EQUATION GAME

Milaina Frydas

Highgate School, 12610

March 2023

Contents

[1 Analysis 2](#_Toc124677184)

[1.1 Problem Identification 2](#_Toc124677185)

[1.2 My Clients 3](#_Toc124677186)

[1.3 Problem Research 3](#_Toc124677187)

[1.4 The Proposed Solution 5](#_Toc124677188)

[2 Design 6](#_Toc124677189)

[2.3 Interface Design 6](#_Toc124677190)

[2.31 Main Menu Scene 6](#_Toc124677191)

[2.32 Game Scene 6](#_Toc124677192)

[2.33 Paused Menu Scene 6](#_Toc124677193)

[3 Developing the Coded Solution 7](#_Toc124677194)

[3.1 Development Cycle 1 7](#_Toc124677195)

[3.11 Aims 7](#_Toc124677196)

[3.12 Objectives Achieved 7](#_Toc124677197)

[3.13 Prototypes 9](#_Toc124677198)

[3.14 Testing 10](#_Toc124677199)

[3.15 Review 11](#_Toc124677200)

# 1 Analysis

## Problem Identification

There is a resounding desire, in today’s society, to have a more logical and intellectual mind- particularly in parents who want their children to have advantageous analytical skills, students who have to sit aptitude tests for university admissions or in a hiring process, and simply individuals who want to keep their minds in shipshape. This desire can easily lend itself to a computational solution, using a ‘brain game’. The way my game displays an achievable, desired ‘answer’, along with some other numbers/operations to choose from, compounded with the rules and other limitations, and subsequently waits on the user to achieve the desired number via the displayed numbers and operations, keeps one’s mind active and healthy while also ameliorating their logical skills and problem-solving skills.

My *Equation Game* is not only didactic but it’s also fun… meaning that many children would (and have) independently decide to play this game in their spare time. There are many brain games alike this (for example, *crossword* or *sudoku*) however my *Equation Game* is, in itself, different although, it has its similarities; it closely resembles the television show and programme *Countdown* (but with its own twist).

## 1.2 My Clients

The clients and demographic for this game would be players (with different experience) of computer games, children and their parents, and students who will be sitting university admissions tests (UATs)/aptitude tests. Due to this range in my market, the stakeholders will be a representative sample, ranging from children whose parents want to keep their developing minds active, to older students (17+) who would like to exercise their logical skills, and then an older demographic who might use the game as a pastime.

The young children I have sought are children who have played games before and are used to such technology, thus wouldn’t struggle with picking it up. Similarly, I sought older people who are familiar with games and would also grasp it with ease.

The youngest stakeholders, aged 6 and 5 have both grown up surrounded by technology and with accessibility to *iPad* games etc. An 18-year-old and a 17-year-old are my older clients; the first of which is an unexperienced gamer who is preparing for his mathematics-based university admissions test and would like to use my game to do so; the second is an experienced gamer who would like to use my game for leisure purposes. I have two middle-aged men as my oldest clients (of ages around 50) who are both looking to take up my game recreationally; the first is retired and would like to use my game as a precautionary measure for boredom prevention; the second is an older, experienced gamer who would use my game to take the edge off.

