Block Game – Development Task List

Prepared By: Muhammet Furkan Bilici

Date: 8 July 2025

This list is designed to guide the step-by-step development of the game, starting from the simplest playable version (MVP). Each "Part" represents a major component of the game.

Part 1: Project Foundations and Game Board

(Goal: Starting from an empty project, create a visible game board on screen.)

- **TASK-01: Create a new 2D Unity project and configure it for mobile platforms (Android/iOS).**
- **TASK-02: Organize the project structure by creating folders: Scripts, Prefabs, Sprites, and Scenes.**
- **TASK-03: Create a simple square Sprite representing a single grid cell, and turn it into a Prefab named GridCell.**
- **TASK-04: Create a GridManager C# script and an associated GameObject in the scene.**
- **TASK-05 [CODING]: Inside the GridManager script, write a function that draws an 8x8 board using GridCell prefabs when the game starts.**
- **TASK-06 [CODING]: Add a logical structure (e.g., 2D array) to GridManager to track filled/empty state of each cell.**

Part 2: Block Creation and Dragging

(Goal: Let the player see and drag blocks using mouse or touch.)

- **TASK-07: Create a BlockData script or ScriptableObject to hold shape data of blocks (e.g., T-block: (0,0), (1,0), (-1,0), (0,-1)).**
- **TASK-08: Create BlockData assets for at least 3–4 different block shapes.**
- **TASK-09: Create a script called BlockSpawner. This should randomly select 3 BlockData objects and visually instantiate them in a waiting area on the screen.**

- **TASK-10 [CODING]: Write a script (e.g., BlockDragger) that lets the player pick, drag, and drop blocks. (This can be done with functions like OnMouseDown, OnMouseDrag, OnMouseUp.)**

Part 3: Block Placement and Rules

(Goal: Make sure the dragged block can be placed on the board according to game rules.)

- **TASK-11 [CODING]: In GridManager, write a function (e.g., CanPlaceBlock) to check if a block can be placed at a given coordinate. It should consider board limits and filled cells.**
- **TASK-12 [CODING]: When a block is dropped, call CanPlaceBlock.**
    - **If placement is valid: Fix the block to the board and update the logical grid.**
    - **If not: Return the block to its original position.**
- **TASK-13 [CODING]: Once all 3 blocks have been placed, trigger the BlockSpawner to spawn 3 new blocks.**

Part 4: Scoring and Line Clearing

(Goal: Implement the core mechanics of line clearing and score gain.)

- **TASK-14 [CODING]: In GridManager, write a function that checks all rows and columns for completion after a block is placed.**
- **TASK-15 [CODING]: Write a function to clear filled rows/columns (visually remove them and mark them as empty in the logical grid).**
- **TASK-16: Create a ScoreManager script and a simple UI Text element to show the score.**
- **TASK-17 [CODING]: Send signals to ScoreManager to increase score and update the UI whenever a line is cleared.**
- **TASK-18 [CODING]: Add a basic combo mechanic: award bonus points if multiple lines are cleared simultaneously.**

Part 5: Game Loop and UI

(Goal: Add an end condition to the game and basic menus.)

- **TASK-19 [CODING]: Write logic to check if the player has any valid moves left after new blocks appear.**
- **TASK-20: If no valid moves remain, display a “Game Over” screen. This screen should have "Retry" and "Main Menu" buttons.**
- **TASK-21: Create a simple "Main Menu" scene with a "Play" button.**
- **TASK-22: Add a system to save and load the player's highest score using device storage (e.g., PlayerPrefs).**