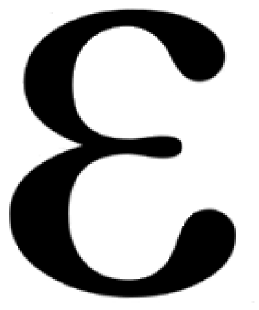
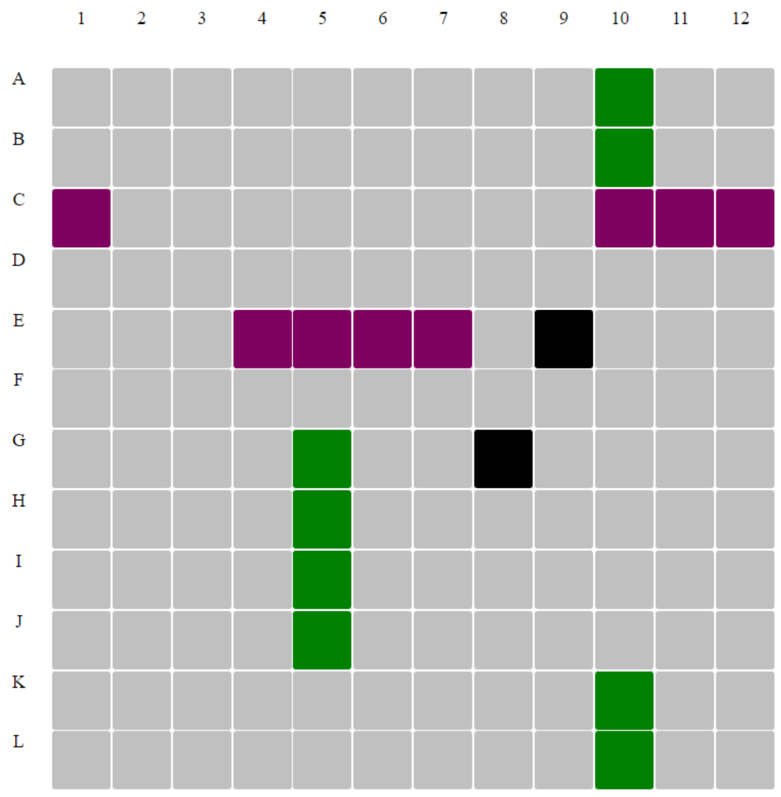
Epsilon Game

-by Mihai Bobeică-



Introduction

Epsilon is a browser game in which two players take turns placing rectangles of sizes 1×4 (purple player) and 4×1 (green player) on a 12×12 board. The pieces can also wrap around the board. The game is won when the opponent is unable to place anymore.



My motivation for Epsilon started in primary school when I enjoyed playing tic-tac-toe. I found myself studying strategies and shortly afterwards I knew what the optimal move was every time. From that point on, I didn't enjoy the game as much as I used to.

I wanted to create a game like tic-tac-toe, where there is no optimal move, but which has a clear winner.

What I took from tic-tac-toe was how the game must finish after a finite number of moves. This shaped the design of the game into 2 players placing pieces until the board is filled.

Designing the game

Choosing the pieces and the name

While I was working on generating functions problems for the 10th grade Mathematics Olympiad, I stumbled upon a problem which to this day I keep close to my heart.

Let there be a rectangle with positive integer sides which can be covered with rectangles of sizes 1×n and m×1 respectively (with n and m being positive integers). Prove that it can also be covered using only rectangles of sizes n×1 and 1×m.

The solution started with considering the following numbers:

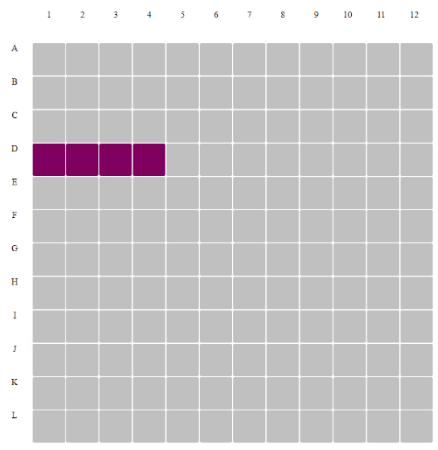
$$\varepsilon_m = \cos\frac{2\pi}{m} + i\sin\frac{2\pi}{m} \text{ and } \varepsilon_n = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$$

I told my best friend about my finding, and she started joking about how we should consider ε in every combinatorics problem, hence the name "Epsilon" appeared.

At first, I thought about creating a game with 1×n and m×1 sized pieces, but for the game to be fair for every player, I settled for the current 1×n and n×1. Moreover, I wanted every game to have a reasonable duration, so after several tests with boards and pieces of different sizes, I found the ideal ones to be 12×12, respectively 1×4 and 4×1.

