Subject**: MATLAB and its Applications in Engineering (MAE)**
Course: **2022/2023**
Student: **Stanciu Iulia-Cristina**
Project: **Final Work – 2048 Game**

Final Work Project

MATLAB Applications for Engineering

2048 GAME

Name: Stanciu Iulia-Cristina

Delivery date: 29.12.2022

MATLAB release used: R2020a

Escola Tehnica Superior d'engineria de Telecomunicacio de Barcelona
Universitat Politècnica de Catalunya

Subject**: MATLAB and its Applications in Engineering (MAE)**
Course: **2022/2023**
Student: **Stanciu Iulia-Cristina**
Project: **Final Work – 2048 Game**

Table of Contents
Final Work Project MATLAB Applications for Engineering

Subject: **MATLAB and its Applications in Engineering (MAE)**
Course: **2022/2023**
Student: **Stanciu Iulia-Cristina**
Project: **Final Work – 2048 Game**

1. Introduction

1.1. Aim of project:

The goal of this project is to use the knowledge from the MAE Course Units 1 to 4 to create a MATLAB application capable of simulating various versions of a 2048 game.

The aim is implementing a complete and functional GUI for this well-known game in which the player to be able to see the board game, move the tiles in one of the four directions.

For the purpose of this project a fully functioning, color-coded board will be implemented, as well as some new features for a better user experience of the game, such as helping buttons (examples: reset (start over), undo).

1.2. Motivation:

2048 was my favorite offline game to play while travelling.

Even though it starts from a very simple idea, it can be played with strategy. In this project I would like to add some features to the original version of the game to make it easier for youngsters.

I would also like to track data and, in the future, see the most efficient strategies and the usual playing trend. I believe that this project is the best start for a reinforcement learning intelligent agent for the 2048 game and I would like to continue working on a better implementation outside of this course. **Theoretical background**

1.3. Description and gameplay:

2048 is a single-player sliding tile puzzle video game written by Italian web developer Gabriele Cirulli. The objective of the game is to slide numbered tiles on a grid to combine them to create a tile with the number 2048; however, one can continue to play the game after reaching the goal. The original game was created on a 4x4 grid, but it can be played on smaller or larger grids. 2048 is played on a plain grid (originally 4x4), with numbered tiles that slide in the purposed direction when a player moves them using the four arrow keys (up, down, left and right). Every turn, a new tile randomly appears in an empty spot on the board with a value of either 2 or 4. The new number appears with a specific probability (of 90% for a 2 and 10% for a 4). Tiles slide as far as possible in the chosen direction until they are stopped by either another tile or the edge of the grid.

If two tiles of the same number collide while moving, they will merge into a tile with the total value of the two tiles that collided. The resulting tile cannot merge with another tile again in the same move.

A scoreboard on the upper-right keeps track of the user's score. The user's score starts at zero, and is increased whenever two tiles combine, by the value of the new tile.

Figure 1. Original 4*4 blocks 2048 Game Board

Figure 2. Original 4*4 Game Board with the winning message when getting a block of value 2048

In some implementations of the game a different scoreboard representing the number of moves played is implemented. In this project both the score board and the moves board are implemented.

For the basic 4x4 game, the winning state is reached when a tile with a value of 2048 appears on the board. Players can continue beyond that to reach higher scores. The game ends when the player has no legal moves (there are no empty spaces and no adjacent tiles with the same value).

2. Game Development

2.1. Basic functionalities of a 2048 game that were implemented

- Pop-up window in which the user specifies the size of the board.
  - → The default size for the board game is 4x4, so the number 4 is written as a default option.
- Game board updated at every move:
  - → The default size for the board game is 4x4.
  - → The board is resizable: different sizes (2x2, 3x3, 5x5, 6x6, and even larger) can be played.
- Board movement
  - → The movement of the blocks is generated by the use of keyboard arrows. The four options are Up, Down, Left and Right and they generate the movement of the tiles in the specified direction.

