2048 Game Documentation

Introduction:

This game was developed to practice the developer's newly equipped knowledge of the Swift programming language. It is not for sale or download on the App Store. Anyone looking to start with the Swift programming language are encouraged to import this project to Xcode and try running it on the device.

Design:

Design for the project was done before the beginning of the development stage. The design was done on Adobe XD. Inspiration was taken from existing 2048 games but to implement some kind of differentiation between existing games and the developed one, a cleaner interface with no borders was used. While it might seem a bit confusing at first, it was done with the intention of seamlessly blurring the tiles together, giving it a cleaner look while playing the game. The colour palette used during the design, along with the Hex codes were copied and pasted to Xcode while development to achieve a nearly equal output in terms of colour reproduction for each of the game elements. It is the developer's first attempt at software design and development.

UI Development:

The development of the UI was fully done with the help of Interface Builder on Xcode using the design reference previously created with Adobe XD. Labels, Buttons and Stack Views were used accordingly to perfectly reproduce the design that was created during the design phase.

OutletCollections were used for the tiles of the game. This helped in easy referencing during development of game logic. The score and high score were created with label elements. Two new options previously not planned with the design were introduced - The New Game and Menu option. Both were deployed as Button elements. Outlets were created for all the mentioned elements in the corresponding Swift file.

Separate View Controllers were created for the game and the menu views. Each of the files held the required outlets for their view and executed corresponding logic. Modal appearance was used for the segue as no navigation controllers are used. The "X" buttons on the corresponding views were used as Unwind Segues to return back to the previous screen.

The elements were constrained appropriately with AutoLayout and rigorous cross verification was done to ensure proper appearance of the elements on the following devices: iPhone XS Max, iPhone XS, iPhone XR, iPhone X, iPhone 8, 7, 6 Plus, iPhone 8, 7, 6, iPhone SE and 5S.

iPhone 4s and iPads are not supported.

Game Logic Development:

The game logic was developed completely by the developer and no part of the code is borrowed. An integer array was created to hold the values equivalent to the value on each tile in the game. Empty tiles were assigned a value of 0 in the array. The array was primarily used for modification of value during a move and at the end of each move, the values of the array were copied to the tiles to reflect on the user interface. This ensured avoidance of type conversion errors that would be otherwise created due to the text Labels of each tile.

The whole vertical stack view holding all the tiles was chosen as the gesture recogniser. Four gestures were caught and handled by the recogniser: Swipe left, swipe right, swipe up and swipe down. Each gesture was handled by their corresponding function.

Loops were used within the function to identity and modify the value of the tiles. As mentioned earlier, all additions and deletions of the values were done on the integer array and after the termination of the loops, the integer array values were copied to the tiles to reflect the changes in the interface.

Two functions ran after the completion of each move. The checkLoss() function was performed to check if a move was possible next and if not, it led to the loss screen from where the user could choose the retry button. The checkWin() function was performed to check if the 2048 tile was arrived at after the completion of a move. If the 2048 tile was created, it led to the Win Screen which congratulated the user and gave the user the option to either “Keep going” or to start a “New Game”.

Persistence:

The game supports persistence. Plist files are used for the storage of data after the termination of application. The values stored in the plist file are: the integer array that holds the game tile values, the highest score so as to reflect the value in the high score label, the status of the game.

Every time the application is launched, the application loads the plist file stored on the phone and checks if a game was previously terminated without completion. If that is the case, the values of the array are loaded back into the tile so that the game can be continued from where it was stopped. If not, a new game is started. After the user loses a game, a function is performed to check if the score of the user is greater than the high score and if it is greater, the high score value in the plist file is updated accordingly.

Conclusion:

The development of 2048 was moderately challenging for a beginner in Swift programming. It was a great learning experience however, and reaffirmed the learning of many of the fundamental concepts in Swift and will serve as a great experience for the preparation of the development of many more complicated applications for the iOS world.