# Tetris Game Programming Project

## Introduction

Tetris, a timeless puzzle game created by Alexey Pajitnov in 1984, has become a gaming icon globally. Known for its simple yet challenging gameplay, players arrange falling geometric shapes to complete lines. Its addictive nature and intuitive design have made Tetris a classic enjoyed across generations and platforms.

## Technical Details

The creation of this project utilized Python version 3.12.1 along with Pygame library version 2.5.2. Pygame, a collection of Python modules intended for game development, is compatible across various platforms. It offers features for managing game graphics, sound, input devices, and other crucial aspects of game design.

An object-oriented programming methodology was employed in the development of the game. The key components include Tetrominos, the grid board, and the scoreboard, each represented by distinct classes. These elements undergo updates every second, facilitated by the Timer class constructed with *pygame.time.get\_ticks()*.

The game's user interface adopts a 2D white grid view, featuring black Tetrominos descending from the top. To introduce randomness, a 14-bag system is implemented, wherein 14 Tetrominos (2 for each shape) are gathered in the bag and randomly drawn during gameplay. When the bag is exhausted, it replenishes with another set of 14 Tetrominos.

In terms of scoring, players earn points by completing rows with blocks. If a row is entirely filled, it is cleared, and the player receives a score based on the number of rows cleared. The more rows cleared in a single move, the higher the score awarded. Additionally, players level up after eliminating every ten rows, leading to an increase in Tetromino drop speed.

To summarize, the game employs a 2D grid interface with Tetrominos falling from the top, a 14-bag system for randomization, a scoring mechanism based on clearing rows, and a leveling system that accelerates Tetromino drop speed.

## Libraries Used

* Pygame
* random
* os
* sys

## Issues faced

While developing the Tetris game, a significant challenge arose in implementing the Tetrimino rotation mechanics. The complexity arose from the need to achieve seamless and precise rotations within the limited grid space. Challenges were particularly pronounced when dealing with edge cases, where Tetriminos were positioned near the borders, resulting in unexpected behaviors and occasional collisions.

Addressing this challenge involved intensive debugging and refining of the rotation algorithm. Meticulous attention to detail and thorough testing were crucial in ensuring the consistent and reliable functionality of the rotation mechanism. Overcoming this obstacle required a careful and iterative approach to guarantee smooth rotations in all scenarios within the game's grid constraints.

## Installation and Playing Game

1. Begin by installing *Python 3.12.1* on your system.

2. Utilize the command line to install pygame by executing the command *pip install pygame*.

3. Download the source code for the project.

4. Execute the file named *main.py*.

5. Initiate the game by clicking the start button, and use the arrow keys to actively engage in gameplay.

## User Interface

The game's user interface adopts a clean and minimalist design. Upon initiation, the start page presents users with a start button to commence the game and an exit button to exit the application. Clicking the start button launches the game, while the exit button closes the game.

The game interface is divided into two sections: the game section and the score section. The top, larger part of the window constitutes the game section, where players manoeuvre tetromino blocks using arrow keys. The bottom section is dedicated to the score, displaying the current level and the total score accumulated during gameplay.

In the event of player defeat, the interface transitions to the 'Game Over!' page, mirroring the start page. From here, players can opt to restart the game or exit the application. This streamlined and straightforward user interface enhances the overall gaming experience.

## Test Cases

Here are some example test cases for the Tetris game:

1. **Movement**:

- Scenario: Verify horizontal movement of Tetrominos using left and right arrow keys.

- Expected Outcome: Tetromino moves correctly according to key presses; falls normally without key input.

2. **Speed Increase**:

- Scenario: Confirm increased fall speed by pressing the down arrow key.

- Expected Outcome: Tetromino falls faster while the down arrow key is pressed; reverts to normal speed when released.

3. **Rotation**:

- Scenario: Ensure Tetrominos rotate 90 degrees clockwise with each up arrow key press.

- Expected Outcome: Tetrominos rotate correctly on key press; remain in last rotated position without input.

4. **Collision Detection**:

- Scenario: Validate collision detection with other Tetrominos, side walls, and floor.

- Expected Outcome: Tetromino cannot pass through side walls, floor, or other Tetrominos.

5. **Scoring**:

- Scenario: Confirm correct scoring based on the number of destroyed rows.

- Expected Outcome: Score increases based on destroyed rows; varied score depending on the number of rows.

6. **Level Up**:

- Scenario: Verify level increase according to the number of rows destroyed.

- Expected Outcome: Level increases after every ten rows are destroyed.

7. **Game Over**:

- Scenario: Ensure the game over page appears after player defeat.

- Expected Outcome: Game ends when blocks pass through the top of the game window; game over page appears.

8. **Quit Game**:

- Scenario: Confirm the ability to quit the game from the start or 'Game Over!' page.

- Expected Outcome: Game closes when the exit button is clicked.

## Conclusion

The Tetris game programming project proved to be both challenging and fulfilling. Throughout this endeavor, we delved into the intricacies of game development and the Pygame library, encountering various obstacles along the way. However, with strategic planning and effective problem-solving, we successfully navigated through these challenges, gaining valuable insights into the nuances of creating engaging and functional games. This project not only expanded our technical skills but also heightened our appreciation for the creative and collaborative aspects of game design. In the end, the satisfaction of overcoming obstacles and seeing the project come to life reinforced our passion for game development.