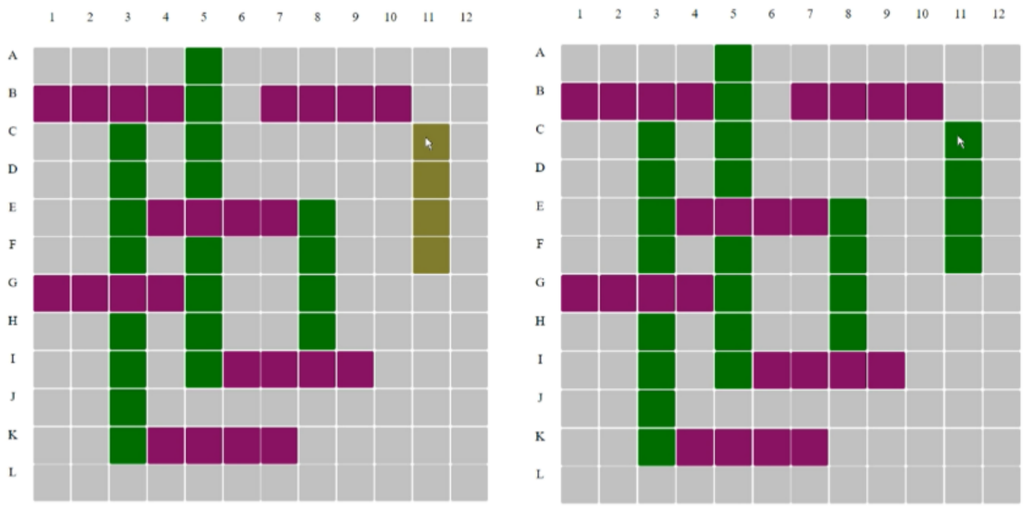
When choosing the colors, I wanted them to be both unusual and colorblind-friendly, so I went with green and purple.

Implementations of the game



The first variant of the game had 2 functionalities: the ability to place the pieces within the bounds of the board and the possibility of previewing where a piece would be placed (as shown below). The first player was always purple.

When a player hovers the mouse on a square, it will display the move they are about to make (if it is valid). When a player clicks on a square, the respective piece will be placed. For example, if the purple player clicks on the I3 square, then I3, I4, I5, and I6 will be colored purple. Conversely if the green player clicks on the C11 square, then C11, D11, E11 and F11 will be colored green.

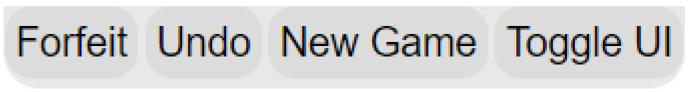


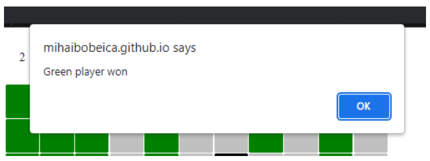
After hundreds of games played, I found an optimal move for the first player, that being D1-D4. Since the green pieces have a height of 4, this means that purple's optimal strategy is that of creating spaces of height 3. It maximizes both the win for purple and the loss for green.

The game quickly became monotonous because every match had the same optimal moves. Thus, I implemented two black squares that would randomly generate on the board to make every game unique. On top of this, the first player was going to be chosen at random.

I wanted the player to think outside (around) the box and the game to become more complex. This is how I got the idea to wrap pieces around the board. This feature changed the strategies completely. D1-D4 was not the optimal move anymore, because green could respond with L1-C1. The game changed from securing as many squares as possible to focusing more on blocking the opponent.

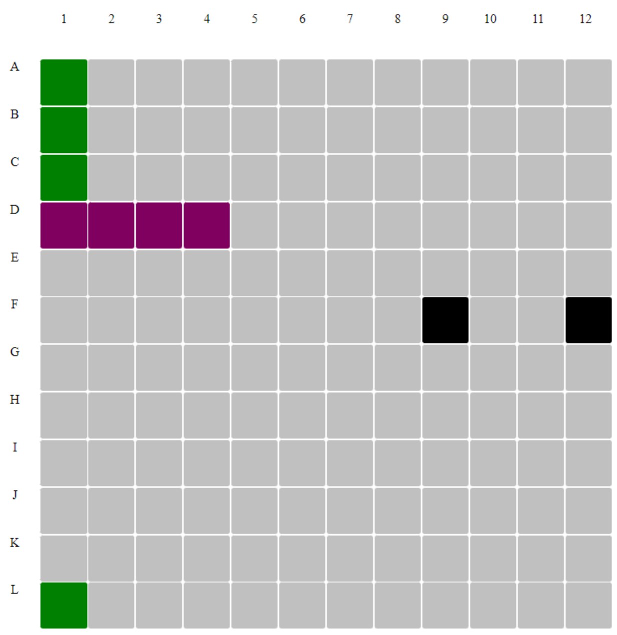
In the code, the table is represented by an array of size 145 named board[]. The element on the nth row and the mth column is associated to the element board[12n+m]. The mapping i↦board[i] is a bijection for every natural number below 146, thus every square has an UID.




mihaibobeica.github.io says

Green player won

OK



Buttons

On the top right of the page there are 4 suggestively named buttons: forfeit, undo, new game, and toggle UI. They were a later addition to the game which improved the player experience.

Winning the match

When it is mathematically impossible for a player to place their piece, a pop-up appears on top of the screen, signaling the respective winner.

Implementations of the bot

The second goal of this project is to create a bot capable of consistently winning the game against a human player.

My first idea of implementation relied on the backtracking method. The bot would search in the game tree for the best move possible. The problem with this variant is that it would have an unfeasible time complexity.

The second idea relied on alpha-beta pruning, but in Epsilon it is almost impossible to force moves, thus it was unsuitable.

After some brainstorming, I came up with a reasonable algorithm to implement. Each square is assigned the number of positions that the opponent would still be able to place in, if that particular square would be chosen as a move. The bot chooses the square with the lowest number assigned.