Countdown - The Game

Joshua Hunt

yr18478@bris.ac.uk

January 10, 2019

1 Introduction

This python program aims to emulate the Channel 4 game show Countdown. In the game show 2 contestants compete to earn the most points on letters games, numbers games or conundrums.

During a letters round, one contestant chooses 9 cards, selecting a consonant or vowel for each. Both players then have 30 seconds to find the longest word they can [1]. The words are checked by 'dictionary corner' who then quote the longest words they could find.

During a numbers round, one contestant selects 6 numbers: between 1 and 4 large numbers (25, 50, 75 or 100) with the remaining numbers being between 1 and 10. A random target number is selected between 100 and 999. Using each number only once along with simple arithmetic operators, the contestants try to get as close to the target number as possible (no decimal or negative intermediate results are allowed) [1]. Rachel Riley then checks the contestants answers and provides a model solution.

A standard countdown match has 15 rounds of 30 seconds each, comprising of 10 letter rounds, 4 numbers rounds and 1 conundrum [1]. To ensure this game doesn't enter tedium, a full game will consist of 6 rounds with 4 letter and 2 number rounds. Alternatively, players can chose to play a single round of letters or numbers.

2 Implementation

2.1 Utility Classes and Functions

I have used classes to experiment with the ideas of inheritance and object oriented programming.

Class: Text Box
This class enables a box of varying size to display text at any location on the screen. If the text variable is changed anywhere in the program, it will dynamically update the text. This is used for text output within the game.

Class: Input Box
This class inherits the display characteristics from the text box class and is extended to allow the user to select it and type their own message. I also implemented a cursor to show where the user is typing. The user input events are dealt with by the object in the update function. The pygame.event.get() function returns a list of the events and clears the internal event log. I found it best to call this function from one place and pass the event list as an argument to ensure the events were read and processed in the right place. This method does result in the input events being looped through multiple times, however it results in much cleaner code as the events are processed by the object, not the main loop of the program.

Class: Game Manager
This class contains the variables required to control the flow of the game and maintain the scores if a full game is played.

Class: Timer
This class adds a progress bar that takes a set time to finish (30 seconds) although this could be passed as an argument. While the progress bar runs, the Countdown music [2] is played. On each update the time elapsed since the last update is used to increase the progress of the bar.

Function: Check Exit
This function is called whenever there is a loop in order to check if the user has tried to exit the program either using the escape key or the close window button.

Function: Draw
This function is called to draw a list of sprites to the screen and helps to clean up the code.

2.2 Numbers Game

2.2.1 Selecting Tiles

To select the tiles, I define a list of all possible tiles (2 of each small number from 1 to 10, one of each large number, 25, 50, 75 and 100). After the player selects how many large numbers they want, I randomly select the appropriate number of large and small numbers from the lists (using pop to ensure duplicate numbers can't be chosen).

2.2.2 Playing the Numbers Game

The selected tiles are displayed along with a randomly generated target number. The timer class is used to display a progress bar with a limit of 30 seconds for the players to get as close to the target as they can.

2.2.3 Replacing Rachel Riley

The players are required to enter their working and this is checked using the built-in function eval(). If this succeeds I check that the correct numbers have been used and that no number has been used twice. I also return the score (10 for a correct answer, 7 for answers within 5 and 5 for answers within 10) which is added to the player's total.

To automatically find a correct solution, I use a brute force approach. There are 33,665,400 possible valid sums, however many of these are not necessary to calculate. If the running total is negative or fractional at any point, all subsequent sums will also be invalid and as such, they don't need to be checked.

Initially I used the eval() built-in python function, however this proved to be too slow so I looked for alternative options. To generate the possible combinations of 6 numbers and the 4 basic operators, I used reverse polish notation as described in [3] and [4]. I pass the sums as lists, with each element either containing a number as an integer or an operator as a string.

Iteratively increasing the number of elements in the sum allows the algorithm to discard lower branches as soon as a sum is invalid e.g. gives a negative or fractional value. Results that are less than 10 away from the target number are stored with the closest being displayed at the end in a user friendly format.

To speed up the calculation, I utilize the multiprocessing module. This allows multiple threads to process different starting numbers simultaneously. It initializes with one process less than the total core count of the CPU to allow the user interface to continue functioning.

To prevent the calculation locking the screen, I use the 'threading' module to set it as a background task. The module is called immediately after the tiles are selected so that it has the 30 second game time to compute the best solution. After the user enters their best solution, the program waits for the thread to finish (if necessary) before displaying the answer.

2.3 Letters Game

2.3.1 Selecting Tiles

I generate the arrays of consonants and vowels separately for the player to choose from. Each array contains all the available letters with an appropriate frequency. Using nested For loops I create the letter arrays each time the game is played.

2.3.2 Playing the Letters Game

Once the letters are selected, the timer starts and runs for 30 seconds. Then each player is invited to declare the length of the longest word they could find. They then enter their words and the game checks them against a dictionary file.

2.3.3 Replacing Dictionary Corner

The frequency of each letter is checked against the list of tiles and the score is calculated (18 points for a 9 letter word, otherwise the length of the word).

The findLongestWord function is called on a separate thread to the rest of the program. This allows the user interface to continue functioning as expected.

In order to find the longest word, I use the module 'itertools', specifically the permutations function. This creates an iterable object that contains the possible different orderings of the letters and uses less memory when compared to a list. I compare each of these to a modified dictionary of words from a github repository [5]. In the official Countdown version, the Oxford English Dictionary is used but this is not publicly available in a text format. If a better dictionary is found, it can easily be replaced. The words are stored in a list of dictionaries, sorted by the length of word in order to reduce the size of dictionary each word is compared with.

3 Discussion and Conclusion

In conclusion the individual elements of the game work well. Future improvements could include:

- Reducing the time taken to find the numbers solution by using a more efficient algorithm. If the time taken could be reduced to less than 30 seconds, the user would not be aware of any calculation going on.
- In order to create an authentic Countdown experience, the conundrum is needed as a third round type.
- Improving the user interface
- Reduce repeated code that was introduced when I extended the game from one player to two player

References

[1] [Countdown Game Show]
`en.wikipedia.org/wiki/Countdown_(game_show)`

[2] [Countdown Clock Music]
`televisiontunes.com/Countdown_-_Clock_Only`

[3] [Data Genetics Countdown Numbers Solver]
`datagenetics.com/blog/august32014/index.html`

[4] [Reverse Polish Notation]
`en.wikipedia.org/wiki/Reverse_Polish_notation`

[5] [Dictionary]
`raw.githubusercontent.com/dwyl/english-words/master/words_alpha.txt`