- Random generator for the new tile
  - → The random generator has a defined probability for the appearance of a 2 or a 4. The default probability is set on 90% for a 2 and 10% for a 4, but it can be changed.
  - → The random generator is used at every move and in the beginning of the game to generate two random tiles.
- Scoreboard
  - → The score starts from 0 and is upgraded at every move. The score update comes from the tiles with the same number colliding with each other.
- Movesboard
  - → The number of moves starts from 0 and is updated at every move.
- Buttons for Restart (new game)
  - → The button generates a pop-up window in which the user can choose whether to continue the current game or to start another.
- Interface for Winning Moment
  - → When the stop point is reached (2048 tile appears in the 4x4 game) a pop-up window with buttons appears. In that moment the game options are: Keep Going (continue playing) and Try Again (restart game).
- Interface for Game End moment
  - → When there aren't possible moves anymore, a pop-up window is generated to tell the player that he lost. In that moment the options are: Start new game and Quit (stop playing)

2.2. New functionalities implemented

- Possibility to change game board size (to 3x3/5x5 or more) with updated goal to be reached (2x2 board => 32 goal, 3x3 board => 256 goal, 5x5 board => 16384 goal and so on). The board dimension can be chosen from the first pop-up window.
- "About" information about game and gameplay button.
- Restart game button for a new game.
- Button for Undo (with limited number of uses per game, with the possibility of going back only to the previous move – only once).
- "Add new tile" move - actioned by a push button.
- Color change mode (light/dark) – actioned by a toggle tool button.

2.3. Functionalities purposed for later development of the (not yet implemented)

- Buttons to generate the four possible moves integrated in the user interface as an alternative for the keyboard.
- Button for Redo (with limited number of uses per game).
- "Recommend a move" button (with limited number of uses per game) .
- A reinforcement learning algorithm for an AI agent.

2.4. Other aspects taken into accountin the project:

- Visual appeal
  - → Choosing visually pleasing colors for the different numbered tiles: The light mode uses the original colors of the 2048 game, while the dark mode uses brighter, more vibrant colors.
  - → A visually pleasing board with round corners.
  - → Well-thought color scheme for the score and moves boards.
- Comfortable experience – button size, button images, form and placement.

2.5. Game Prototype:

The game starts with a pop-up window welcoming the user. From there the player can choose its preference for the board size. The window had two butoons: Ok fotr fontinuing the game and cancel for closing the window and not playing.

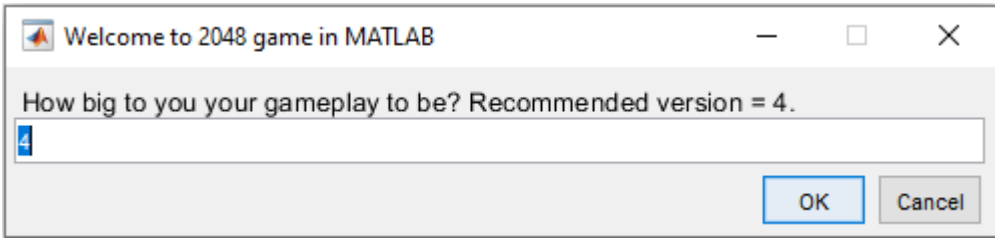

Figure 3. Starting window of the 2048 Game Board

When the "OK" button is pressed, the NxN board appears. If N is not provided by the user, the board will be set to the default size of 4x4. Along with the board, the score 0 is displayed as well as the number of moves, set to 0 in the beginning. When the game has been opened, two blocks of number 2 or 4 are generated with the probability set beforehand (the default is 90% for a 2 and 10% for a 4)

Subject: **MATLAB and its Applications in Engineering (MAE)**
Course: **2022/2023**
Student: **Stanciu Iulia-Cristina**
Project: **Final Work – 2048 Game**

