Term Project

Compsec, SNU

Grading Policy

- Midterm (20%)
- Final (25%)
- Project (20%)
- Lab attendance / assignment (30%)
- Class attendance / Quiz (if exist) / Attitude (5%)

Term Project Overview

Project Title: Implementing Tetris with C++ and Raylib

Tools:

- C++
- Raylib (https://www.raylib.com/)
- CMake (Skeleton CMakeLists.txt provided)

Reference Material:

- Tetris with Raylib – Programming With Nick
- Github: educ8s/Cpp-Tetris-Game-with-raylib

Term Project Overview

What You'll Build:

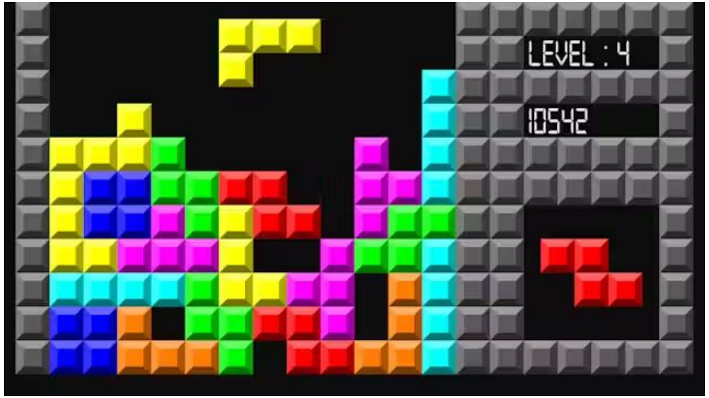
- A playable version of the classic **Tetris game** using *Raylib*.

Your Mission:

- Understand game loop logic
- Practice OOP and modular design in C++
- Extend and personalize your own version of Tetris

What is Tetris?

- Tetris is a tile-matching puzzle game where blocks ("Tetrominoes") fall from the top of the screen.
  - Your goal:
    - Move, rotate, and place Tetrominoes to **clear complete horizontal lines**.
    - Score more by clearing multiple lines at once.
    - The game gets faster over time!
  - Key gameplay features:
    - Line clear logic
    - Piece rotation
    - Holding and swapping pieces
    - Combos and scoring



What Are Tetrominoes?

- Tetrominoes are geometric shapes composed of 4 connected blocks.

- There are 7 types of tetrominoes.

- Each block:
    - Can be rotated (usually 90° clockwise)
    - Has a specific spawn orientation.

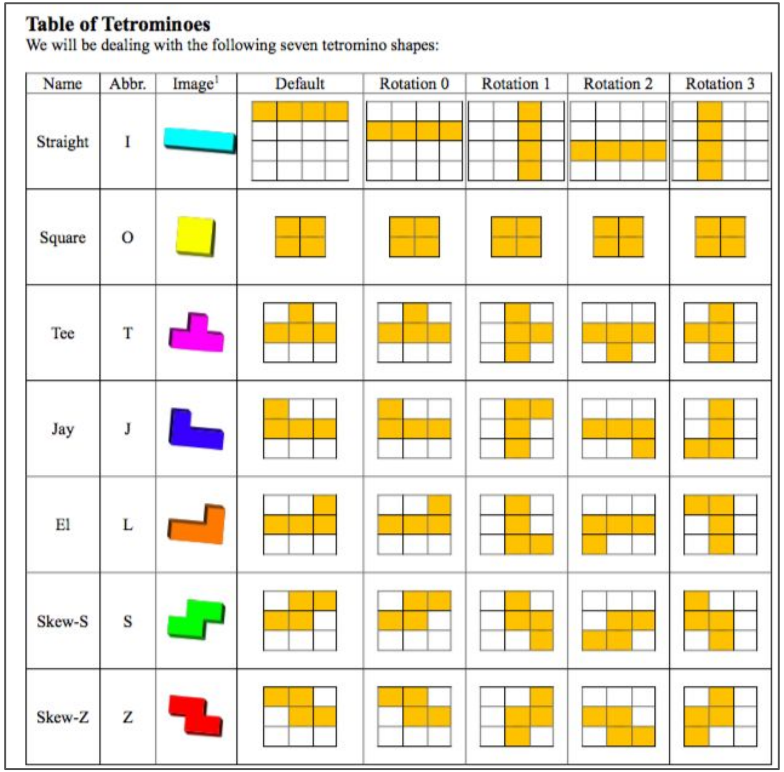
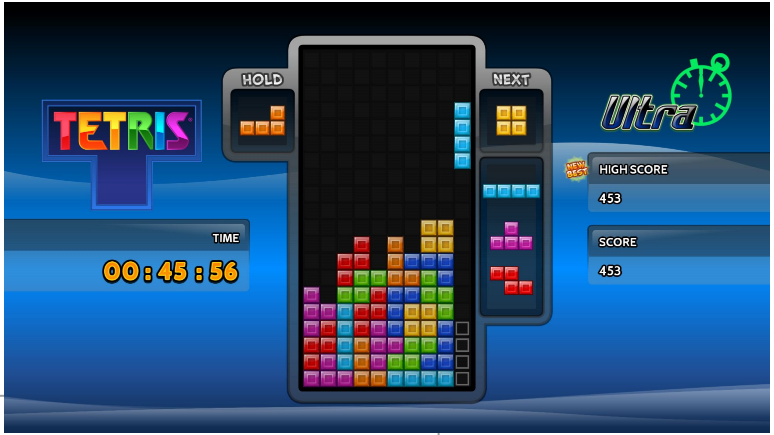


