Chess Game | Developer Guide

Piece.cs

This file defines how each chess piece behaves. At the top there's the PieceColor enum and a basic Piece class that stores color. Right after you'll find BoardBounds which has a method to keep row and column values inside the 0–7 range. Then come the six piece classes. Each one has a Symbol property for text output and an **IsValidMove** method that enforces movement rules and checks for clear paths or valid captures. The Pawn class handles single-square moves, two-square jumps from the starting position, and diagonal captures. Rook and King include a **HasMoved** flag so castling logic knows if they've already moved. Knight does its L-shaped jumps regardless of obstacles. Bishop moves diagonally without jumping. Queen combines Rook and Bishop logic. King allows one-square moves in any direction plus two-square castling when conditions are met.

Board.cs

This file holds the game state and the logic. It uses an 8×8 **Cells[,]** array where each cell holds a Piece or is null. **SetupDefaultPosition** fills the array with pawns and back-rank pieces in their standard spots. **CloneCells** makes a quick copy of that array so you can simulate moves without altering the real board. **WouldKingBeInCheck** runs a virtual move on the clone and returns true if the king is in check there, which helps reject illegal moves. After move validation the file provides end-of-game checks: **IsKingInCheck** looks for dangers, **IsCheckmate** makes sure no moves can escape the check, **IsStalemate** detects a draw when no legal moves remain and the king isn't in check, and **IsInsufficientMaterial** covers draw by lack of material.

ChessDisplay.cs

This file connects the board logic with the **Windows Forms UI**. In the constructor it sets window properties, defines saveFilePath for **save.json**, calls **LoadPieceImages** to load **board.png** and all piece PNGs from **/pngs** folder, then runs **LoadGameState** if a save exists. **OnPaint** draws the board background and pieces each time **Invalidate** is called, adding a white square around the selected spot and change the images when a king is in check (scared king ;)). **OnMouseClick** handles selecting a piece or moving it, including en passant captures, castling rook shifts, pawn promotions using **ShowPromotion**, turn switching, redrawing, and endgame pop-ups for checkmate, stalemate or insufficient material. There are helper methods **RestartGame** to reset the board, **SaveGameState** to write the current Cells[,] into **JSON**, and LoadGameState() to rebuild the board from that JSON.