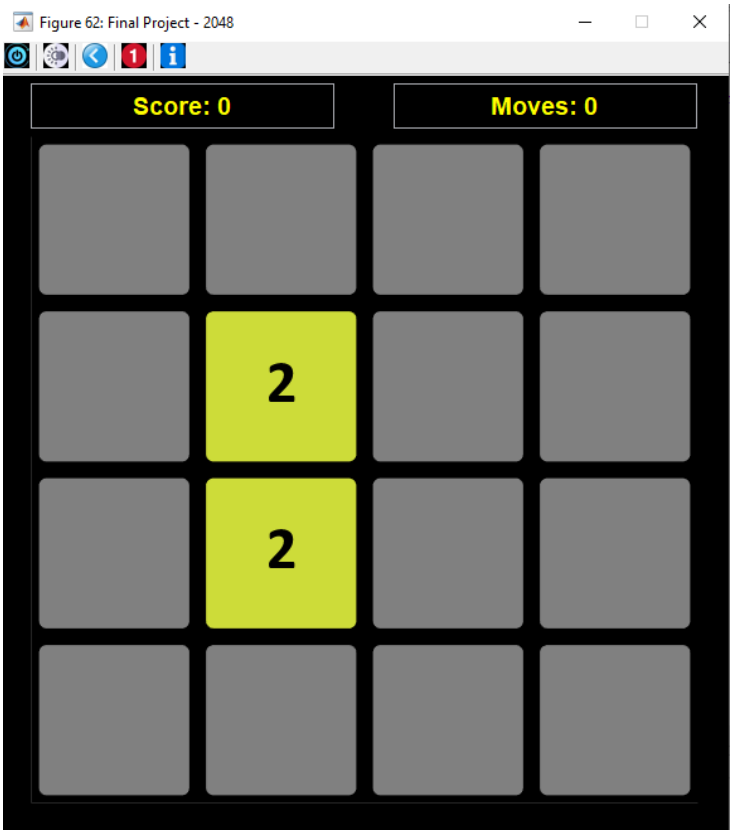

Figure 4. Starting point of the 2048 game, dark mode

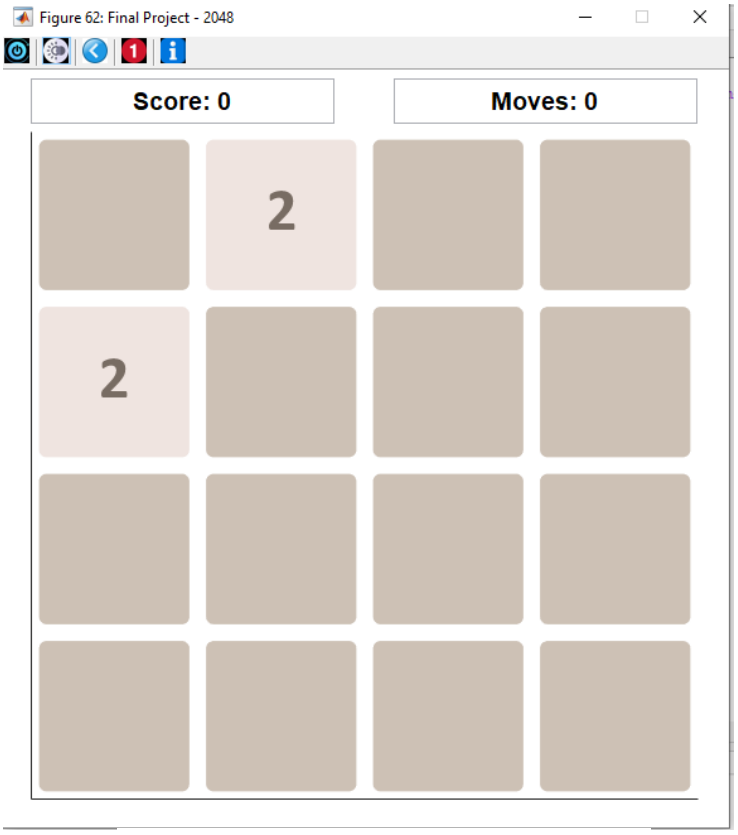

Figure 5. Starting point of the 2048 game, light mode

The board is created depending on the size of the screen on which it is played. The board comes with five buttons with the following functionalities (from left to right):

1. New Game = Restart
2. Dark/Light Mode = Switch between color modes
3. Undo Last Move
4. Add New Tile = Generate a new random tile to help the user in the game
5. About… = Info button

The moves are generated in the moment where one of the arrow keys are pressed, causing the numbered blocks to move in the chosen direction and combine to make a bigger number if they are the same. The only keys that can be used are the arrows.