Table of Tetrominoes

We will be dealing with the following seven tetromino shapes:

Name	Abbr.	Image[1]	Default	Rotation 0	Rotation 1	Rotation 2	Rotation 3
Straight	I						
Square	O						
Tee	T						
Jay	J						
El	L						
Skew-S	S						
Skew-Z	Z						

Interface Overview



Section	Description
Hold	Shows the current held piece. The player can swap the current piece with it.
Next	Previews the next few upcoming Tetromino pieces. Helps with planning moves.
Score	Tracks the player's current score based on line clears and combos.
High Score	Displays the best score achieved in the session or all time.
Time	Shows how long the current game has been running.
Game Area	The central grid where Tetrominoes fall, rotate, and stack.

Implementation Requirements

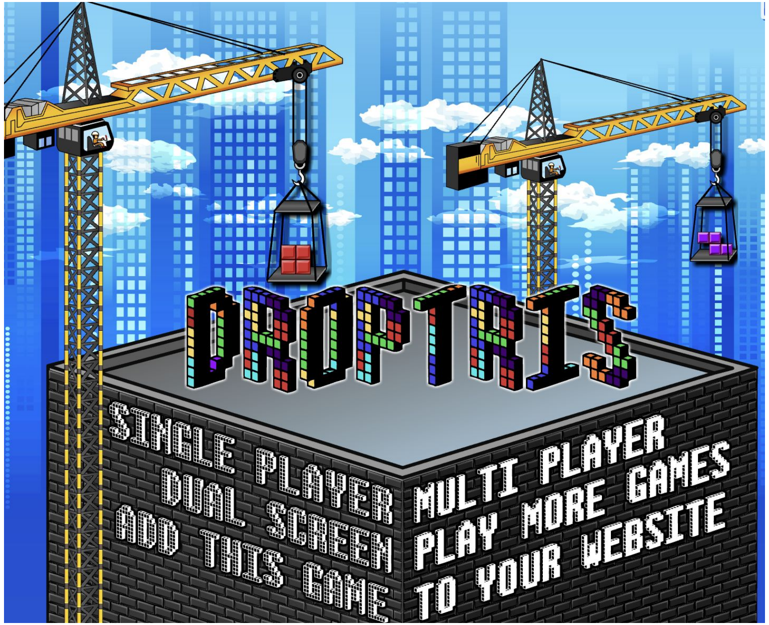
- R1. Start Page
- R2. New game modes
- R3. Tetromino Control
- R4. Difficulty Control
- R5. T-Spin
- R6. Scoring System
- R7. Ranking System

- + Optional Features

R1. Start Page

- Serves as the main entry point of the game.
    - Provides users with a clean interface to select game modes and begin gameplay.

