

Ultimate 2048

Project Report

Team 4  
Slot-11 Group-18

Prepared for  
CS 101 course project  
Instructor : Prof. D.B. Phatak,  
Prof. Supratik Chakraborty  
Autumn 2014

## INTRODUCTION

In this project, we have attempted to make a game of 2048.

Our team includes :

1. Mohit Vyas    140050015
2. Vishal Meena    140050010
3. Akash Bhaneria    140020075

## Game Overview:

*2048* is a single-player puzzle game created in March 2014 by 19-year-old Italian web developer Gabriele Cirulli, in which the objective is to slide numbered tiles on a grid to combine them and create a tile with the number 2048. It can be regarded as a type of sliding block puzzle.

*2048* is played on a simple gray 4×4 grid, with numbered tiles that slide smoothly when a player moves them using the four arrow keys. Every turn, a new tile will randomly appear in an empty spot on the board with a value of either 2 or 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. Higher-scoring tiles emit a soft glow.

A scoreboard on the upper-right keeps track of the user's score. The user's score starts at zero, and is incremented whenever two tiles combine, by the value of the new tile. As with many arcade games, the user's best score is shown alongside the current score.

The game is won when a tile with a value of 2048 appears on the board, hence the name of the game. After reaching the target score, players can continue in a sandbox mode that continues beyond 2048. The maximum possible tile is 131,072 (or  $2^{17}$ ); the maximum possible score is 3,932,100; the maximum number of moves is 131,038. When the player has no legal moves (there are no empty spaces and no adjacent tiles with the same value), the game ends.

## Interface:

This section explains how to use GUI of game & start playing. Once you run the executable file(.exe) two windows will open, one of them is the console output and other is the home screen of the game.

### *1.Homescreen :*

It consists of three options:

- i).Play :- Selecting this will take to the main game screen where you can play the game.
- ii).Game Options :- Selecting it will lead you to options screen, where you can choose the size of grid, colour theme and the game mode.
- iii).Read instructions : Selecting it will take you to instructions screen, from where you can review the rules & objective of the game.

### *2.Game Options Screen :*

Here you can choose:

- i).Size of Grid :- The Grid size can be set to 4x4, 5x5, 6x6 or 7x7 (default being 4x4)
- ii).Colour Theme :- Colour Theme can be set either Red, Green or Blue.
- iii).Game mode :- Here you can choose from the following game modes:
  - a). X-tile : This mode has one X-tile which cannot be added to any other tile.
  - b).Survival : In this mode 2 or 4 numbered tiles keep popping out randomly after a fixed interval of time. If you fail to add them and the screen gets filled completely then the game is over.
  - c).Classic : This mode is the normal **2048** game without any modifications.

### *3.Instructions Screen :*

It displays the game rules for your reference.

### *4.Game Screen :*

It consists of:

i).Grid :- It is the space where all the tiles are situated and are moved.

ii).Tiles :- Tiles are coloured square with numbers on them.

iii).Scoreboard : It shows the current score and high score.

## **FUNCTIONAL SPECIFICATIONS**

The basic functional specifications require that inputs shall be taken from the mouse and keyboard, as and how the user responds to situations in the game which is being played. The user shall then see output (or the results of his actions) on screen and in a reasonably standard format. More details are as follows :

The following are the important functions which have been defined in the program :-

1. int getMovesLeft(int Gamearray[7][7], int grid ); :-

This function takes an array of size 7 and an integer, and returns the number of valid moves available to be performed on the array of size grid .

2. int emptyCells(int Gamearray[7][7], int grid ); :-

This function takes an array of size 7 and an integer, and returns the number of Empty cells in array of size grid .

3. bool isGameOver(int Gamearray[7][7], int grid); :-

This function takes an array of size 7 and an integer, and returns a Boolean value, which is true if game is over and false if not.

It uses the functions emptyCells and getMovesLeft.

4. void reset(int &score,int &highScore,bool &showGameOver,int Gamearray[7][7], int grid,char gamemode,int Count); :-

This function re-initialises the gamearray after the game is restarted, and logs the highscore , if previous score is the highest till now.

5. void renderGscreen(int Gamearray[7][7], int grid,sf::RenderWindow &window,bool showGameOver); :-

This function sets style, colour, position of text and rect objects.

Draws tiles(rect) and tile text, score,highscore and gameover score (text).

Uses sf::Font, sf::Vector2f, sf::RectangleShape, sf::Text and various other sfml classes.

6. void survival(int Gamearray[7][7]); :-

This function generates 2's,& 4's after a fixed time interval at random position.Uses sf::Clock, sf::Time classes of sfml.

This functions is launched using another thread along with main thread; Unfortunately we have not been to get it to work properly.

## **DESCRIPTION OF DATA (INPUT/ OUTPUT)**

In general, Input and output are standard and as expected, with slight modifications from what might be seen by the user. These have been specified as follows:

Input: The inputs comprise of the user telling the computer the directions in which tiles have to be moved. The method to do this has been elaborated in the User Interface Requirements.

Output: The output of course is the state of the game at any point in time as well as specific messages conveyed by the system to the player(s). These also have been elaborated on in the section User Interface Requirements.

## **Future Ideas :**

Following are the features which we wish to add if the time permits:

1. Virahanka 2048 :- In this game the tiles containing two consecutive Virahanka numbers will merge to produce a tile containing the next Virahanka number.
2. Undo :- A button which revert the game to a state before the last move.(a finite number of times)
3. Hexic 2048 :- In this, the grid will not be a square but instead will be a honeycomb grid containing hexagonal tiles.

## **Acknowledgements :**

**Simple and Fast Multimedia Library (SFML)** a open source portable API for multimedia programming developed by Lorent Gomila.