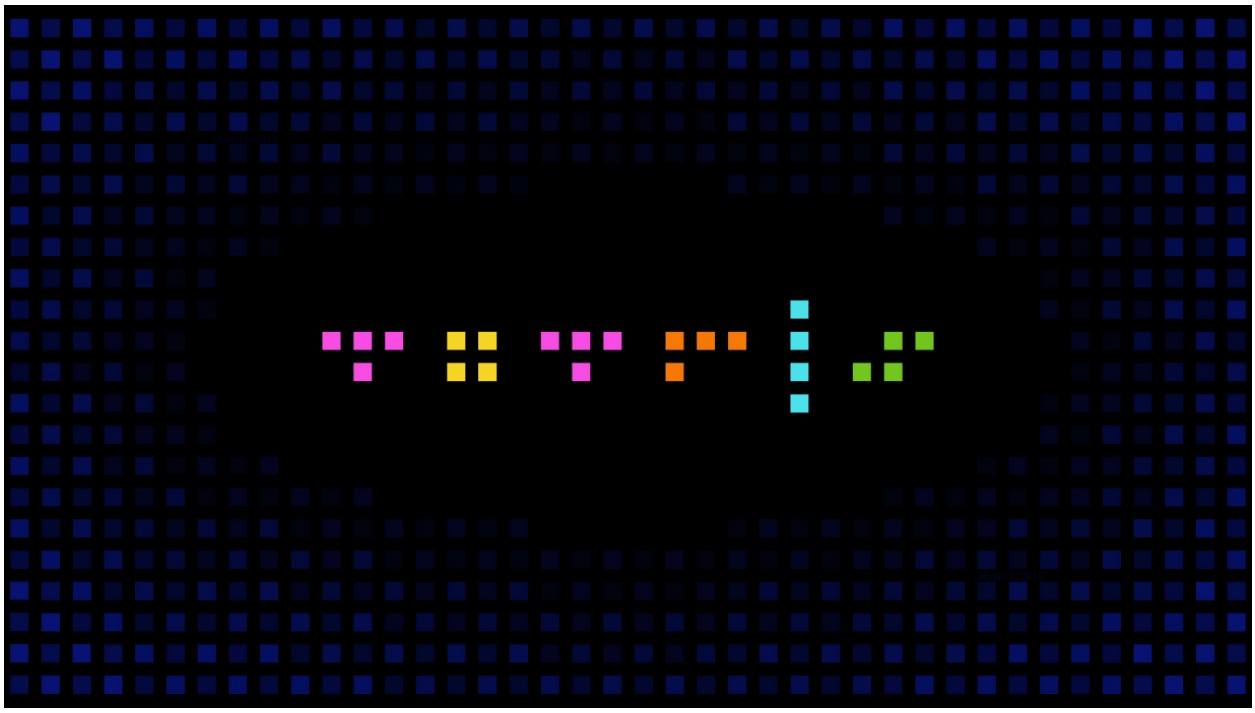


Project Name: Tetris

Author: Gherasim Teodor-Samuel

Group: 30412

What is the project about?

I am very passionate about the Tetris videogame and very intrigued about its history. Therefore, I tried to create a replica using my own style and ideas.

The game has a GUI that allows the player to start playing a new game or see how he compares to other players in the game's leaderboard. The game itself is challenging, having random piece generation and level progression.

The goal of the game is to place pieces such that you fill whole lines. Clearing a line grants points and the more points you get, the harder the game gets. Leveling up means pieces fall faster and you have less time to react.