Different interfaces are being implemented for different moments in the game:

- Restart window to make sure that this is what the user wants,
- Winning Message with "Continue" or "Start new game" buttons.
- Losing Message with 'Try Again" and "Quit" options

3. Implementation and Results

Functions involved in the actual game play

The GamePlay class contains the board and most of the functionalities mentioned in the previous pages. Some are implemented and others will be implemented in the future. Some parts of the code will be provided below and the functions will be explained.

The class contains the board as the board object, the size as in N (the number of blocks per row/column), the number of moves initialized to 0, two booleans for the game status, the score initialized as 0 as well and the „StopNumber" variable as the value of the winning point. The winning point depends on the size of the board (which can be provided by the user of 4 as default). Another boolean variable monitories if the move wanted by the user changes the board configuration or not. If the configuration is changed, the number of moves and score changes, otherwise it does not.

Two events are declared in this class: the winning event as "Win" and losing event as "GameOver".

In the GamePlay (initializing) method, the board size is provided by the user. The winning point is set accordingly. The board is initialized with 0 values for all the blocks except for two which are randomly assigned with values of 2 or 4.

```
function obj = insertRandomBlock(obj)

    % finding the empty places on the board where the random number
    % can appear
    empty_tiles = find(~obj.Board);
    % getting a random index from the array of empty places on theboard
    index = randi(prod(size(empty_tiles)));

    % setting the probability for the randomly generated number on
    % the board:
    % default 10% for 4 and 90% for 2
    probability = [0.9, 0.1];
    p = 1:2;
    sample_size = 1;

    % getting the power of 2 for the strating tile
    power = randsample(p, sample_size, true, probability);

    obj.Board(empty_tiles(index)) = 2^power;
end
```

Figure 6. Screenshot of the MATLAB Code for generating the random tiles with values of 2 or 4

As seen in Figure 6., the probability is set to the default value of 90% for a 2 to appear as the new block. The function find is used to discover the empty block and the function randi is used to generate a random index for the newly created tile.

```
%get highest value in the game board (used to check win)
function val = getHighestBlock(obj)
    val = max(max(obj.Board));
end
```

Figure 7. Screenshot of the MATLAB Code for the GamePlay class – getHighestBlock method to get the highest value on the board

If the highest value on the board is equal or bigger than the winning point set in the beginning of the game, the "Win" event is activated (set to true).If all the blocks have non-zero values and there is no move that can be made, the event "GameOver" is activated.

Three functions are defined to serve the game mechanism. Depending on the key pressed by the user, the tiles are moving to the right, left, up or down, as explained in the game description. In order for the number of moves and score to be implemented, at every move the methods check if there was any change in the board configuration. If not, the moves and score stay the same and the user needs to make a different move to advance in the game. A random block is generated only if there was a change in the previous move. After every move, the game checks isf the user has won or if the game ended in a lose. This is made with the help of the previously mentioned functions.

The movement is processed row by row or column by column depending on the direction. For each row or column the same algorithm is used, finding the non-zero values and performing the addition of tiles to the left. Therefore, the output is a vector with the non-zero values to the left and 0 values (if they exist) to the right. The score is being updated at every move, adding twice the value of the blocks that are about to collide.

A function for displaying the score, number of moves and board as a matrix is also defined. This particular one was used a lot for debugging purposes.

The GameBlock class is used for the user interface (drawing the tiles and board in their specific colors). This class works with the position of rectangular tile and its text. Two color schemes (with both background and foreground colors have been implemented, one with the original colors of the game and a dark mode. Different text sizes have also been defined and assigned to tiles depending on how large the corresponding number is.

The FinalProject_2048Game class uses both Gameplay objects and GameBlock objects to create a game and a user interface for the game. Knowing the screen size of the computer, the board is configured to fit and be easily usable. Coordinates for the blocks are defined (depending on the number of blocks) and two gameplay objects are created, on for the current board and one for the last move.

The user interface includes a figure with the board which is updated at every move, a toolbar and two control units/boxes for score and number of moves.

The KeyPress function generates the movement of the board. Using update_blocks and update_scores functions, the figure is modified to the current state. In Figure 8 the differences after a left movement can be seen.

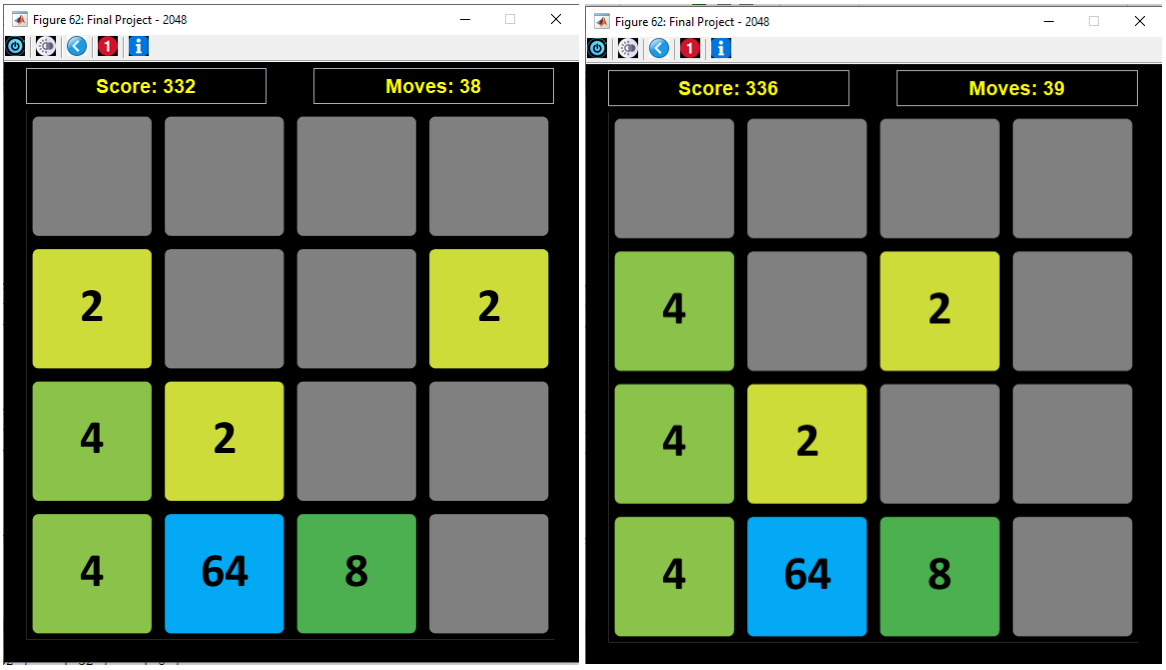

Figure 8. Difference in the board after one move to the left.

The toolbar buttons are the following:

- The reset pushbutton that opens a pop-up window to make sure of the user intentions. (Figures 9 and 10)

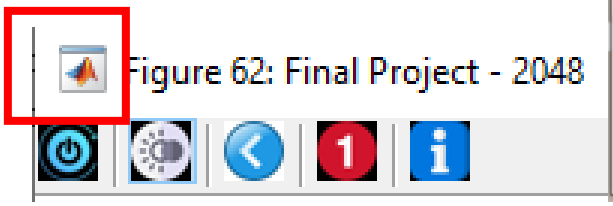

Figure 9. reset Button in toolbar

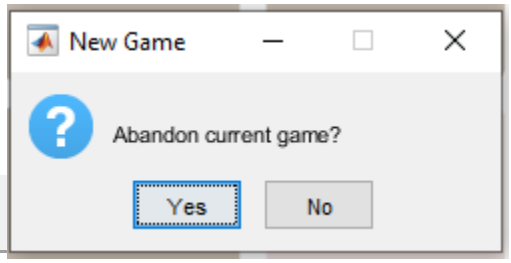

Figure 10. New Game pop-up window

- The light/dark mode button. (Figure 11)

