ynov INFORMATIQUE

B2 - Ytrack C#

Evaluation Project - Tetris

By Julien SOSTHENE - 2021

Adapted from Tetris Project by Jordan GRILLY @ Lyon Ynov Campus

Outline

1. Project Presentation
2. Evaluation criteria
3. Step 1 : Gather the notions
4. Step 2 : Design
5. Step 3 : Production

1. Project Presentation

1.1 Gist : Tetris in C#, in the browser

The goal of this project is to implement the Tetris game in C# using the Blazor framework.

Blazor is a front-end web framework in C# using the .Net Framework: your interface will be in HTML and CSS, but your code will be written entirely in C#, using WebAssembly to run it in the browser.

1.2 App pages

`/` Menu

The menu page will contain

- The title of the game
- A `Play` button that takes you to the `game` page
- An `Options` button that takes you to the `/options` page
- A `Credits` button that takes you to the `Credits` page

`/options` Options

The options page will contain inputs to set up

- The **speed** of the game (as `slow`, `normal`, and `fast`)
- The the **width** and **height** of the game grid in *blocks* with the default being 10 by 20
- Any additional settings that you wish to include

`/game`

The game page will contain

- the grid : each cell of the grid will being a separate HTML element (Or blazor component).
- A score box
- 5 buttons:
  - One to move a piece to the left
  - One to move a piece to the right
  - One to turn a piece clockwise
  - One to make it fall faster
  - One to pause the game

- The game starts immediately when moving to the `/game` page or can start on the push of a button
- A random piece is spawned at the top of the grid at the beginning and each time the previous piece stacks onto the pile.
- When one of the lines is complete, make it disappear from the grid
- When one of the columns is as high as the grid, a "Game Over" message must be displayed, along with a "Play again" button

`/credits` Credits

The credits page must contain :

- A link to the original Tetris game
- A mention that this is a student project
- The technologies that you used and links to their documentations (at least Blazor, C# and .Net)
- Your names!

1.3 Project steps

- First, you'll have to learn the tools of the trade : follow a Blazor tutorial and test it on simple cases, and ensure that the whole team is comfortable with the concepts
- Then, you'll have to produce a design document, described hereafter
- Then, you'll start coding.

2. Evaluation Criteria

2.1 Groups

The project will be held in groups of 3 with a tolerance for 1 group of 4.

2.2 Deliverables

The evaluation will be focused on the following criteria :

- Relevance, accuracy and presentation of the design document (UML diagrams)
- Object-Oriented Programming
- Implementation of features
- Respect of best practices in your code (DRY, KISS, etc.)
- Balance of the commits between the members of the group

2.3 Project Defense 👩‍🎓👨‍🎓

You will be presenting the project on the **last** session of the module for **30 minutes** among which :

- 20 minutes will be reserved for your formal presentation
- 10 minutes will be reserved for Q&A.

💡 The usual presentation structure applies.

3. Step 1 : Gather the notions

3.1 Learn how to make a *Class Diagram*

Learn how to make a Class Diagram

You can use tools such as Draw.io to make actual class diagrams

3.2 Get some insight on the following notions

If you don't know how to use them, first read the following documentation

- C# Documentation : Class
- C# Documentation : Inheritance
- C# Documentation : Timers

3.3 Follow the Blazor tutorial

Follow the Blazor tutorial and make sure you understand the concepts

4. Step 2 : Design

4.1 Don't dive in head first!

Before you start coding, take time to design your application's architecture:

- List the entities of your application (core game and display components)
- Make quick Wireframes of your pages
- Formalize your design using a Class Diagram
- List the Razor Components you will need alongside their parameters or lack thereof

4.2 Gather your design in a document

Make a design document containing :

- A quick description of the project
- Your wireframes
- Your UML Class Diagram
- Any technical information that is relevant to your planning
- A list of issues that you still have to solve