- Features:
    - Title logo, game name, author information
        - e.g., Modified by Sangyoon
    - Menu options
        - e.g., “Marathon,” “Sprint,” “Quit”.
    - Keyboard navigation
        - e,g., ↑ ↓ to move, Enter to select.

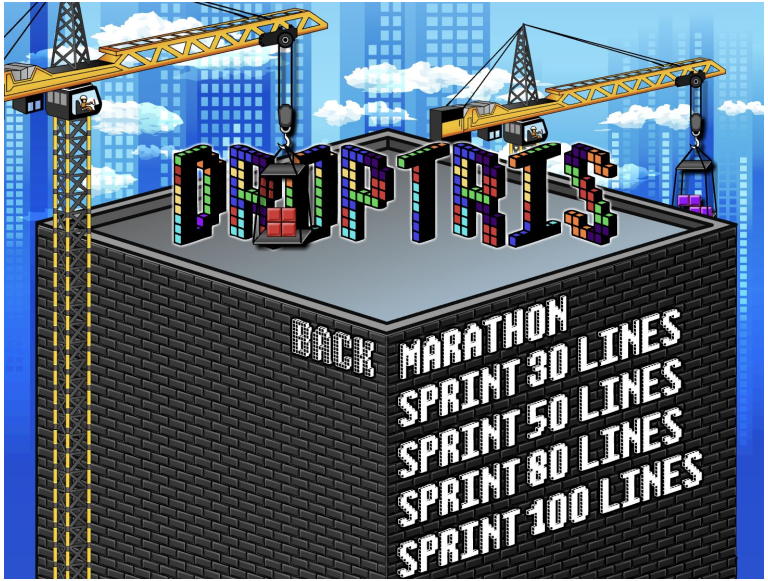


R2. New Game Modes

- Marathon (Endless Mode)
  - The game continues indefinitely until the player loses.
  - The falling speed increases progressively to raise the difficulty over time.

- Sprint (Clear N Lines)
  - This mode will challenge the player to clear a fixed number of lines (e.g., 40 lines) as quickly as possible.

- Blitz
  - Initial drop speed can vary by the difficulty level (e.g., Easy, Normal, Hard).



R3. Tetromino Control

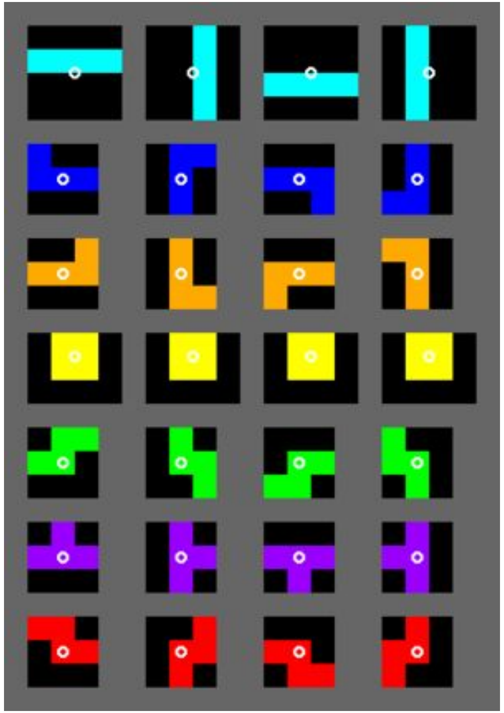
- Players can manipulate each falling block (Tetromino) using the keyboard.

