2048 GAME

Team Members:

AM.EN.U4AIE20042 KONIJETI SRI VYSHNAVI

AM.EN.U4AIE20049 METHUKU SAMHITHA

INTRODUCTION

- 2048 is a single-player sliding tile puzzle video game written by Italian web developer Gabriele Cirulli and published on GitHub.
- This game is mobile compatible and you can play it on any device - iPhone, iPad or any other smartphone.
- Solving this game is an interesting problem because it has a random component. It's impossible to correctly predict not only where each new tile will be placed, but whether it will be a "2" or a "4".
- 2048 is an exciting tile-shifting game, where we move tiles around to combine them, aiming for increasingly larger tile values.
- If you beat the game and would like to master it, try to finish with a smaller score. That would mean that you finished with fewer moves.

HOW TO PLAY

- 2048 is a one-player puzzle slide game.
- It is played on a 4x4 grid.
- Each time after the movement it creates a tile with item 2 in it.
- When two tiles with the same number collide, they combine with double the value of the tile.
- To win, slide the numbers and produce a tile of 2048 or higher.

PART 1 : AM.EN.U4AIE20042

- I worked on how to initialize different functions like :
- **INITIALISING THE GRID :**
- I started by importing modules like “random”.
- Define a function called “new_game()”.
- Initialized "matrix" as an empty list.
- Created a 4x4 matrix by appending it into matrix after every iteration.
- Call “add_two()” and returned the matrix.

Adding a new 2 in a random empty cell

- Define a function called “add_two()” to add a new 2 in a random empty cell.
- Initializing variables a, b to get random positions in the matrix
- If the position has item “0” we add 2 into it. (mat[a][b] = 2)
- Return the matrix

Get the current status of the game.

- Define a function “game_state()” to get the current position of the game.
- If a random cell in the grid contains the element “2048” we return “win”.
- If a random cell in the grid contains the element “0” we return “game not over yet”.
- If 2 cells merge together and form an empty cell even then we return “game not over yet”.
- Else we return "lose".

Compress the grid after every step before and after merging cells.

- Define a function “cover_up()” to compress the grid.
- Initialise a boolean variable to determine any change happened or not.
- Initialising two lists “new” and “partial_new”.
- We shift the entries of each cell to its extreme left row by row.
- Traverse through each column in the respective row
- If the cell is non empty then we shift it to the previous empty cell in that row.
- Return the compressed matrix and flag variable.

Function to merge the cells in matrix after compressing

- Define a function merge() to merge the cells after compressing.
- If the current cell value equals to the next cell value we double the value of the current cell.
- Empty the value of next cell.
- Make the Boolean variable true indicating the new grid after merging is different.

Reverse the matrix : reversing the content of each row (reversing the sequence)

- Define a function reverse() to find the reverse of the matrix.
- Initialise a list “new”
- Append the elements into the list in the reverse order of every row.
- Return “new”.
- And Importing logic.py to 2048.py

PART-2 : AM.EN.U4AIE20049

- In logic.py I have worked in initializing different functions like:
- **<u>UP</u> : Initializing a key such that we can swipe/move up.**
- It is to define the function to update the matrix, if we move up / swipe up.
- Game is a matrix that is passed to a function that executes multiple function to preform up logic on the game. Multiple functions like transpose, cover_up and merge.
- After printing up then it will shift up so that it will return matrix.
- **<u>DOWN</u> : Initializing a key such that we can swipe/move down.**
- It is to define the function to update the matrix, if we move down / swipe down.
- Game is a matrix that is passed to a function that executes multiple function to preform down logic on the game. Multiple functions like transpose, reverse, cover_up and merge.
- After printing down then it will shift down so that it will return matrix.

- **<u>LEFT</u> : Initializing a key such that we can swipe/move LEFT.**
- It is to define the function to update the matrix, if we move left / swipe left.
- Game is a matrix that is passed to a function that executes multiple function to preform left logic on the game. Multiple functions like cover_up and merge.
- After printing left then it will shift left so that it will return matrix.

- **<u>RIGHT</u> : Initializing a key such that we can swipe/move RIGHT.**
- It is to define the function to update the matrix, if we move right / swipe right.
- Game is a matrix that is passed to a function that executes multiple function to preform right logic on the game. Multiple functions like reverse, cover_up and merge.
- After printing right then it will shift right so that it will return matrix.