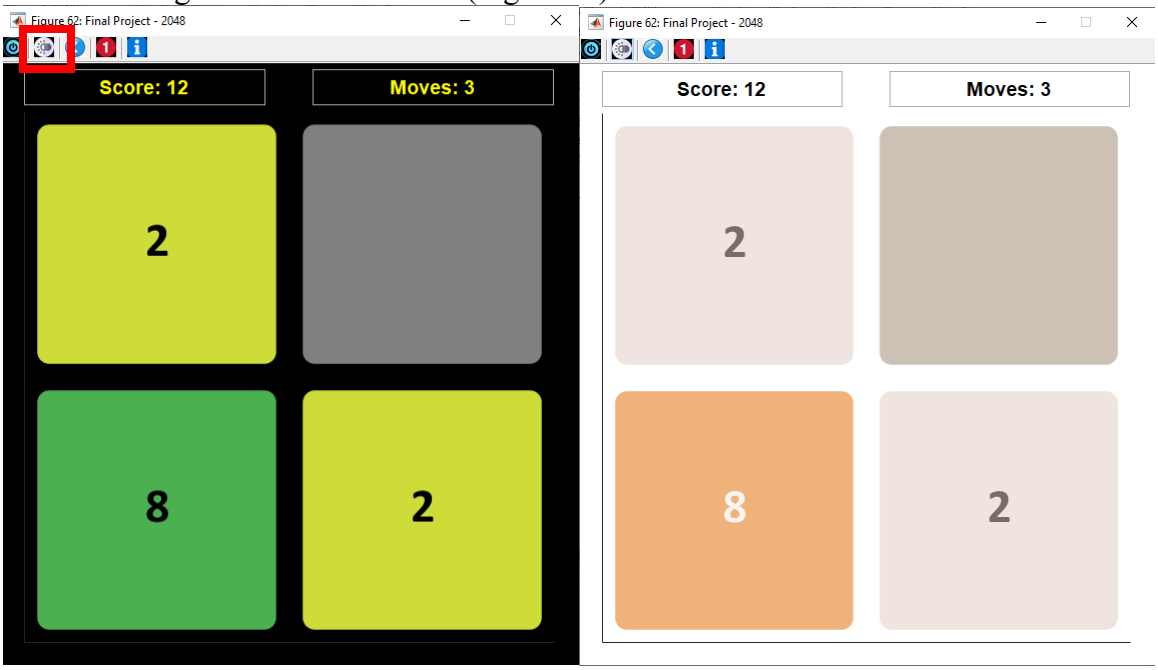

Figure 11. Dark/Light mode button effect

- An undo button (Figure 12)

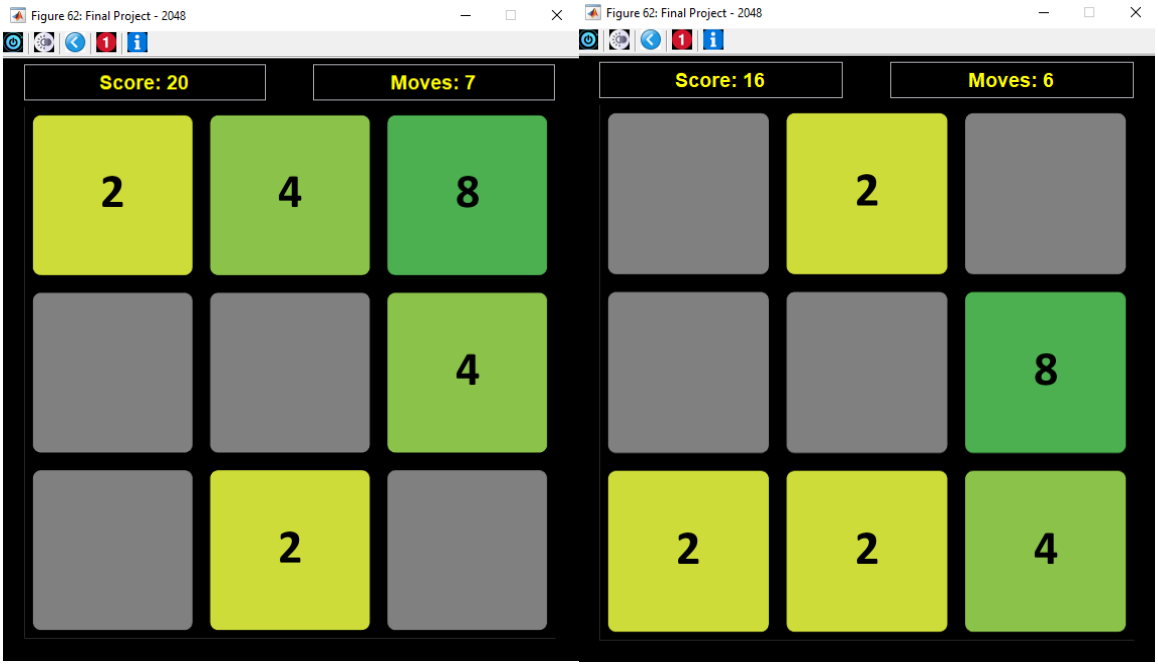

Figure 12. Undo button effect

- A button to add a new tile (Figure 13)

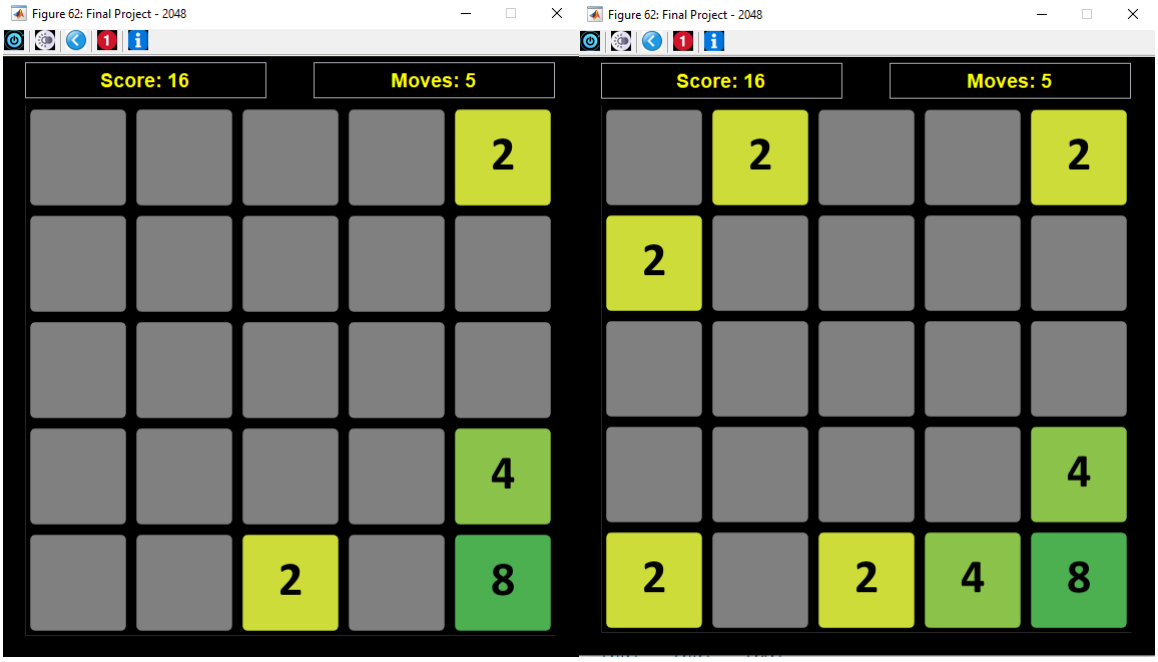

Figure 13. New tile button effect(4 times)

- An information button. (Figure 14)