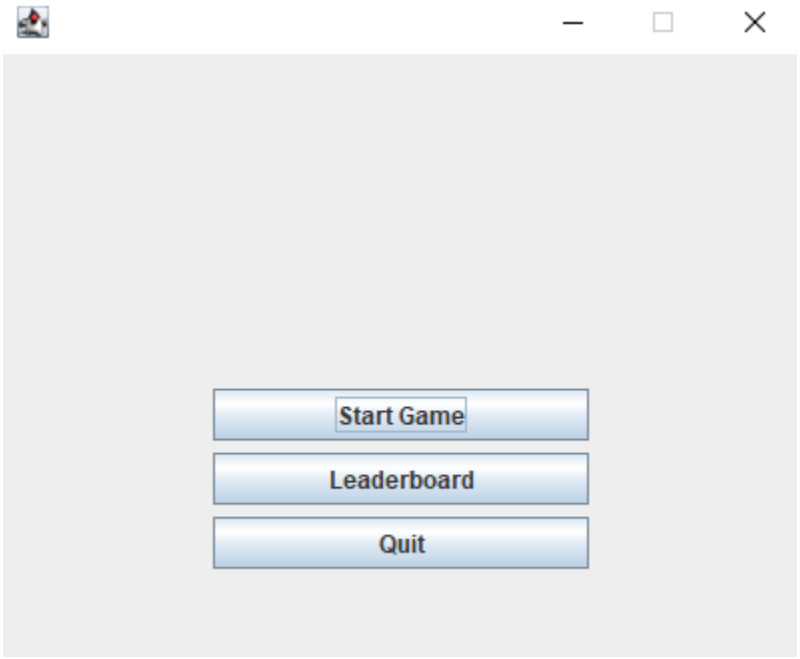
When you reach the top of the board, the game ends and prompts you to insert a name that it will then insert in the game's leaderboard.

What is the user experience?

1. **Starting Form**

When starting the game, the user is welcomed by a menu with 3 buttons:

- **Start Game –** when clicked, it opens a game form that allows the player to play the game
- **Leaderboard –** when clicked, it opens a separate Leaderboard table, that shows the current leaderboard present on the player's computer
- **Quit –** closes the game when pressed

