COMP228 Assignment 1 - The Chase Document

Start View

The start screen has two buttons: one to start the game and one to view the leaderboard.

Game View

When the game is started, players are asked to select an offer. This selection will also define their starting position on the ladder.

The cells of the table are defined in sections, not rows. This is for aesthetic purposes - it allows the cells to have rounded corners, as well as spaces in between them to make movement between cells clearer as well. Each colour used for the ladder is taken from the TV show. The table used for the ladder is fixed to prevent user scrolling.

The high offer is randomised between £10,000 - £50,000, rounded to the nearest £1000.
The middle offer is randomised between £2000 - 8000, rounded to the nearest £500.
The low offer is randomised between £100 - £1000, rounded to the nearest £100.
Each offer is rounded to present the user with neat numbers to choose from. The gap between the highest number of one offer and the lowest number of the next highest offer is double the higher offer's rounding point. For example, the middle offer's rounding point is £500, and the gap between the low and middle offer is £1000. The high offer's rounding point is £1000, and the gap between the middle and high offer is £2000. This is to make it feel like there is enough of a difference to potentially want to go for the higher offer if two offers are close in value.

Once the offer is selected, the ladder is initialised based on player position, and the first question is generated. Once the player picks an answer, a dark blue token will appear alongside their choice, and a red token will appear next to the chaser's choice. Despite the chaser picking at any time regardless of player action in the show, in the game, if the player runs out of time before picking a choice, the chaser choice will not show until time is up, to prevent the player copying the chaser's answer.

To simulate the chaser picking a response, I first randomised a number between 1 - 4. If the random number was between 1 - 3, the chaser was correct, and their answer is the same as the correct answer read in from the JSON questions. However, if the random number was 4, this was their 25% chance of being incorrect. To avoid a repetitive choice pattern whenever incorrect (such as incorrectly choosing 2 if 1 was correct, or choosing 3 if 2 was correct, for example), I coded their incorrect choice as follows:

- I defined an array of the answer choices as [1,2,3] inside the chaserPicksAnswer method, to avoid the array being permanently altered between turns.
- I then removed the int identical to the correct answer, to prevent the chaser from being able to pick it.
- Finally, I let the chaser's answer be a random selection between the two answers left.

This method made the chaser's incorrect choices feel more organic.

Three seconds after choosing, or after the player has run out of time - the answers are highlighted (correct = green, incorrect = red). After another second, the ladder updates based on player and/or chaser movement. Both the player and the chaser have a pointer for their position on the chase ladder. If they get a question correct, their pointer value increases by 1, allowing them to move 1 space further down the ladder. If they are incorrect, they remain in the same place.

If the chaser and player pointers are ever the same value, the player has lost, and the view is segued to the end screen under losing circumstances, regardless of their position on the ladder. While the player pointer is lower than the total steps on the ladder, they move down and a new question is generated after one more second. This gives users a chance to register the change in the ladder without feeling like they've lost time for the new question. At this point, the answer colours are reset and the answer tokens are hidden.

If the player pointer becomes equal to the number of steps on the ladder, they are now transitioned to the end screen, under winning circumstances.

End View

If the player won, this information is passed in from the game view and a congratulations message is shown on the screen, as well as their score (so they know what to look for on the leaderboard in case they forgot at any point.). If they lost, a "Better luck next time!" message is shown instead, and their score isn't presented.

From here, users can either click a button to replay the game, restarting from the offer selection screen, or click a button to view the leaderboard, same as the start view.

Leaderboard View

Depending on which screen the user segued from (start or end), different 'back' buttons are hidden, allowing users to return to the screen they segued from. For example, if the users segue from the start view, the 'back to end view' button is hidden, preventing them from segueing to the end of the game instead. However, the two buttons are identical visually, and are in the same position, to prevent confusion and streamline the user experience of simply wanting to go 'back'.

The leaderboard table is dynamically sized. Each cell contains its position in the leaderboard table, and the score associated with that position. The latest score is added to the score array and the array is sorted in viewDidLoad, to allow the leaderboard to be presented in descending order.

If the player loses, their score is 0 and is not added to the leaderboard.

The leaderboard is saved as persistent data using UserDefaults.