TETRIS

Adam Novocký: **adamnov.eth#4357**
Monika Pinteková: **lemon#1821**
Dušan Podmanický: **Swenter#2755**

We are planning to create an application of a classic game of Tetris. We believe that Tetris can be a great project for beginners in Rust as it offers a fun and engaging way to learn game development while also exploring the capabilities of the Rust programming language.

Building Tetris in Rust will help us learn several key features of the Rust programming language, including:

1. **Ownership and borrowing**: Rust's ownership and borrowing system is one of its most distinctive features. Building Tetris in Rust will give us hands-on experience with these concepts.
2. **Error handling**: Rust has a strong focus on robust error handling. Building Tetris in Rust will give us the opportunity to work with Rust's error handling mechanisms, such as Result and the ? operator, which can help us write more reliable and maintainable code.
3. **Concurrency**: Rust provides powerful concurrency primitives such as threads, channels, and locks. Building Tetris in Rust will give us experience with these tools.
4. **Traits and genetics**: Rust's traits and genetics system is a key part of the language's type system. Building Tetris in Rust can give us experience with these features and help us learn how to write generic and reusable code.
5. **Lifetimes**: Rust's lifetime system will allow us to manage memory and ensure that references are valid for as long as they are needed.
6. **Performance**: Rust is known for its performance characteristics, which make it well-suited for systems programming tasks like game development. Building Tetris in Rust will give us a chance to explore Rust's performance features, such as fine-grained control over memory allocation and deallocation.

Project requirements:

- single player Tetris game
- multiple levels with varying rising difficulty
- starting level selection
- different game modes (classic, modern)
- scoring and high score leaderboards
- background music

Dependencies:

- game engine such as **piston**, **raylib** or **bevy**
- audio library such as **rodio** or **cpal**
- random numbers from **rand** library
- timing library such as **std::time**
- threading library such as **std::thread**