- Basic Movement:
  - ← **/** →: Move Left / Right
  - ↓: Soft Drop (increases fall speed)
  - **Spacebar**: Hard Drop (instantly places block)

R3. Tetromino Control

- Rotation:
  - Z: Rotate Counter-Clockwise
  - X or ↑: Rotate Clockwise
  - Rotation follows Super Rotation System (SRS) logic

- Advanced Controls:
  - C: Hold / Swap current Tetromino
  - Ghost Piece: Shows where the block will land



Rotation

R4. Difficulty Control

- Tetrimino Speed

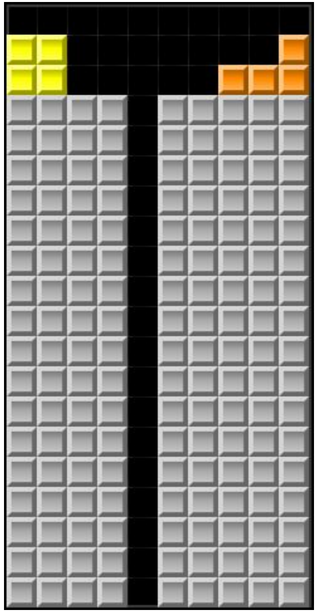
  Speed increases over time or as the player clears more lines

- Garbage Line

  Random or patterned garbage lines appear from the bottom

  Can be introduced:

  - Periodically (e.g., every 30 seconds)
  - After a certain number of lines cleared
  - In multiplayer mode (sent by opponent, optional)



Super Rotation System

- The Super Rotation System (SRS) is a standardized rotation rule used in modern Tetris games.

- Instead of simply rotating a Tetrimino in place, SRS allows the block to be **“kicked” (slightly moved)** if a rotation would otherwise be blocked by walls, the floor, or other blocks.

- Required for **T-Spin** and intuitive wall-kicks
- Reference: https://tetris.wiki/Super_Rotation_System

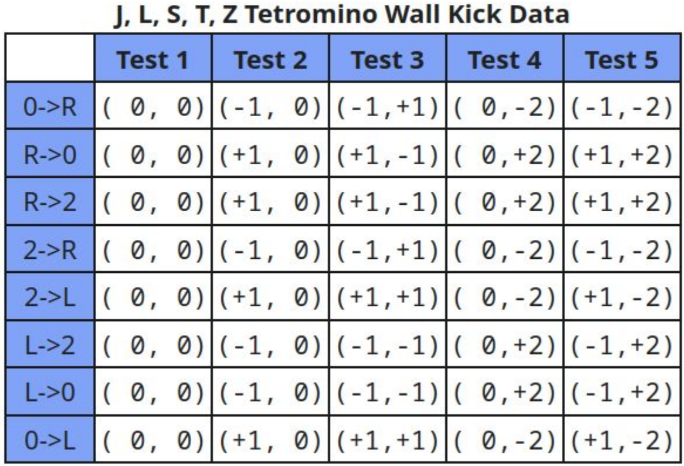
Super Rotation System

- Normally, when a player tries to rotate a Tetromino:
  - Near the wall
  - Close to the ground
  - Inside a tight space

- The rotation might fail due to collision.

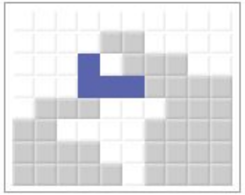
- SRS solves this by attempting to slightly shift the block to make the rotation succeed.

Super Rotation System - Example



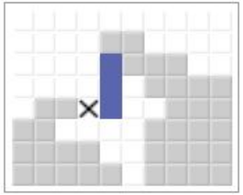
J, L, S, T, Z Tetromino Wall Kick Data

	Test 1	Test 2	Test 3	Test 4	Test 5
0->R	(0, 0)	(-1, 0)	(-1,+1)	(0,-2)	(-1,-2)
R->0	(0, 0)	(+1, 0)	(+1,-1)	(0,+2)	(+1,+2)
R->2	(0, 0)	(+1, 0)	(+1,-1)	(0,+2)	(+1,+2)
2->R	(0, 0)	(-1, 0)	(-1,+1)	(0,-2)	(-1,-2)
2->L	(0, 0)	(+1, 0)	(+1,+1)	(0,-2)	(+1,-2)
L->2	(0, 0)	(-1, 0)	(-1,-1)	(0,+2)	(-1,+2)
L->0	(0, 0)	(-1, 0)	(-1,-1)	(0,+2)	(-1,+2)
0->L	(0, 0)	(+1, 0)	(+1,+1)	(0,-2)	(+1,-2)



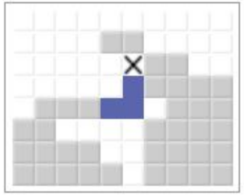
1. Initial position.

