Block Blast Project - Detailed Development Task List

PHASE 1: CORE MECHANICS & MINIMUM VIABLE PRODUCT (MVP)

Part 1: Project Setup and Game Board

Objective: To go from an empty Unity project to having a visible and interactive game board on screen.

- **TASK-01: Project Initialization & Configuration**
    - **Description: A new Unity 2D project will be created, and the target platform will be set to mobile (Android/iOS).**
    - **Implementation: Switch the platform via File > Build Settings. A standard folder structure (_Scripts, _Prefabs, _Sprites, _Scenes, etc.) will be created under Assets to organize the project.**
- **TASK-02: Procedural Grid System**
    - **Description: The game's 8x8 board will be generated using code (procedurally). This provides flexibility for future changes.**
    - **Implementation: A GridManager.cs script will instantiate a GridCell prefab to draw an 8x8 visual grid when the game starts.**
- **TASK-03: Logical Grid Backend**
    - **Description: Alongside the visual grid, a data structure will be established to track the state (empty or occupied) of each cell.**
    - **Implementation: A 2D array, Transform[,] logicGrid, will be added to GridManager.cs. This array will serve as the foundation for block placement and line-clearing checks.**

Part 2: Block Mechanics

Objective: To present the player with random blocks, allow them to drag these blocks, and place them on the board according to game rules.

- **TASK-04: Flexible Block Data Structure (ScriptableObject)**
    - **Description: A data structure will be created to manage all block shapes and their rotations without hardcoding them.**
    - **Implementation: A BlockData ScriptableObject will be created. This object will hold a block's cell positions (List<Vector2Int>) and its**

visual prefab. At least 8-10 BlockData assets will be created for various shapes (T, L, I, square, etc.) and their rotations.

- **TASK-05: Block Spawning**
    - **Description: A system will be developed to provide the player with a set of 3 random blocks each turn.**
    - **Implementation: A BlockSpawner.cs script will randomly select three BlockData assets and instantiate them visually in a waiting area on the screen.**
- **TASK-06: Drag-and-Drop Mechanic**
    - **Description: The player will be able to pick up blocks with a mouse or finger and drag them onto the game board.**
    - **Implementation: A BlockDragger.cs script will handle this functionality using OnMouseDown, OnMouseDrag, and OnMouseUp events. A Physics 2D Raycaster component will be added to the Camera, and Collider 2D components will be added to the block cells to enable click detection.**
- **TASK-07: Block Placement Logic**
    - **Description: A check will be performed to determine if a dropped block is within the grid bounds and over empty cells.**
    - **Implementation: A CanPlaceBlock function will be added to the GridManager. When a block is dropped, BlockDragger will call this function. If the placement is valid, the block snaps to the grid; if invalid, it returns to its initial position.**

Part 3: Game Loop and Scoring

Objective: To complete the main game loop, including line clearing, scoring, getting new blocks, and the game over condition.

- **TASK-08: Line Completion and Clearing**
    - **Description: When a row or column is completely filled, it will be cleared to make space on the board.**
    - **Implementation: A CheckForCompletedLines function in GridManager will detect filled lines. A ClearLines function will then clear these lines both visually (Destroy) and logically (updating the logicGrid).**

- **TASK-09: Scoring and Combo System**
    - **Description: A scoring system will be implemented to reward the player.**
    - **Implementation: A ScoreManager.cs will award a base score for each cleared line and an extra "combo" bonus for clearing multiple lines simultaneously. The score will be displayed via a TextMeshPro UI element.**
- **TASK-10: Game Loop Continuity**
    - **Description: When the player places all 3 available blocks, a new set will be automatically provided.**
    - **Implementation: The BlockSpawner will track the remaining blocks in a list. When the list is empty, it will trigger the SpawnNewBlockSet function to generate a new set of 3 blocks.**
- **TASK-11: Game Over Condition**
    - **Description: The game will end when the player has no valid moves left with any of the available blocks.**
    - **Implementation: An IsAnyMovePossible function in GridManager will check if a block can fit anywhere on the board. The BlockSpawner will use this check for the remaining blocks to trigger the "Game Over" state.**
- **TASK-12: UI and Scene Management**
    - **Description: A main menu and a game over screen will be created for the game.**
    - **Implementation: Two scenes, MainMenuScene and GameScene, will be created. "Play" and "Restart" buttons will use SceneManager.LoadScene to transition between scenes. A GameOverPanel will be activated when the game ends.**
- **TASK-13: Data Persistence (High Score)**
    - **Description: The player's high score will be saved даже after the game is closed.**
    - **Implementation: PlayerPrefs will be used to save the new high score to the device's local storage if it exceeds the previous one. This score will be displayed in the main menu and during gameplay.**

PHASE 2: POLISHING AND EXTRA FEATURES

Part 4: Visual and Audio Feedback

Objective: To make player actions feel more satisfying and the game more alive.

- **TASK-14: Visual Effects and Animations (Juiciness)**
    - **Description: Small animations will be added to provide a more dynamic feel.**
    - **Implementation:**
        - **Add a slight scale-up/scale-down animation when a block is successfully placed.**
        - **Add a flash or shrink effect for cells in a line before they are cleared.**
        - **Animate the score text (e.g., scale-up briefly) when points are added.**
- **TASK-15: Audio Management**
    - **Description: Sound effects and background music will be integrated into the game.**
    - **Implementation:**
        - **Create a central AudioManager.cs script to manage all audio playback.**
        - **Add sound effects for key actions: block placement, line clearing, invalid move, and button clicks.**
        - **Add relaxing background music suitable for a puzzle game.**

Part 5: User Experience (UX) and Bonus Features

Objective: To make the game more user-friendly and to increase replayability.

- **TASK-16: "Ghost Block" Indicator**
    - **Description: Display a semi-transparent "ghost" or "preview" of the block on the grid to show the player where it will land if dropped.**
    - **Implementation: During OnMouseDrag, calculate the nearest grid position and draw a translucent copy of the block at that location.**
- **TASK-17: Android Back Button Support**

- **Description: Prevent the application from closing instantly when the device's "Back" button is pressed.**
- **Implementation: Check for Input.GetKeyDown(KeyCode.Escape) in Update(). Show a "Do you want to quit?" panel during gameplay.**

- **TASK-18: Settings Menu**
  - **Description: Add a simple menu or buttons to allow the player to control audio levels.**
  - **Implementation: Control audio levels (music and SFX) via AudioListener.volume or AudioSource.mute. Save these settings to PlayerPrefs to make them persistent.**