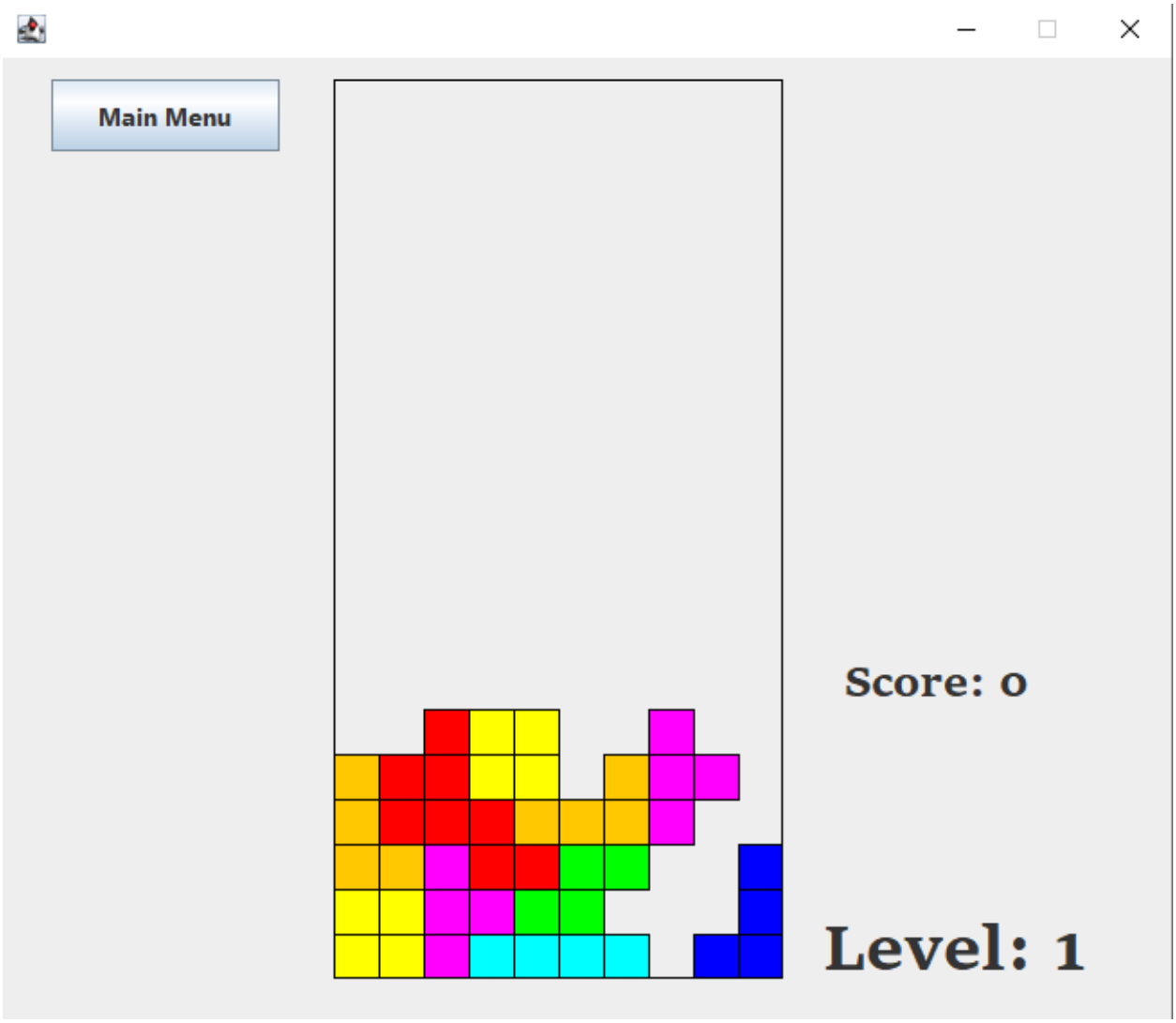


2. Game Area

This is the main part of the game. Here, the user controls falling pieces and rotates them accordingly as to gain as much score as possible. There are game sounds for dropping pieces, clearing lines, leveling up and finishing the game. The user can input any of the following using the keyboard:

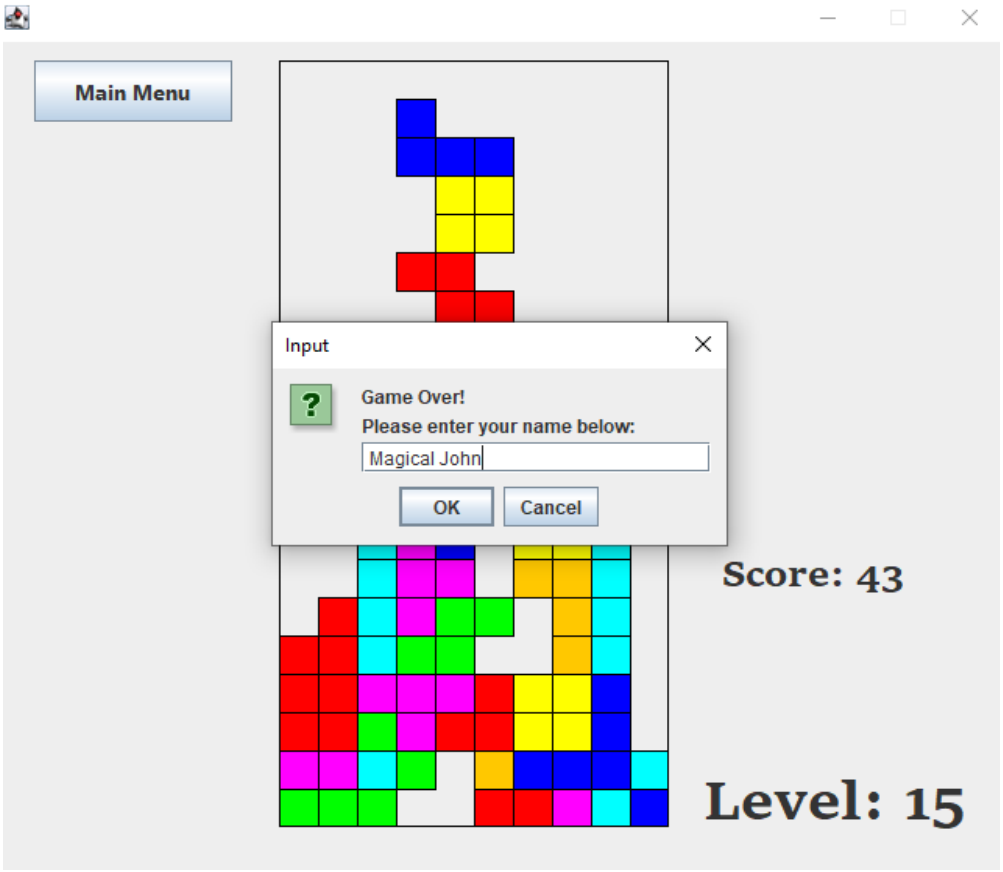
- **Space** – drops the piece to the bottom of the board and locks it in
- **Up Arrow –** rotates the piece clockwise
- **Z –** rotates the piece counterclockwise
- **Left Arrow –** moves piece to the left by one square
- **Right Arrow –** moves piece to the right by one square
- **Down Arrow –** holding it down hurries the fall of the pieces