Attempt to rotate 0->L.

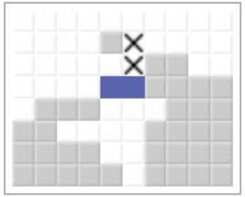


2. Test 1, (0, 0) fails.

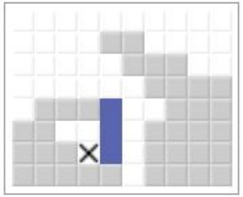
(Basic rotation fails.)



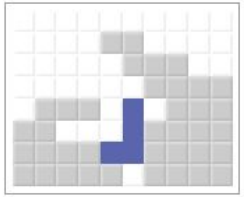
3. Test 2, (+1, 0) fails.



4. Test 3, (+1,+1) fails.



5. Test 4, (0,-2) fails.



6. Final position.

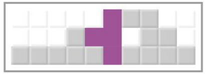
Test 5, (+1,-2) succeeds.

Super Rotation System

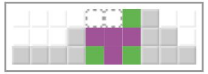
- SRS follows these steps:
  1. Attempt to rotate the Tetrimino around a reference point.
  2. If the rotation causes a collision, try predefined kick offsets (small positional adjustments).
  3. If any offset results in a valid position, the rotation succeeds at that new location.
  4. If none work, the rotation is canceled.

- Each piece and rotation direction (clockwise / counterclockwise) has a specific kick table that defines which positions to try.
- This system is especially important for the T-Tetromino, enabling advanced moves like T-Spins.

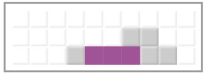
R5. T-Spin



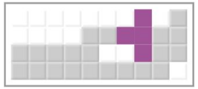
A T is dropped.



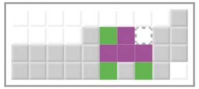
Three corners are detected, two to the front.



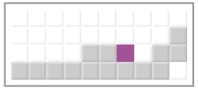
T-spin single.



Another T.



Three corners are detected, but two are to the back.



Mini T-spin single.

A T-Spin is a special move in Tetris where a T-Tetromino is rotated into a tight space using the Super Rotation System (SRS).

It is considered an advanced technique and is rewarded with bonus points.

How It Works

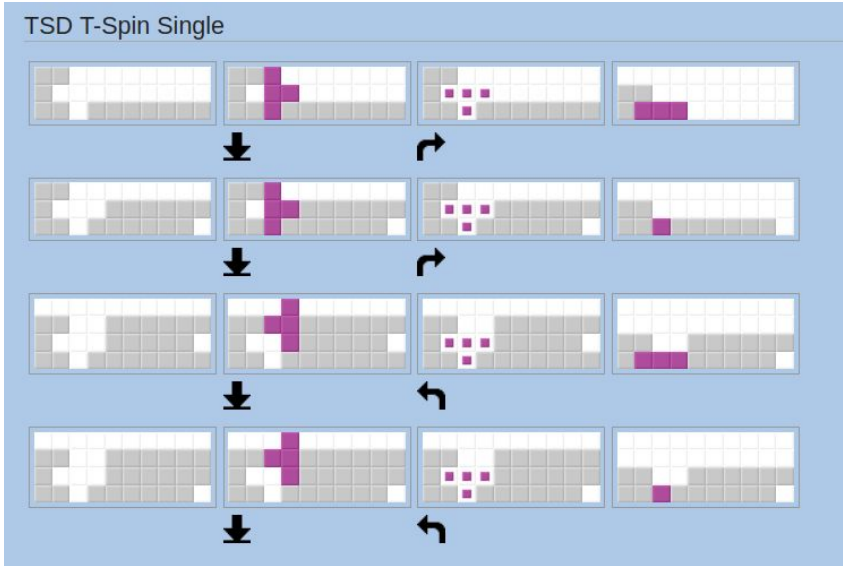
- The T-Tetromino is placed into a slot that cannot be entered by movement alone
- Using SRS kicks, the T-piece is rotated into place
- If a line is cleared immediately after the rotation, it's counted as a T-Spin

