Team 5

Jake Marotta:188004016

Albert Joseph: 187003433

Joshua Ratcliffe: 185002996

Owen Heller: 186006160

Srinu Koritela: 188003926

Gagan Kattula: 190007626

Final Project Write-Up

We started working on the project very early by communicating with our group members electronically and deciding upon the first meeting. After the meeting, we proceeded to plan out the development process and figure out what we wanted our code to do. This was done by writing out pseudocode to indicate the general ideas behind what we planned to do. Loops and conditional statements played a big role in our initial plan. A “while” loop would advance each question with a counting variable inside of it. It was also clear that we needed to make a GUI that contained all the push-buttons and textboxes needed for the game, so we began to lay this out in MATLAB. We assigned roles for each one of our group members to alleviate the work each person would have to do. Ultimately, a major portion of the work was merging the general code with the pushbutton code.

After we made the GUI figure and added what we could from the code involving loops and if statements to the GUI code, we made global variable that would be used throughout the entire program. We had an array that contained the name of the push-button with the correct answer, a matrix containing all the names of the wrong answers, a matrix with the questions and answers, and a global variable that would be used for advancing the question. The initial focus was just on making sure that pushing the correct push-button would advance the program. Within the answer choice callback functions there were conditional statements that checked to see if the push-button pressed was the correct one for that question number. The array of correct answer choices and matrix of incorrect answer choices were used for this. If the question was correct the counting variable goes up by one and thus references the next index of the correct or incorrect matrices would be referenced.

Once we established that our code was able to correct identify the right sequence of correct answers, we set about implementing the lifelines. The Fifty-Fifty lifeline worked by using the handles.Visible structure on two of the push-buttons for incorrect answers. The Ask-The-People lifeline worked by displaying a textbox containing percentages for each answer, depending on which answer was correct. The Phone-the-Genius lifeline worked in a similar way to pressing the correct answer, except it disappeared after. Other features added after this were: the pushbutton to start the game, the message displaying the money won after the game is over, code to change the strings on the pushbuttons and textbooks for each question, and the pause functions to add the time delay after pressing a button.

Collectively as a team, we learned a plethora of new concepts. We learned how to utilize global variables within our code. These work great with push-buttons because they can be found anywhere in MATLAB. We needed to remember to identify them at the beginning of each push-button callback function. We also learned how to use GUI callback functions, and importantly, that they do not work within an enclosed loop or enclosed conditional statements. Another new Matlab tool we learned to use is the function “newline” that acts like the enter key on a keyboard, except it is used for strings. By experimentation, we learned what the parts of the GUI handles structure represent such as “Visible,” “BackgroundColor,” and “String." The structure handles.Visible was very helpful with making the lifelines disappear after use, operating the Fifty-Fifty lifeline, and displaying the end of game message. We also learned that to make the pause function work effectively throughout the entire code, “pause on” should be identified earlier on. After the code was finished, we tested the code and looked for bugs. People outside the group were asked to play the game and give us feedback. Through this process we found some issues such as the endgame code not appearing if you got the second question incorrect.

We encountered two challenges during the coding process. The first was figuring out how to use the push-buttons to advance the code. This took a long trial and error process and we needed to use utilize our problem-solving skills to make it work. The second was figuring out meetings. Due to everybody’s busy personal schedules and locations at certain times, finding a time and meeting place to gather proved difficult. However, this problem was resolved much quicker than the other. We were able to share the code and project updates with each other electronically, which allowed us to work individually.

If more time was allotted, our group would have added three more features to our code. The first feature we would have added are audio effects. For example, after a correct or incorrect answer, a jingle would play. Then at the end, if successful, music would play after the final correct answer. Second, we would have added a countdown clock to answer each of the questions. Finally, we would have added a way to change the questions for each time you play the game again.

Looking back at the coding process, we would have started working directly with the GUI code right away. This would have saved much more time altogether and made the debugging process a lot shorter. Also, this would have saved us from creating complicated code with multiple loops and conditional statements that were essentially useless.

Our group was very satisfied and proud of the way our code turned out. We completed all the objectives in the project rubric under the A deliverables and added a few more features. A great deal was learned and accomplished. Our sources of information on things we were unsure about were the help function in MATLAB, lecture slides from class, and question and answer sessions during class.

Honor Statement:

On my honor, I have neither received nor given any unauthorized assistance on this assignment.

*Joshua Ratcliffe, Jake Marotta, Albert Joseph, Owen Heller, Srinu Koritela, Gagan Kattula*