Main Menu Button – returns the user to the main menu



Every line cleared increases the total score. Every 3 lines cleared, a level up sound is played and the game gets harder, as indicated by the level counter to the right of the window. This means pieces will fall faster and the game will be harder.

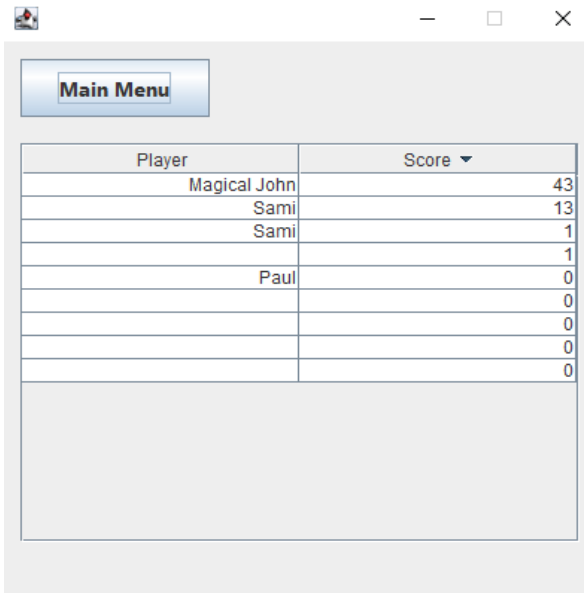
When reaching the top of the playing field, the game ends and plays music. The user is then prompted to enter his name and he is then transferred to the leaderboard to see where he stands.



3. Leaderboard

The game has a leaderboard stored in the system's memory using serialization. Therefore, the leaderboard will save it's changes whenever a new score is added, and any game reset will maintain the same leaderboard.

Main Menu Button – returns the user to the main menu



How was it implemented?

I used multithreading to run multiple programs at once. For example, the game field has a thread that is freshly generated every time a new game is run. The game window, start window and leaderboard window do not need to be recalculated, therefore they work in separate threads and are always active. This contributes to faster speeds and smoother gameplay.

All elements of the GUI and all sounds are implemented using Java Swing elements. The starting window has a JPanel and 3 JButtons. The game window has a JPanel, a JButton and 2 JLabels. The leaderboard window has a JButton and a JTable for storing data in a DefaultTableModel.

The game itself uses separately implemented subclasses of TetrisBlock, inserts them in the playing field and controls their movement according to user input. The pieces are moved on a 20x10 grid and register collisions with walls or other pieces.

The leaderboard is sorted using a TableRowSorter and a SortKey and is stored in a file via serialization.

Sounds are played by an AudioPlayer instance, through Clips.

User input is controlled using input and action mapping.

Event action listeners for all buttons are implemented with lambdas.

ArrayList and Random classes are also used to ensure random 7-bag system generation of pieces. It guarantees that any piece doesn't repeat itself for at least 7 pieces, but its generation is still random.

Conclusion

I've learned a lot about the Java language while making this project. Many tens of hours were required, but I feel I've gained a huge wealth of knowledge from all the mistakes I've made along the way. The project is by no means finished, as many aspects can be improved upon and many functionalities can be added.

In the future

I would:

- Implement a way to see the next 5 pieces in the queue and show them to the user
- Implement SRS kick table for rotating pieces against walls
- Implement a Hold system for skipping the current piece and holding it for later
- Add background images for all forms and make them more graphically pleasing
- Reimplement the GUI using JavaFX
- Add more game modes
    - Cheese mode with already existent pieces on the board when starting
    - Sprint mode - clear 40 lines as fast as possible