Practical C#14

2048

Assignment

Tarball

You shall submit a tarball of zip format (.zip extension), with respect to the following architecture:

```
rendu-tp-firstname.lastname.zip
    |-- rendu-tp-firstname.lastname/
        |-- AUTHORS
        |-- README
        |-- 2048
            |-- 2048.sln
            |-- 2048
                |-- Program.cs
                |-- Grid.cs
                |-- Score.cs
                |-- Controller.cs
                |-- Number.cs
                |-- Tout le reste sauf bin/ et obj/
```

- Obviously, *firstname.lastname* are to be replace by your firstname and lastname in order to form your login.
- You shall submit **code that compiles** and be readable.
- You can be required to modify the architecture of the assignment in order to include some bonuses, suggested or of your own choosing.

AUTHORS

This file shall contain one line with respect to the following format : a star (*), one space, your login and linefeed.
Here is an exemple with the character $ representing the linefeed :

```
*␣firstname.lastname$
```

Beware that the name of the file is **AUTHORS** without any extension. To create such a file you can use the following command and replace by your login :

```
echo "* firstname.lastname" > AUTHORS
```

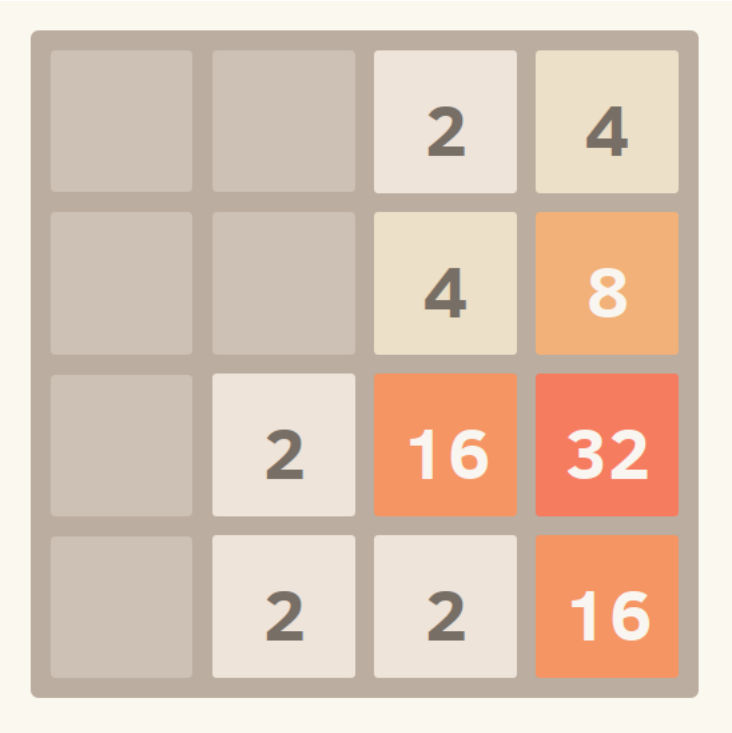
README

Write in this files any comment you find relevent concerning this practical or more generally your strengths and weaknesses. Don't forget to mention any bonus that you choose to implement.
An empty README will be considered as an archive failure.

1 Introduction

No doubt you are all aware of the famous game called 2048 (`https://gabrielecirulli.github.io/2048/`). This week's practical's aim is to implement one using C#. This should be a very good exercise for you to review the notions you've seen so far as this subject already appeared on an exam subject. You will see that this practical allows you more freedom concerning the implementation, and focuses more on bonuses. We advise you to try and do it as if you were in an exam, that is with as little help as you can manage from the outside.
Again, you shall respect all the assigment given on the first page and the grading will take into account your use of the C# language.

The game is played on a 4x4 grid with goal to merge tiles of same value in order to obtain the biggest value possible. The game is over when the value reaches 2048 (win) or when there is no way to merge tiles (loss).



2 2048

2.1 Program.cs

This file will contain your main function, instanciate all the classes that need it.

2.2 Controller.cs

The player will have to interact with the game, and so you will have to handle the keybord events. Since you have already had the occasion to do so several times, it shouldn't be much of a challenge. One way consists in creating a **`Controller`** class which will have a method to handle keyboard events, and from them, you return the appropriate action. By doing so, you will increase the modularity and readability of your code.
You may also use an **`enum`** in order to handle the possible actions and then choose what to do with your grid.

```
1 public class Controller
2 {
3         public enum Action{Up, Down, Left, Right, Quit};//add more if needed
4         public Action GetAction();
5 }
```

2.3 Grid.cs

This class should be the biggest one of your project, indeed, it will be the "engine" of your game. You must be able to handle all of the following parts :

- Move tiles in the 4 directions
- Update the **`Score`** class.
- Check the state of the game (win, loss). (Bonus)
- Handle the printing of your grid.

When you move tiles, the tiles that are on the edge of the same direction you are moving will not move (but can still be merged). Each time you move your tiles, you create a new one with a value less than or equal to 4.

2.4 Score.cs

This class shall be **`static`** (no instance). It should contain the score of the current game.

```
1 public static class Score
2 {
3         /* FIXME */
4 }
```

Each time you merge two tiles together, you add both of their values to the current score. (If you are merging two tiles of values 2, you will add 4 to the score).

2.5 Number.cs

This class is not mandatory, it can however be very useful in order to add some features to your game. You can for example use it to store the color and the value of a tile. (Remember that the color is a bonus but we would like it very much if you add it to your 2048).

2.6 Bonus

You are free to implement whatever you'd like as a bonus. You are encouraged to do as many as possible (but remember that the basics have to work fine before to do any kind of bonus).
Here are a few examples that you can add to your game :

- Print a message in case of victory (You reach the value 2048)
- Print a message in case of defeat.

- Save you score in a file (to create a Leaderboard)
- Save the current state of the grid in a file.
- Handle colorful printing.

Don't forget to add the bonus that you implement in your README!

The code is the law.