- In 2048.py first, I have imported frame, label, center from **Tkinter. Tkinter** is the most commonly used library for developing GUI (Graphical User Interface) in Python.
- And also imported random, logic.

Generator Function

- Define a function "Generator Function"
- Generator function contains one or more yield statements.
- When called, it returns an object (iterator) but does not start execution immediately.
- It returns the random module in Python allows you to generate pseudo-random variables and the randint Python function is a built-in method that lets you generate random integers using the random module of (0, c.GRID_LEN - 1)
- Where grid length is 4.

GameGrid

- First I created a class and use the keyword class.
- The class including the __init__ function, hence it is important to declare self as the first parameter in those functions to hold the reference of the object.
- Since a Frame needs a container, I added an argument to its __init__() method and call the __init__() method of the Frame class.
- The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.
- I have passed self.master as the first argument and add a name and bind key down.
- Self.commands are some commands of logic up, down, left, right.
- I have defined the properties of a class as **self.[something]**. So, whenever we create an instance of the *class*, the **self** will refer to a different instance from which we are accessing class properties or methods.

init_grid

- Define a function "init_grid"
- The background property specifies an image to use as the background of an element. Where background frame color, width, height of the grid is mentioned.
- Next to that I created **the grid**
- The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.
- **Using self I used** append() method appends an element to the end of the list.

Updating color of grid and cell

- Define a function "update_grid_cells()".
- I used "new_number" to get the random position of the matrix.
- If the position has element "0" then return the color of 0 declared in the constants.
- Else we read the number in a position and print the required color from the constants.

KEY_QUIT and KEY_BACK

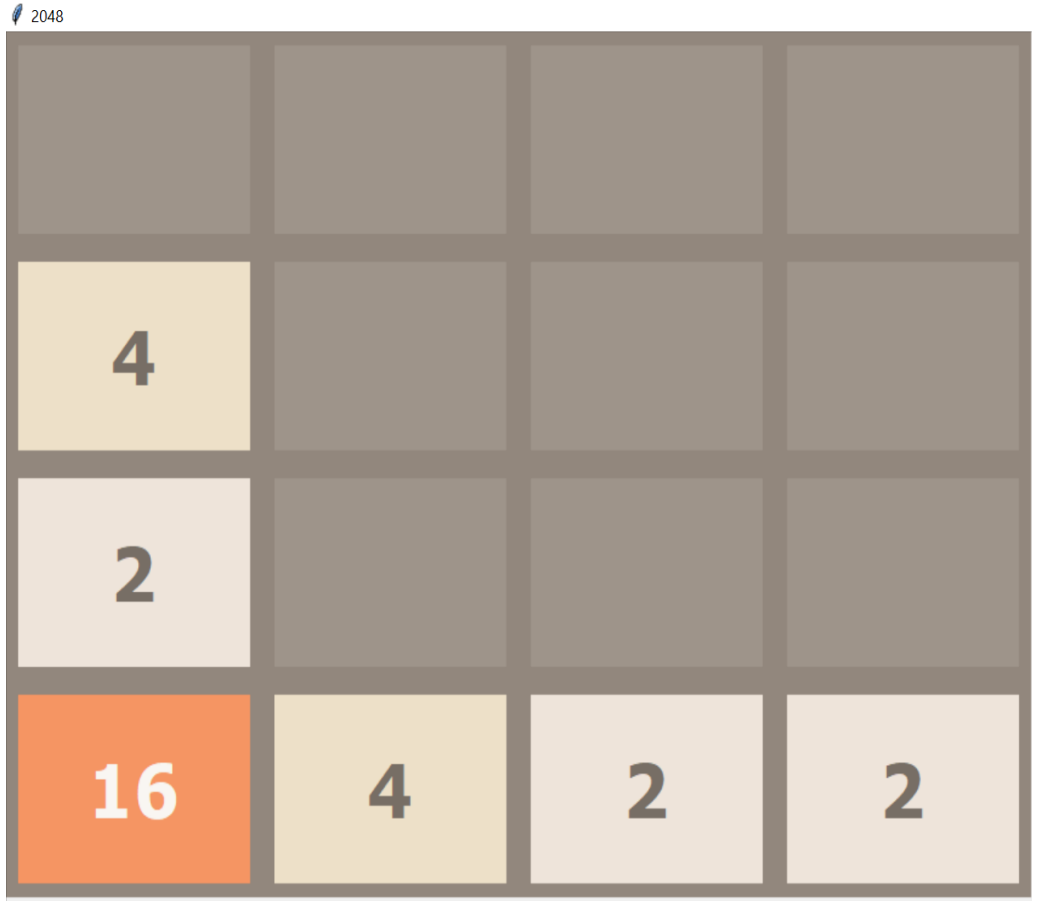
- I used exit() for quitting the game.
- Initialized a list called "history_matrixs" to load the matrix after every iteration.
- If key "b" is pressed it pops the matrix from "history_matrixs" making the matrix to move a step back.
- If the game state is "win" it prints "YOU", "WIN!" in matrix[1][1] and matrix[1][2] positions respectively.
- If the game state is "lose" it prints "YOU", "LOSE!" in matrix[1][1] and matrix[1][2] positions respectively.

