David Yao
Alex Cole

Computer Science 111 final project

The module reflexgame.py contains six functions, **screenInitialize, instructionScreen, scoreScreen, difficultyScreen, reflexGame, readScore, and main**, and one class, **Buttons**. The modules **pygame**, **datetime**, **time**, and **random** are imported. reflexgame.py requires the folders game_sound and game_images in order to function.

The **Buttons** class defines the properties of the buttons used to call many of the other functions. The method **buttonText** allows text to be displayed overtop of the button when the button is called in a function. The method **highlightButton** takes the mouse position and uses it to determine whether or not to set the **shouldHighlight** method to True, and in turn tells the method **highlightColor** to change the button color appropriately. Finally, **makeButton** allows the button to be displayed on screen as a rectangle, and also allows the mouse to interact with the button.

The **main** function simply gets username as input and assigns it to the appropriate global variable, USERNAME. It then calls the primary function, **screenInitialize**.

The **screenInitialize** function takes no arguments. It initializes pygame and allows the user to call **instructionScreen, scoreScreen,** and **difficultyScreen** by clicking on the appropriately labeled button and therefor initiating the appropriate function call. While the player is on the screen, a while-loop allows the window to stay open. The background image on the screen is called from the game_images folder using pygame. When the player clicks a button, a new window opens and the while-loop is broken, allowing the screenInitialize to close. This prevents the player from having to exit out of multiple windows and conserves processing power. If the player chooses to quit out of the application, exiting the window closes the while-loop without opening another window.

The **instructionScreen** function takes no arguments. It is called via the “Instructions” button on the **screenInitialize** screen. Once called, it initializes pygame and starts a while-loop that keeps the window open, and displays an image containing the instructions, called from the game_images folder using pygame. Clicking the “Back” button calls the **screenInitialize** function, ending the while-loop and bringing the player back to the initial page. As above, if the player chooses to quit out of the application, exiting the window closes the while-loop without opening another window.

The **scoreScreen** function also takes no arguments. It is called via the “Scoreboard” button on the **screenInitialize** screen. Once called, it initializes pygame as well as the **readScore** function. It then starts a while-loop that keeps the window open. Using the global variable SCORELIST, it gets a list of the top five scores. It iterates through the list using a for-loop and uses pygame displays each of the top five scores on the screen as they are pulled from the list. Clicking the “Menu” button calls the **screenInitialize** function, ending the while-loop and bringing the player back to the initial page. As above, if the player chooses to quit out of the application, exiting the window closes the while-loop without opening another window.

The **difficultyScreen** takes no arguments. It is called via the "Play" button on the **screenInitialize** screen. Once called, it displays several buttons. The "Beginner", "Novice", and "Expert" buttons each set one of the corresponding global variables (EASYMODE, MEDMODE, HARDMODE) to True, while setting the variables to False in order to prevent conflicts. The background image on the screen is called from the game_images folder using pygame. If the player clicks the "Menu" button, **screenInitialize** is called and the player is returned to the initial screen. In this case, all corresponding global variables are set to False. If the player clicks the "Play" button, the **reflexGame** function is called and the current screen is closed. As above, if the player chooses to quit out of the application, exiting the window closes the while-loop without opening another window.

The function **reflexGame** is called from the **difficultyScreen** menu. It, too, takes no arguments. After loading the window and fonts, it loads all player character images, as well as the level 1 enemy images. It also loads the sounds, begins the level counter at 1, sets the time counter to 0, and sets the player-lose variable to False. Depending on the state of the HARDMODE, MEDMODE, and EASYMODE global variables, it adjusts the reaction time cutoffs of each level. A for-loop then begins the gameplay, looping five times before going on to the ending screen. During the for-loop, the level is checked, the appropriate enemy image is loaded, and the reaction time adjusted accordingly. A wind sound and musical cue is played following the level introduction screen, and another for-loop is initiated that plays a short animation for both the player and the enemy. Following the animation, and after a random amount of time, the player is prompted to react with an audio and visual cue. The screen then cuts to black and plays a slashing sound, along with a slashing image, before returning to the game screen. Depending on the difficulty and player reaction time, the lose-variable may remain False or be set to True, causing either a player victory image or a player defeat image to be displayed, and ending the for-loop in the case of defeat. During the for-loop, the screen will then cut to black and display the player's reaction time for that level, followed by the next level's introduction. Once the for-loop has completed, or the player has lost, the player's time, averaged over 5 levels, is displayed. The player's username, average score, and level completed are then appended to a text file, scoreDoc.txt. Appending is necessary to prevent the constant overwriting of old scores. Once the information has been written, the function closes the text document and shows a screen that asks the player to enter the score screen by pressing any key. Pressing a key calls the **scoreScreen** function, and closes the current window. As above, if the player chooses to quit out of the application, exiting the window closes the while-loop without opening another window.

The function **readScore** takes no arguments. It is called as part of the **scoreScreen** function. readScore opens the text document, scoreDoc.txt, and reads each line, splitting the lines into lists as part of 'raw', and then assigning the reaction time to the variable 'score'. By appending the 'score' variable to a list, and then sorting the list, the function is able to iterate through the list and compare the top five scores (the first five in the list), to the scores in the text document. An if-elif loop checks the number of scores in the text document, preventing range errors from occurring if there are less than five scores. Another if-loop checks to make sure that the item is not already in the global variable, SCORELIST. If the item in the list is unique, it is appended to SCORELIST as a string.

The module should be run from the command line as "python3 reflextest.py". Username should be given when prompted. All related folders (game_images, game_sound) should be stored in the same directory as the module. The module requires the latest version of pygame to run correctly.