SCALE FOR PROJECT RED-TETRIS (/PROJECTS/42CURSUS-RED-TETRIS)

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere-v2.1337.ma:vogsphere/intra-uuid-c07186fc-499f
```

Introduction

For the smooth running of this evaluation, please respect the following rules:

- Remain polite, kind, respectful and constructive whatever happens during this conversation. It's a matter of confidence between you and the 42 community.

- Highlight the potential problems you 've had with the work you're presented to the person or the group you're grading, and take the time to talk about and discuss those issues.

- Accept the fact that the exam subject or required functions might lead to different interpretations. Listen to your discussion partner's perspective with an open mind (are they right or wrong ?) and grade them as fairly as possible.
42's teaching methods can make sense only if peer-evaluation is taken seriously.

Guidelines

- You must only evaluate what you will find in the student's or group's GiT repository.

- Take the time to check that the GiT repository matches the student or group and the project.

- Double check that no malicious alias was used to mislead you and make you

grade something different from the official repository content.

- If a script supposed to help evaluate the exam is supplied by either side, the other side will have to strictly check it to avoid nasty surprises.

- If the evaluating student has not yet taken this project, they will have to read the exam subject in its entirety before starting the evaluation.

- Use the flags available on this grading system to signal an empty or non-funcional project, a norm flaw, cheating, etc. In that case, evaluation stops and final grade is 0 (or -42 if it's a cheating problem). However, if it's not a cheating problem, you are invited to keep talking about the work that has been done (or not done, as a matter of fact) in order to identify the issues that lead to this stalemate and avoid it next time.

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/29965/en.subject.pdf)

Game functionalities

Running the game in solo mode

A player can join a new game and launch it. The access URL to a game is faithful to the doc (hash-based). ex.:
http://:/#[]

Yes No

Launching the game in multiplayer mode

Multiple players can join a new game. Only the first one can launch it. A player cannot join a game in progress. The game is faithful to the principles of Tetris. The game is over when on player is left.

Yes No

Relaunch a game

At the end of a game, only the top player of the game can relaunch it. If this player has left the game, a new player replaces them and can launch

a new game instead. After the end of a game and before relaunching it, new players can join the game.

☑ Yes ☒ No

Blocks dispatch

During a game, players receive the same sequence of blocks in the same position and coordinates.

☑ Yes ☒ No

Moving the blocks

Blocks rotate, move to the left, to the right, and fall as indicated in the documentation. You can move a block during a timer tick once it's landed, except if you forced the fall.

☑ Yes ☒ No

Line injection

When a player destroys lines, opponents receive a n - 1 line malus. They are indestructible and appear at the base of their structure.

☑ Yes ☒ No

Graphic interface

Respect of the HTML / DOM constraints

A flexbox or a grid layout is used instead of a

tag. No use of canvas or
SVG. No DOM handling library usage.

☑ Yes ☒ No

Spectres visualization

In multiplayer mode, it is possible to identify the opponents thanks to their names and observe the specter of their structure. Each modification of an opponent's structure shows in their spectre.

Yes No

Client implementation

Socket.io encapsulation

The use of a socket.io must be completely encapsulated in a middleware.

Yes No

Functional programming

Never call `this` except to define new 'error' sub-classes. All the logic relative to the load management of the blocks must be written as pure functions.

Yes No

Server implementation

Objet oriented programming

Games and players management server logic must be implemented as object programming.

Yes No

Unitary tests

Sufficient cover

The `npm run coverage` command must indicate that tests cover at least 70% of the statements, functions, lines and at least 50% of the branches.

☑ Yes ☒ No

Bonus

Bonus will be taken into account only if the mandatory part is PERFECT. PERFECT meaning it is completed, that its behavior cannot be faulted, even because of the slightest mistake, improper use, etc... Practically, it means that if the mandatory part is not validated, none of the bonus will be taken in consideration.

Bonuses

In this part, you will count the bonuses proposed by the subject or added by the project owner.

Each bonus must be:

- At least a little useful (at your discretion)
- Well implemented and 100% functional

Rate it from 0 (failed) through 5 (excellent)

Ratings

Don't forget to check the flag corresponding to the defense

✔ Ok ★ Outstanding project

Empty work Incomplete work No author file Invalid compilation

Norme Cheat Crash Incomplete group Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation