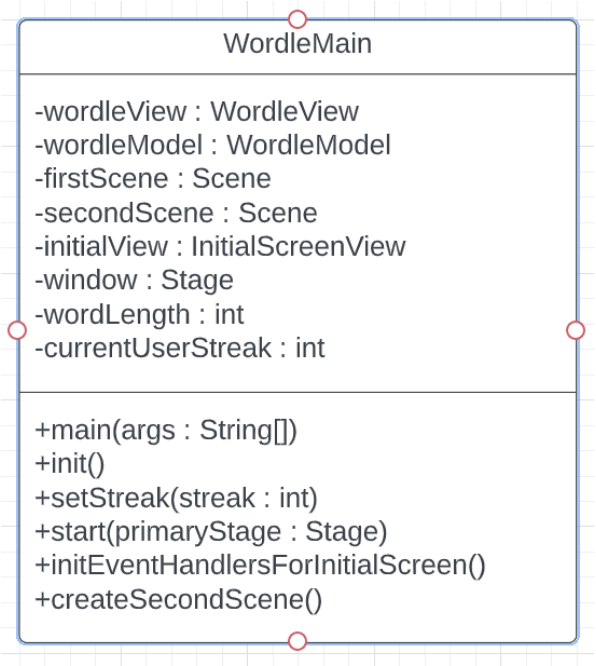
translate to green on the tiles. Make sure to remove the letters from both arrays. Next, it maps the remaining letters of the guess and secret word to find any misplaced letters contained in the guess, and replace the respective dash to be a +, which will later translate to yellow on the tiles. Lastly , any remaining dashes will translate to gray on the tiles, meaning the letter is not contained in the secret word. The transformed StringBuffer is then returned as a string.



The controller is key to this project. In the eventHandlers() method, it will capture clicks on the virtual keyboard, typing on the physical keyboard, and also clicking the settings button. In addition, the “takeAction” methods will bind the letter typed/clicked to the tiles. Key conditions in the controller are checking if an input has enough letters and is present in our set of five letter words, and not taking action if the backspace is clicked when the current guess has length zero. The method getGuessFromTiles() is crucial so the given guess can be used in the view, and effects such as flipping and changing tile colors can be performed with success. Lastly, restartGame() will ensure to keep the streak increasing in the model in case of a win. That will take care of our second user persona.

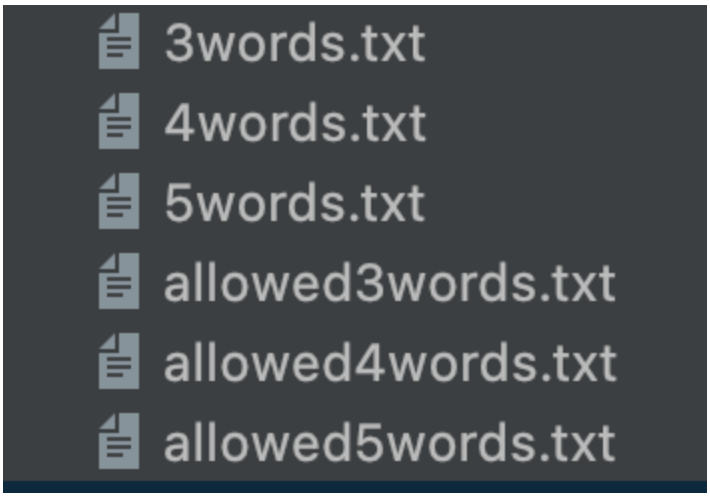


The view is extremely crucial in this project. As the methods suggest, the view is responsible for updating the typing and deleting on the screen, flipping the letters or shaking the letters, changing the color of the tiles as well as the buttons on the virtual keyboard. That said, the view is responsible for any kind of animation, but also basic displaying of the screen.



The WordleMain method will take care of the wordLength chosen by the user. Users can choose between 3-letter mode, 4-letter mode, and 5-letter mode, which represents hard, medium, and easy modes respectively, since the shorter the length, the less letters you can get to know about based on the secret word. This takes care of the first user persona.

This project also ensures to differentiate between words that are allowed to be guesses, and words that are allowed to be the secret word. That said, users will have a variety of guess options, but extremely hard words or plurals won't be picked to be the secret word. Take a look at the structure →



Lastly, users can choose to switch between light mode and dark mode by simply clicking a button. That takes care of the last user persona, number 3. Unfortunately, we didn't implement several themes as we initially desire, but at least there's a light and dark mode that partially suffices the need of user persona 3. Keep in mind that for this implementation, the event of clicking on the button must be handled in wordleMain, where you have access to the second screen. The dark mode is done by reassigning a new css file to the scene.