Constants.py

- ***We initialize the:***
- *Size*
- *Length of the grid*
- *Grid padding*
- *Color of the cell and grid*
- *Font of the numbers*
- *Also specifying about the “KEY_QUIT”, “KEY_BACK”, “KEY_UP”, “KEY_DOWN”, “KEY_LEFT”, “KEY_RIGHT”.*

RESULTS

According to code :

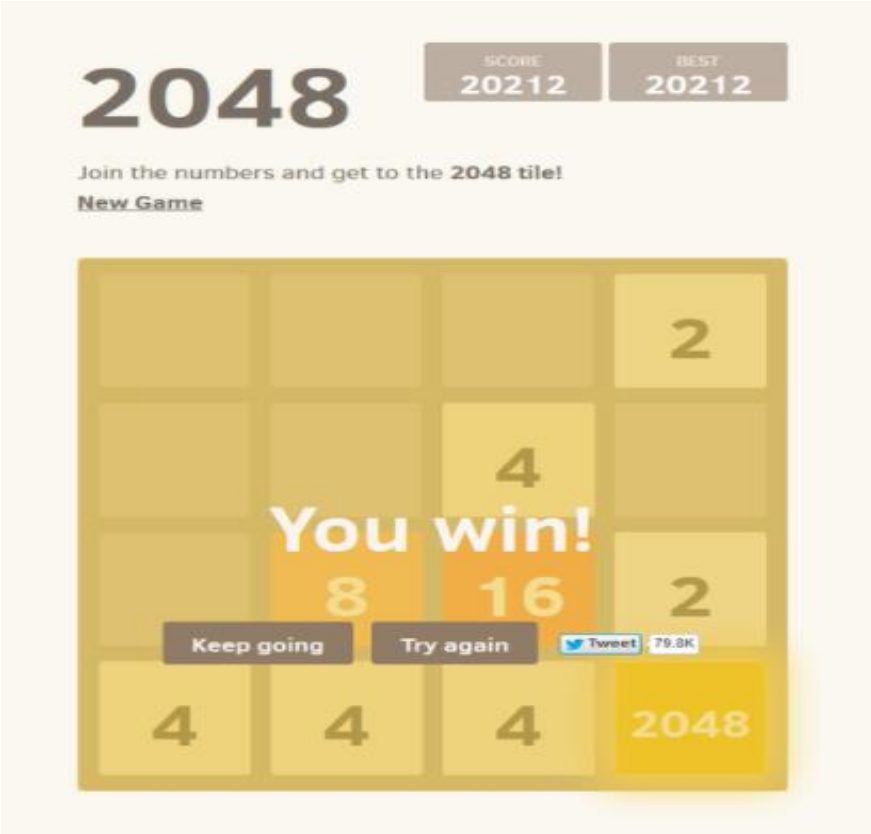


RESULTS

When total grid is filled such that no numbers match each other. So, that game Is over.

RESULTS

As the game name is 2048 ,on joining the numbers and getting tile 2048,then we will win the game.



FUTURE WORKS :

- *In our version of the game our code only adds on a 2 in an empty cell for every move. In our future works we are planning in such a way that every turn a new tile will randomly appear in an empty spot on the board with a 90% chance of being a 2 and 10% chance of being a 4.*
- *We are planning to develop an AI for 2048 in an efficient way to solve the puzzle on its own through expectimax algorithm.*

THANK YOU