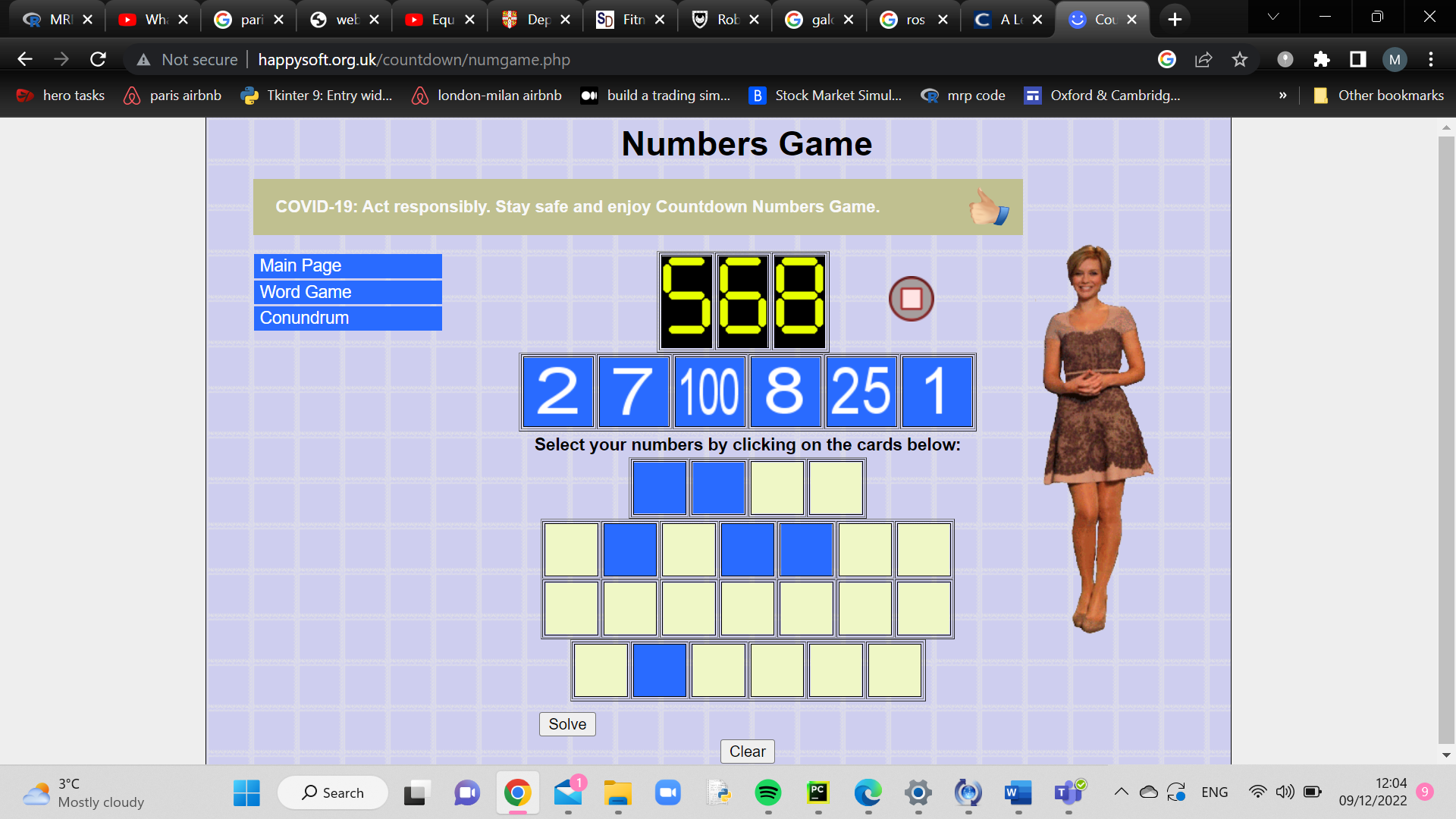
## 1.3 Problem Research

As previously mentioned, the crave for a more agile mind can, basically, be resolved via some computer-based method: brain games. Brain games, such as *Sudoku, Countdown,* and, *Nerdle* help improve cognitive function. These types of games typically involve challenges that require the player to use various mental skills, such as problem-solving, memory, and, attention to detail. By regularly engaging in these activities, the brain can develop and strength1en these skills, which can improve overall brain health and function. Additionally, brain games can be enjoyable and provide a sense of accomplishment, which can boost mood and reduce stress. Overall, engaging in brain games can provide a variety of mental and emotional benefits.

A screenshot of a computer

Description automatically generatedThe first example, *Sudoku:*

*Sudoku* is a popular logic-based number puzzle game that originated in Japan. The objective of the game is to fill a 9x9 grid with numbers so that each row, column, and 3x3 sub-grid contains all of the digits from 1 to 9. The puzzle begins with some of the squares already filled in, and the player must use logic to determine the correct placement of the remaining numbers. The game can be challenging and requires the player to think strategically and make deductions to solve the puzzle. *Sudoku* is considered to be a brain-training game and has gained popularity around the world. The element of *Sudoku* which I would like to replicate in my own work is the number simplicity- the numbers only range from 1-9 which keeps the arithmetic simple, allowing you to complicate other elements of the game.

The second example, *Countdown:*

*Countdown* is a British television game show that has been airing since 1982. The show involves a series of word and number puzzles, with contestants competing against each other and against a time limit to solve them. The format of the show includes a variety of rounds, such as the "Numbers Round" in which contestants must use arithmetic to reach a specific target number using a set of given numbers. This is the biggest influence of my game. They will be extremely similar with the some additional game rules in my own game – one of which being that you have to use all of the operations displayed per round, and another of which is the ability to press ‘shuffle’ and skip a round. However, overall, my game will follow the main concept of there being a target number and then a multitude of other numbers which can you can operate on in order to achieve the target number. The element of *Countdown* which I do not like is that there is never any certainty regarding whether or not the target number can actually be achieved given the patient numbers so in my game, I would like to ensure that to only display target numbers which can be achieved using the patient numbers.