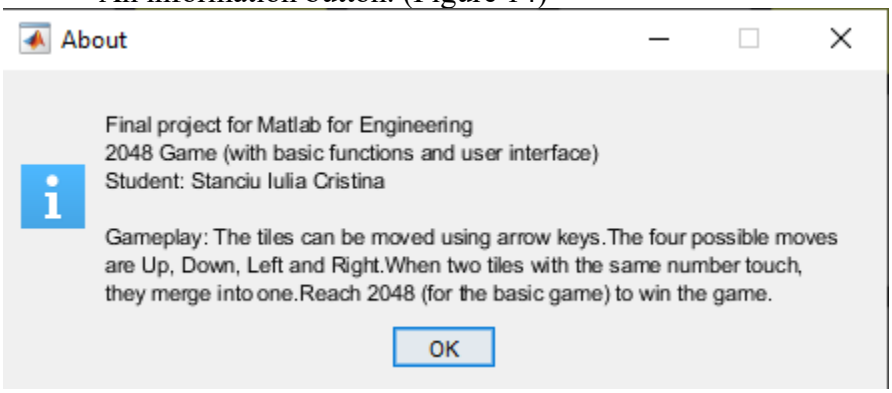

Figure 14. Information window

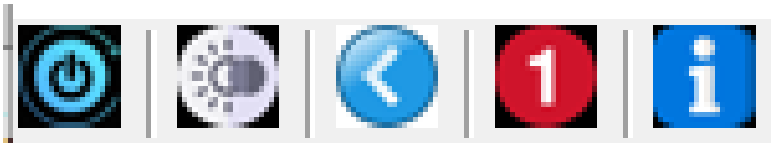

Figure 15. Toolbar full view

Interfaces for the winning and losing moment have also been defined (please take notice that for the losing moment the board is not updated with the last move). The losing moment can be seen in Figure 16.

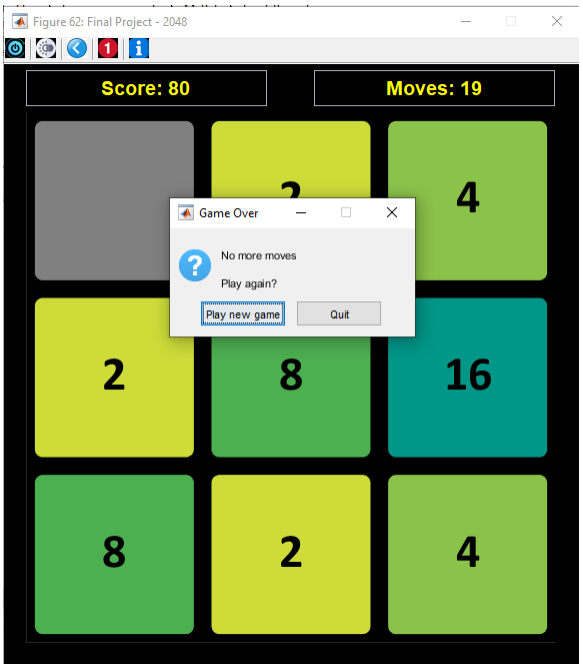

Figure 16. Losing moment in the 3x3 2048 game play

4. Conclusion

The basic game functions were implemented in the last weeks. The development of this game has been a complex and interesting task which challenged my MATLAB programming knowledge and expanded my knowledge regarding the creation and use of a user interface.

Some difficult aspects were defining the axes and blocks for the game, making sure that the number of moves is not modified every time, just when the board is changing.

My favorite parts of this project were adding new functions and buttons to help the gameplay. A correct use of the keys was implemented during multiple days and the color codes were well thought for the user experience. The user interface makes the game easily playable.

Finally, I believe I have implemented a correct version of the game 2048 with some interesting functions and small additions. However, I would like to continue this project and make a recommendation system for the moves analyzing the board and the preferred moves until then, a graphical representation of the moves-score development and an AI/ game analyzer will be implemented to analyze the game moves and score trend.

5. References

- https://www.mathworks.com/
- https://2048game.com/
- https://en.wikipedia.org/wiki/2048_(video_game)
- https://blogs.mathworks.com/pick/2014/04/04/submit-your-algorithms-to-solve-2048/?s_eid=PSM_7378&fbclid=IwAR1ahuGVJ9CgaBo28vvETjnrP2jbk0UlVT8v08tXlDHvzthWjZsnzrn1ZpQ