Checkers –

Complete Technical Documentation

Version 1.0

1. Introduction

This asset provides a complete, high-quality Checkers game for Unity. It includes accurate official rules, local hotseat multiplayer, and a strong AI opponent with 5 difficulty levels. The code is modular, clean, and easy to customize.

2. Project Structure

- Scenes: MainMenu, Game
- Prefabs: AudioManager, UI prefabs, Piece prefabs, Highlight prefab
- Scripts: Core, AI, UI, Managers
- Audio: Centralized in AudioManager prefab

3. Script-by-Script Documentation

AspectRatioPreserver.cs Attached to the main Camera.

- Forces the game to maintain the 9:16 portrait aspect ratio.
- Key method: ApplyAspect() – Calculates the correct viewport rect to prevent distortion on different screen resolutions.

AudioManager.cs Singleton prefab that manages all sound effects.

- Key methods:
  - Play(AudioClip clip) – Plays one-shot sound (moves, captures, buttons)
  - PlayLoop(AudioClip clip) – Plays looped sound (AI thinking tick)
  - Stop() – Stops the current sound

DatabaseManager.cs Static class for saving/loading settings using PlayerPrefs.

- Key methods:
  - SaveDifficulty(int index) – Saves AI difficulty
  - LoadDifficulty() – Loads saved difficulty (default = Medium)

HighlightManager.cs Manages visual highlights for valid moves.

- Key methods:
  - HighlightDestinations(List<Vector2Int> destinations) – Shows highlights on valid squares
  - ClearHighlights() – Removes all highlights

InputHandler.cs Handles all player input (mouse and touch).

- Key method: OnPointerPress() – Processes clicks/taps for piece selection and move execution

Loader.cs Static class for loading scenes.

- Key method: LoadScene(Scene scene) – Loads the specified scene by enum

MainThreadDispatcher.cs Singleton for running code on the main thread from background tasks.

- Key method: Enqueue(System.Action action) – Schedules code to run on main thread

Piece.cs Attached to piece prefabs.

- Key methods:
  - Initialize(PieceColor, bool) – Sets color and king status
  - Promote() – Promotes man to king and updates visuals

Board.cs Main board logic and visuals.

- Key methods:
  - SetupStandard() – Sets up the initial board
  - MakeMove(Move move) – Executes a move (logic + visuals) and plays sounds
  - GetAllLegalMoves() – Returns all legal moves for current player
  - GetState() – Returns BoardState for AI simulation
  - Evaluate() – Evaluates the board for AI
  - GetPieceAt(Vector2Int pos) – Returns piece value at position

BoardState.cs Pure logic struct used by AI for fast simulation (cheap copies).

GameManager.cs Central game state manager.

- Key methods:
  - NewGame() – Starts a new game
  - OnMoveCompleted() – Handles turn end and win check
  - SetState(State newState) – Changes game state (Playing, Paused, GameOver)
  - PlayerResign() and AcceptDraw() – Handle game end conditions

AIManager.cs Controls AI behavior in SinglePlayer mode.

- Key method: ThinkAndMove() – Runs AI search in background thread and executes the best move

MinimaxAI.cs The AI search algorithm.

- Key method: GetBestMove(BoardState state, int maxDepth) – Returns the best move using Minimax with alpha-beta pruning

AIDifficulty.cs Static class for current AI difficulty level.

- Key method: SetCurrent(int index) – Sets the current difficulty index (0-4)

DifficultyMenuUI.cs Handles the difficulty selection screen.

- Key methods:
  - OnSliderChanged() – Updates difficulty when slider moves
  - PlayGame() – Saves difficulty and loads game scene

WinOverlayUI.cs Shows win/draw/loss screen.

- Key method: ShowOverlay(int winner) – Displays the result and plays appropriate sound

ResignConfirmationUI.cs Handles the resign confirmation popup.

- Key methods:
  - ConfirmResign() – Confirms resignation
  - Cancel() – Cancels and returns to game

HomeButtonUI.cs / **RestartButtonUI.cs** / **ResignButtonUI.cs** / **DrawButtonUI.cs** Handle their respective buttons.

- They manage button visibility, interactability, and play click sound.

TurnTextUI.cs Shows whose turn it is.

- Key method: UpdateTurnText(int player) – Updates the text ("White's Turn" / "Red's Turn")

PlayVsComputerButtonUI.cs / **PlayVsHumansButtonUI.cs** / **QuitButtonUI.cs** Handle main menu buttons.

- Open difficulty menu, load scenes, or quit the game.

4. How to Customize

- Change sounds → Edit AudioManager prefab
- Change board/pieces → Replace prefabs in Board Inspector
- Change AI strength → Modify depth mapping in AIManager.cs
- Change UI theme → Modify materials and TMP styles

5. Support

If you have any questions or need help, feel free to contact me.

Thank you for purchasing **Checkers - Offline Edition**!