Graphical user interface

Description automatically generatedA picture containing table

Description automatically generatedThe third example, *Nerdle:*

After 2 rounds

The aim of the game is to guess the ‘Nerdle’ in six tries, by guessing the equation that fills the eight tiles. After each guess, the colour of the tiles will change to show how close your guess is to the right answer. There is one ‘Nerdle’ per day. I like that *Nerdle* uses other (and more complex) operators like division and multiplication which I would like to incorporate into my game, and hopefully a squaring operator, too.

Outside of researching other brain games, I followed a Pygame tutorial on YouTube by *Free Code Camp* where I built a space invaders game. I really liked the way Pygame worked so I decided to employ it for my game. I also had interviews with each of my stakeholders to gather reconnaissance. After holding these interviews, this is what I understood an ideal game to be: a game that you can personalise each time you play so that it doesn’t get too repetitive otherwise it can become boring which is counteractive (especially for the older clientele who are looking for a solution to their boredom); a game that is easy to use and has an almost intuitive ‘rule-book’ because the students who are using these games while practising for UATs and such-like, do not have time to spend learning how to play a game… they want to pick it up while playing and have it all be a fast process; a game with some sort of leader board so that there is a competitive element to it and giving the younger children an incentive to play.

Taking all of these things into account, my solution is a software that targets one’s mathematical skills in doing the following: prints eight positive integers, compounded with a number which the user must make using at least two of the eight positive integers. The main restriction (and the main challenge) is that all of the operations displayed, must be used (minimum number of operations being one, and the maximum number of operations being four). You cannot use the displayed numbers more than once otherwise the game becomes fairly easy (easy being synonymous with boring). As mentioned earlier, I strongly dislike that in *Countdown* the target number’s achievability is never certain hence I programmed my game such that it generates a target number which always has a solution where all of the operators are used.

## 1.4 The Proposed Solution

Software Requirements:

|  |  |  |
| --- | --- | --- |
| Program/module | Version | Justification |
| Windows, Linux, or Mac OS | Most recent version | For python to operate on |
| python | 3.9 | Main programming language |
| random | Built-in-module | Used to generate the random integers which are displayed |
| Pygame | 2.1.2 | Used for the game functionality |

Hardware-wise, the only requirement is a mouse and/or touchscreen enabled.

Success Criteria:

|  |  |
| --- | --- |
| Criterium | How to Evidence |
| Main window appears when code is run | Screenshot of the main menu when |
| When level is selected on main menu, the game round runs | Screenshot of main menu running for each level selected |
| The numbers fall and in a calibrated manner | Screenshot of the main window 5 seconds after the program started running- the numbers should appear to be at a different position, still aligned |
| The target number is achievable | A valid equation should be returned to the shell every time there is a new target number |
| Colour-change when number is selected | When a number is clicked, it should turn yellow- screenshot of this |
| Operator disappears when it is selected | Before and after screenshots of an operator being selected |
| Shuffle button calls a new round perfectly | Screenshot of the new round after the shuffle icon has been clicked (it should look and function perfectly well) |
| Shuffle button brings you down a round | When the shuffle button is pressed, the number round should decrement by 1 – screenshots of round number before and after shuffle button is pressed |
| New round is automatically called when a valid equation is input by the user | Screenshot of game after the user has entered their equation |
| Game screen is shortened if user enters incorrect equation | Screenshots before and after incorrect equation has been entered |
| Game screen is shortened if user takes too long to input equation and numbers hit the bottom of the screen | Screenshots before and after the numbers have hit the bottom of the screen |

# 2 Design

## 2.3 Interface Design

### 2.31 Main Menu Scene

### 2.32 Game Scene

### Paused Menu Scene

# 3 Developing the Coded Solution

## 3.1 Development Cycle 1

### 3.11 Aims

* Main menu/ first screen appears when code is first run
* The numbers appear and fall correctly
* The calculated target number is achievable
* Operators disappear when selected

### 3.12 Objectives Achieved

A picture containing graphical user interface

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceI was successful in getting the main menu up and running (this wasn’t too difficult). Also the transition into the game was fine as it does not take that long for the game round to load up so no buffer screen was needed while the game loaded up. When it came to the speed at which the numbers were to fall, there were a few different versions I drafted using feedback from clients (regarding how long one should have to formulate the equation), before deciding on intermittently adding 0.03 to the y-coordinate-value.

*Position 2*

*Position 1*

My teenage clients had said that they believe 30 seconds would be the optimal amount of time; my older clients much preferred having somewhere from 50-60 seconds to formulate their equation. This feedback, compounded with the effect of my intention to introduce levels (this is discussed below) on the time needed to formulate an equation, led me to conclude that 50 seconds was the most suitable time to give the use to form their equation. The two figures (*position 1* and *position 2)* displayed above show where the numbers are at 0 seconds and at 50 seconds. Technically, the calculated target number is always achievable; I programmed an algorithm that calculates a number from the numbers displayed in such a way that the numerical ‘aim’ is always achievable.

