
Om Blocks
A Tetris Clone in Python & Pygame

Project Overview

"Om Blocks" is a complete, playable clone of the classic puzzle game Tetris, built from scratch. This project demonstrates core game development principles, including state management, collision detection, and event handling, all within the Pygame framework.

The Tech Stack

Python

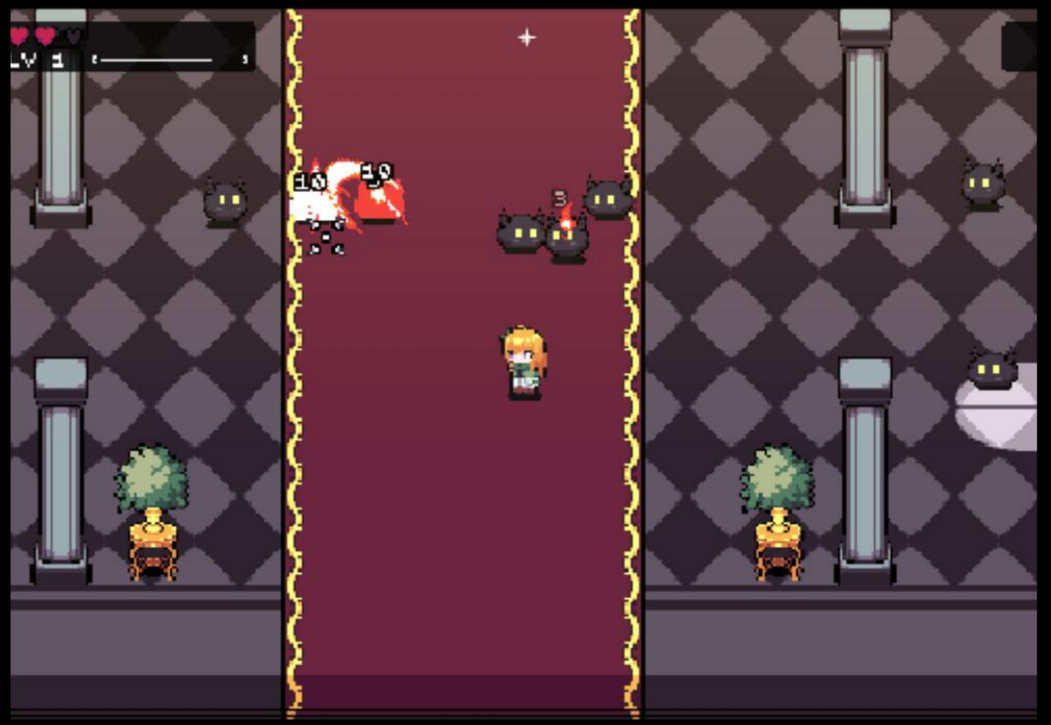
The core programming language, chosen for its readability, versatility, and powerful libraries.

Pygame

A cross-platform set of Python modules for writing video games. Used for rendering graphics, handling user input, and managing audio.

Core Game Features

- All 7 classic Tetromino shapes
- Smooth movement, rotation, and hard-drop
- Dynamic leveling and difficulty scaling
- Score system based on lines cleared
- "Next Piece" preview UI
- Pause functionality & music integration



The "Brain": Board Class

State Management

Manages the 20x10 game grid (`self.grid`). Tracks `score`, `level`, and `lines`. It's the central "source of truth" for the game state.

Core Logic

Contains the most critical methods: `valid()` for collision detection, `lock_piece()` to freeze blocks, and `clear_lines()` to manage scoring.

The "Actor": Piece Class

The Blueprint

Represents a single active tetromino. It stores its `shape` (from the `SHAPES` dict), `color`, and current grid `x`/`y` coordinates.

The Action

Contains the `rotate()` method to change its own shape. Its properties are constantly updated by the main game loop based on user input.

Challenge: Rotation

- Rotating a piece is a complex matrix transformation.
- The `rotate` function transposes the piece's shape matrix and then reverses the rows to achieve a 90-degree turn.
- The code also implements a basic "wall-kick" by checking `offset_x` if the initial rotation is invalid, making it feel smoother.

The "Heartbeat": Main Loop

1. Handle Events

Listens for user input (Quit, Pause, Left, Right, Up, Space).

2. Update State

Manages "gravity" using `fall_time`, checks collisions with `valid()`, and locks pieces to spawn new ones.

3. Render Screen

Clears the screen, calls `draw_board` & `draw_piece`, and updates the display with `pygame.display.flip()`.

Project Challenges

- Rotation Logic: Getting the matrix rotation and "wall-kicks" to feel intuitive and bug-free.
- Collision Detection: Ensuring the `valid()` function was robust for all 7 shapes in all 4 rotation states.
- Game Feel: Tweaking `fall_speed`, `FPS`, and level scaling to create a balanced difficulty curve.

Future Improvements

- Ghost Piece: Add a faded outline showing where the piece will land.
- 7-Bag Randomizer: Implement a fairer piece-spawning system to prevent long "droughts" of one piece.
- Super Rotation System (SRS): A more advanced, official wall-kick system.
- Sound Effects: Add audio cues for rotation, locking, and line clears.

Live Demo

Let's play "Om Blocks"!

Questions?

Thank you for your time.