2048 Game – Terminal Application

AdamDobson_T1A3

Feature 1 – Interactivity

As the outcome of this game is almost completely reliant on the input of the user, it allows the user to have true control over the direction of each move and ultimately the outcome of the game (win or loss). This high level of interactivity and control gives the user a deeper level of immersiveness throughout each and every game. Along with this, the controls of this game are easy to understand as there is only 4 options of movement available to the user - these being 'w' (up), 'a' (left), 's' (down) and 'd' (right). Upon starting the application, the user is prompted to read through some instructions to gain a better understanding of how 2048 works if they are unfamiliar with the game. The aim of this is to lessen the learning curve of the game for the user. If the user enters an invalid input, they will be prompted with message that asks for a valid input. Upon achieving 2048, the user is given a congratulatory message to make them feel victorious after beating the game. On the contrary, if the user runs out of spaces on the board for new numbers to spawn, the user is given a message of defeat. Overall, these interactive features and controls work together to enhance the user experience.

Visual Board

Perhaps the most prominent feature of this application is the visual depiction of the game board/grid throughout the entirety of the game. This allows the user to actually see the current state of the game, and make their next move accordingly. Between each move the game board will dynamically update and display, giving the user visual feedback to their choice of move as well as the position of the newly added number between each move. The visual game board was developed using a list of lists (4 lists in the board used to represent the rows, and 4 numbers in each row. Ultimately, these rows stack atop each other to form a 4 * 4 grid of number). Through the use of loops, the program will scan through every number on the board and locate the number with the largest length in characters. This is required to ensure uniformity in the widths of all columns in the board, hence making the board far more readable for the user. This function allows for the column widths to automatically and dynamically update throughout the game.

Random Placement and Generation of Numbers

Through the utilisation of the random module, between each move this application will randomly generate a number (either 2 or 4) as well as a random placement of this number in a random empty space on the 4 * 4 grid. This element of randomness adds a degree of luck, excitement and improvisation to the game. As the amount of empty spaces increases and decreases throughout the game, it was important to create a function that automatically scans through all the possible empty spaces on the board in between each move, and only places the new number in one of those empty spaces. It was also important to ensure that two numbers are randomly added to the board at the very start of the game, before the first move. This is to allow the user to actually get the game started and merge these two numbers together. However, after the first move we must only add one new number in between each move. This meant that we needed to create a function that could take these two scenarios into account, and add the respective amount of new numbers to the board.