A screenshot of a computer

Description automatically generated

Function to calculate the ‘aim’

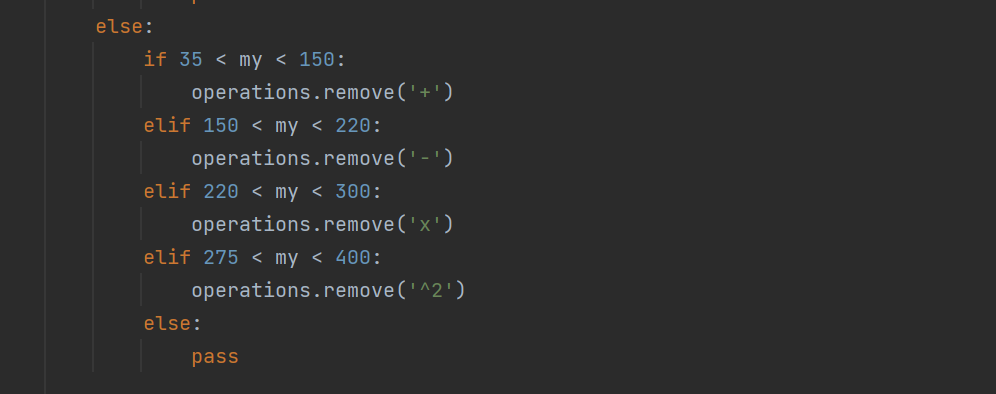
The above figure shows said function. The list ‘nums\_used’ is a list of numbers that have been randomly selected from the 8 displayed random numbers, in order to be used in the calculation of the ‘aim’. The list ‘shuffled\_operations’ is a shuffled list of the operations: addition, subtraction, multiplication, and, squaring.



The list ‘operation\_options’ is simply a list like so: [‘+’, ‘-‘, ‘\*’, ‘^2’]. So, the function calculate\_number iterates through the list of shuffled operations and performs them with the subsequent number in the list ‘nums\_used’ on the current aim, then updating the aim to become this new result. This keeps on happening until we have used all of the operations needed for the random selection if numbers (‘nums\_used’). As you can see, in the elif statement for when operation\_options index is 3, the squaring operation is performed on the number but the counter (i) is not incremented by 1- this is because squaring does not involve the next number in the list of nums\_used thus the counter does not need to be increased.

A blackboard with white writing

Description automatically generated with low confidenceAlthough this algorithm ensures that the ‘aim’ is always achievable, the ease of achievability for the average player was uncertain thus I asked different clients in what range they would like the ‘aim’ to lie; due to the variety in responses, I decided to create different levels where the ‘aim’ is of different calibres.

Achieving the disappearance of an operator upon being clicked was fairly simple: this was a matter of setting just boundaries and adhering to them.

As you can see above, there was an if statement following the reception of a mouse click where the ‘else’ part of the statement accounted for the coordinates that correlate with the positions of the operations. Evidently, if an operation were to be clicked on, the function ‘remove()’ would be performed on the list called ‘operations’ which is a list containing all of the operations currently being displayed in that round.

### 3.13 Prototypes

This was my prototype for the game screen which I have very closely adhered to:

Diagram

Description automatically generated with medium confidence

Then when creating the main menu I started off with my first prototype which is shown in *Main menu 1,* however later on, when I decided to implement levels I obviously created a different main menu prototype where you can select your levels and this is shown in *Main menu 2*.

Graphical user interface, application

Description automatically generatedGraphical user interface, website

Description automatically generated

*Main menu 2*

*Main menu 1*

### 3.14 Testing

I wrote tests that correspond to the test plan, to validate my code. Something I struggled with was importing sub-functions into the test file, however I overcame this by rewriting certain sub-functions outside of their parent-function which involved refactoring and substituting certain local variables.

This is an example of a test I wrote for the function ‘calculate\_number’:

Text

Description automatically generated

### 3.15 Review

Thus far the I have a main menu that appears when the code is run and smoothly transitions to the game screen upon selection of a level. When this game screen is run, the numbers fall together and the correct pace. The ‘aim’ displayed at the top of the screen is calculated correctly and I have ensured it is always achievable with at least one solution. When an operator is clicked on, it successfully disappears and so cannot be used again. My next steps involve correctly calculating the input equation from the user and seeing if the result of this equation correctly matches the ‘aim’; this will involve some boundary tests in order to validate an incalculable equation for example a user selecting the addition and multiplication operators sequentially. Also making sure the numbers successfully change colour and then disappear when selected.