Reference: https://tetris.wiki/T-Spin

R5. T-Spin

T-Spin Types

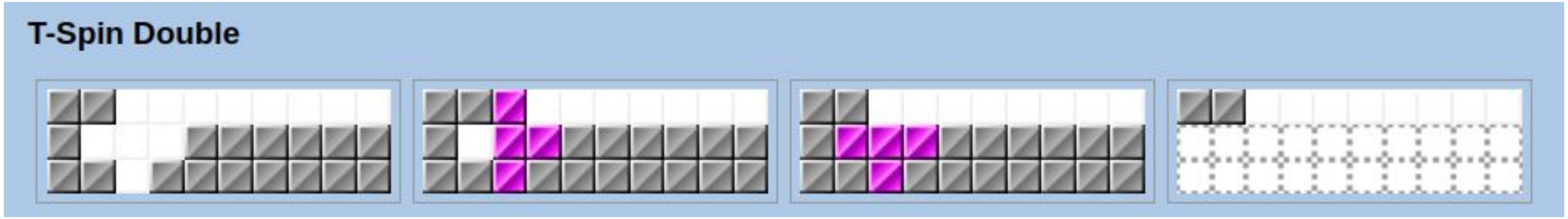
- T-Spin Single: A full spin with a line clear
- T-Spin Double / Triple: Clear 2 or 3 lines with a T-Spin



R5. T-Spin

T-Spin Types

- T-Spin Double / Triple: Clear 2 or 3 lines with a T-Spin



T-Spin Double

R6. Scoring System

- The scoring system encourages efficient, skillful, and strategic play. Points are awarded based on line clears, combos, and advanced moves.

- Score UI
  - The current score is displayed on the screen during gameplay
  - The UI is updated in real-time and typically shows
    - Score: Total points
    - Lines: Number of lines cleared
    - Level: Current speed level
    - Combo and Back-to-Back indicators

R6. Scoring System (Example)

- Basic Scoring
  - Single Line Clear: 100 points
  - Double: 300 points
  - Triple: 500 points
  - Tetris (4 lines): 800 points

- Bonus Scoring
  - T-Spin: Extra bonus based on type (Mini, Full, etc.)

R6. Scoring System (Example)

- Combo Bonus
  - Additional points for each consecutive line clear
  - Example: Combo x2 -> +50, Combo x3 -> +100, etc.
  - Resets when a drop does not result in a line clear

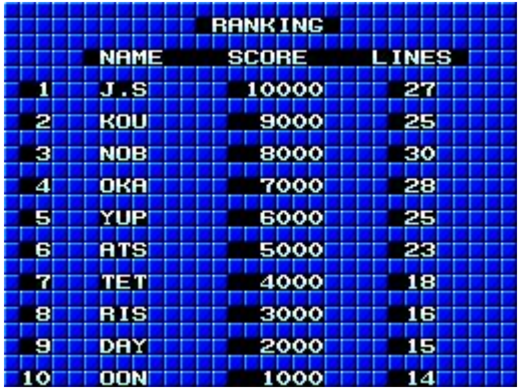
- Back-to-Back (B2B) Bonus
  - Repeated Tetris or T-Spin clears grant B2B bonuses

- Design your own scoring system!

R7. Ranking System

- The ranking system allows players to compete by tracking and displaying their best performances.

- UI Design
  - Accessible from the Start Page ("View Rankings")