4.3 List and distribute tasks

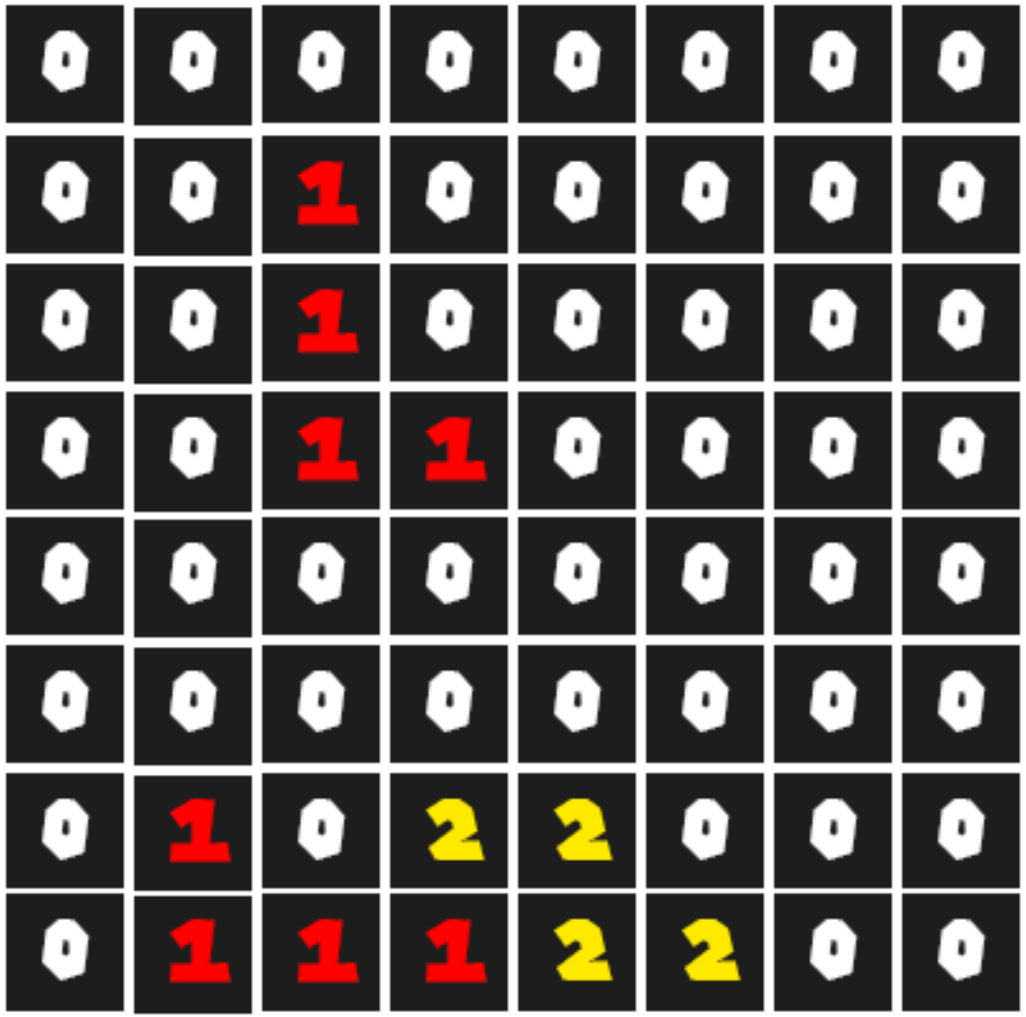
List and distribute the tasks among the group.

- 💡 Ideally, concurrent tasks should be as independant from each other as possible

4.4 💡 Tips for your design

- Separate your Data and Logic (regular C# classes) from your display mechanisms (Razor components)
- Think KISS: each class should have a limited set of responsibilities

- Your grid data can be represented as a 2-dimensional array of `int` s (or Enum variants)



5. Step 3 : Production

 Start coding!