Report for Economic Analysis with Matrices Group Project

**Group name:** The Algebros

**Group project title:** Countdown Gameshow

**Group members:**

1. Chris Finch
2. Callum Donnelly
3. Harry Goodwin
4. Will Vaughan

Introduction

When forming our group for this project we began brainstorming possible ideas for our project, based on our own ideas and previous projects. Some of our ideas included a Football Score predictor in line with the ‘Super 6’ football game, a game similar to flappy bird, a version of the TV gameshow ‘The Chase’, and a version of the TV show ‘Countdown’. After discussion with Antonio at our introductory meeting, we decided on creating the ‘Countdown Gameshow’ as we believed that this would be achievable in the time limit, we all had prior knowledge of the format of the game and could also use some of the Matlab skills we had acquired such as for loops and a variety of functions.

First stages

The first part of the project required all of the group members to create Github accounts, and to become familiar with how Github, Github desktop and Matlab were integrated. We became aware of the ability to use GUIDE to create our Graphical User Interface (GUI) for the game and began using this feature to build the Countdown GUI.

Our concept of the game would be extremely similar to the TV version, the player would choose 9 letters, each time they chose a letter it could be either a randomly chosen consonant or vowel depending on their choice. After the player had received all of their 9 letters, a 30 second timer would start to count down. The aim is for the player to create the largest letter they can using only the given letters in the 30 second time limit. They would have to press a submit button before the 30 seconds are up for their word to be processed. The word would then be checked against a dictionary of words to check it is a real world, and if so their score will be recorded.

We aimed to split up the workload between us with certain members tasked with looking at different parts that would be needed, for example a timer object, the design of the GUI, the consonant and vowel buttons functions and many more. In the working situation, all team members collaborated on all of the areas of the code.

Code developments

Writing our code required us to do a lot of research on MATLAB features and functions that we had not come across in class, for example using the ‘Global’ function to define important variables across several functions. Furthermore, a tool that was extremely important to our code for the word check was to use ‘fopen’ in order to search through our list of valid words. These were all concepts that we had not seen before, but throughout our group meetings we researched areas of our game, and found the relevant functions tools on Matlab. We began creating the GUI using GUIDE by integrating push buttons, and text boxes and also inputted a countdown background image from the internet. We then started to develop the code in Matlab for the random letters for both consonant and vowel push buttons which were almost identical. We used several nested ‘if’ statements in order for our code to produce a randomly chosen consonant or vowel from the list that we would have in another part of code. The reset function created was necessary to reset the game, this meant that the number of consonants and vowels would reset to zero, along with the timer back to 30 seconds in the GUI. One part that required a lot of development was the word check, which linked to an external file that contained the list of valid words that we had, and returned a message box which would either say that the word was invalid if it did not match, or gave a score for the number of letters that the word used if it did match. To do this we used the statements ‘fopen’ (which would open the external file), ‘textscan’ (which would search through the strings of text in the file) and also ‘inputdlg’ (this would take the word input and create a message box that will appear on the screen in the case where a word is valid or invalid), as well as a for loop which uses ‘strcmpi’ which compares the letters of the string to the valid word. Other parts of the project were simpler, such as the component for calculating the player’s score, which only required us to use the length statement to calculate how many letters were in the answer.

How to run the code and play the game

The game is contained in one GUI and can be run using these instructions as seen also in the ReadMe file:

1. The whole zip file should be downloaded from the master branch (all files inside the master are necessary to run).
2. Make sure all of the files are in your workspace and open the ‘RunMe.m’ file.
3. Once opened run the code. This will open 1 of 2 separate GUI’s which are formatted differently and compatible with the operating system which your computer runs on.
4. Once the GUI comes up, the player can begin by either choosing a consonant or a vowel, it does not matter which they choose.
5. Repeat choosing a vowel or consonant for the remaining 8 letters (subject to constraint of a maximum of 6 consonants and 5 vowels).
6. As soon as all of the letters have been chosen the 30 second timer on the screen will start automatically.
7. The player should then come up with the longest word possible using the letters on screen.
8. Before the 30 seconds have reached 0 the player should type their answer in the text box.
9. They should press the submit button also before the 30 seconds have reached 0 in order for their word to be processed. It is very important that players remember to do this before the time is up or the word will not be accepted.
10. If the word is accepted and is in the dictionary then a message will pop up asking for the player to enter their name.

Limitations during development

* A problem that we experienced when testing playing the game was that a lot of the time, letters kept on appearing multiple times in one round which made it very hard to get a high scoring word. To solve this problem, we altered the game from the original version by decreasing the probabilities of certain letters (mainly letters such as Z, X etc) from appearing by lowering the number of these letters when defining the consonant and vowel options in the string of letters as seen in ‘reset.m’.
* One of our aims was also to include a feature where the letter check, instead of being one that checked through a list of words, checked an online dictionary instead for valid words. However, we felt that it would be simpler to have a file containing all the valid words and then do a check through this using the statements like ‘fopen’, ‘textscan’, etc so we decided on this option instead.
* A further problem we faced was to make the code compatible on both Mac and Windows, when testing we realised that the appearance wasn’t always the same so we had to make some changes. We altered the appearance of the GUI’s in GUIDE and then we used an if ‘ismac’ statement, along with a ‘elseif ispc’ statement in our ‘RunMe.m’ file, to determine if the player’s operating system is MacOS or Windows. The functions under the statements will then run the GUI which is configured for MacOS or Windows.
* Further to the above point, none of us had a machine which ran on Linux so we used a virtual machine to test the GUI on Linux and found it ran the same as it did on MacOS.

Mark Split

All group members contributed to the project in many areas, but some were more involved in the Github side of the project and so we have decided to split the marks accordingly.

|  |  |
| --- | --- |
| **Name** | **Mark split** |
| Chris Finch | 27 |
| Callum Donnelly | 25 |
| Harry Goodwin | 24 |
| Will Vaughan | 24 |