RANKING

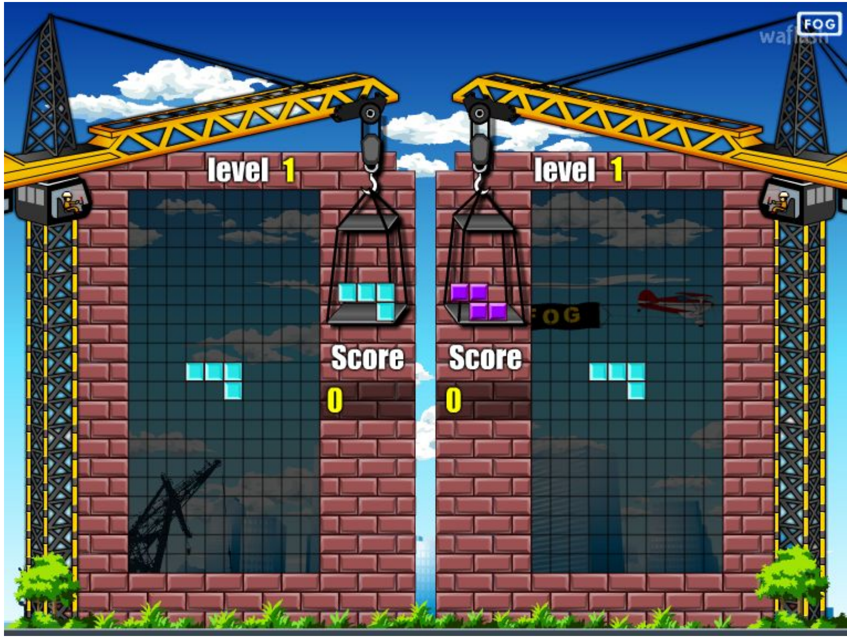
	NAME	SCORE	LINES
1	J.S	10000	27
2	KOU	9000	25
3	NOB	8000	30
4	OKA	7000	28
5	YUP	6000	25
6	ATS	5000	23
7	TET	4000	18
8	RIS	3000	16
9	DAY	2000	15
10	OON	1000	14

Optional Features

- You are encouraged to add unique features beyond the basic gameplay.
- These enhancements are not required, but they may positively influence your peer review

Optional Features

- 2-Player Mode
  - Support for local multiplayer
  - Split-screen or alternating play
  - Garbage line attack between players
  - Cooperative or versus mode



Optional Features

- Background Music & effects
  - Dynamic background music based on level or speed
  - Sound effects for line clears, drops, combos, etc.
  - Background animation (e.g., moving elements, flashing effects on Tetris)

Optional Features

- More Ideas
  - Item Blocks (e.g., clear all, slow down)
  - Replay System to view previous games

Team

- Teams may consist of 1 or 2 students
  - Individual work is allowed, but no extra credit will be given

- You must form (or join) the team!

- Your team list must be submitted by the next Wednesday (May 14th)
  - Submit your team using the online form (Also announced on eTL)
  - All students who are not associated with any team will be randomly assigned to a team.
  - By the next Thursday (May 15th), the team list will be announced.

Submission

- Code
  - Submit your full project code via Github
  - Keep the repository private during development
- Videos
  - 2-minute video: For presentation
    - With captions to “very clearly show” your game meets each requirement
  - 5-minute video: Full gameplay demonstration
    - Show all implemented features and overall game flow
- Report
  - A short (max three pages) write-up for
    - Describing the features your team implemented
    - Focus on design decisions and key implementation ideas
    - We don’t want you to spend too much time on this reporting! Focus on implementation and video!

Evaluation

- The **2-minute video** will be used for the final presentation and evaluation
- Make sure the video clearly showcase **all implementation requirements**
    - **Use captions,** annotations, or visual highlights to clearly show if you implemented the requirements!
- After the presentation session, there will be a **peer voting** to select outstanding teams
- Teams selected as **top projects** will receive **extra credit**

Tip: A well-edited, informative video can significantly impact your evaluation.

Notes & Implementation Tips

- Build your project incrementally
  - Start with core mechanics (falling block, collision, line clear)
  - Add optional features later when the base system is stable

- Use version control (e.g., GitHub) to track progress and collaborate effectively

- You may leverage GPT and other AI tools
  - Use them to get help with C++ syntax, game logic ideas, debugging, and UI design
  - But remember: You are responsible for understanding and maintaining your own code
  - Don’t copy blindly—use AI as a tool, not a crutch

- Test thoroughly after adding each feature
- Keep your code modular and well-commented for easier debugging and peer review

Notes & Implementation Tips

- **Only eTL Open Questions are allowed for project**