Cameron Jewell, Paul Ryu, Lance Little

5/3/18

Wireless Software Engineering

Term Project

Project Synopsis

Our app, Simon, is incomplete. We ran into significant difficulty with user and system synchronization. The game is complete in facets including Main menu, Settings menu, and Score menu. Each of these pieces functions as desired. The issue, however, arises with the Play Game portion.

When the game is begun, the system creates an ArrayList in which to store the randomly generated pattern the player will have to memorize and re-enter appropriately. The system is designed using Handlers in order to delay the changing of button resource colors. Once a color is added to the pattern, it is “flashed” using the method flashButton(int color). This function changes the appropriate button to a brighter, distinguishable color to indicate the first (next) color in the pattern sequence. Ideally, the player would wait for the pattern sequence to finish displaying and then repeat the sequence back to the system to be stored in an ArrayList called userInput. The two arrays pattern and userInput are then compared for exactness. If they are the same, the user has successfully entered the pattern and the score is incremented and the process is repeated, adding another color to pattern. If the score is higher than the system high score, the current score is written to the high score (this is done after every successful pattern entry). However, the pattern is not displayed correctly due timing errors that we have not been able to correct.

The main method, simon(), is broken into two sections: the system output (adding to and displaying the pattern) and the user input (waiting for user input of the sequence and comparing to the actual pattern). Each of these sections are given an integer representation in the variable gameMode; gameMode = 1 represents the system output state, gameMode = 2 represents the user input state, and gameMode = 3 represents game over and score keeping. Each of these conditions is checked prior to entering its respective stage. This prevents any extraneous user input from being considered in the comparison between the userInput and pattern ArrayLists.

Beginning the Play Game sequence with four sample colors to test the functionality of the flashing of pattern colors, all of the colors in the pattern are flashed simultaneously leaving the user unable to discern both the pattern order and the pattern length. This is not intended. We cannot use Thread.sleep(time) (in the UI thread) as this puts the system into an ANR (Application Not Responding) state. Attempting a similar Handler approach (like the one used to flash individual buttons) requires a nested Handler to be declared on the fly with final access to iterative variables which was not possible with our setup. Attempting to brute force a solution using the system clock (System.currentTimeMillis()) in a while loop constantly checking for a delta time of 1-3 seconds depending on difficulty results in a system timeout also proved fruitless.

We soon realized the problems we were facing were outside the skillset obtained through this class and unfortunately would require more time than we had at our disposal.