Final Project Chess UML Diagram

[Mohamed Helwa, Umar Muhammed, Ibrahim Khan]

Board

- whitePieces: vector<ChessPiece *>
- blackPieces: vector<ChessPiece *>
- whiteKing: ChessPiece *
- blackKing: ChessPiece *
- whiteCheck: Bool
- blackCheck: Bool
- whiteQueenCastle: Bool
- whiteKingCastle: Bool
- blackQueenCastle: Bool
- blackKingCastle: Bool
- board: vector<vector<ChessPiece *>>

+ getPiece(const int row,const int col): ChessPiece*
+ printText(): void
+ makeMove(tuple<int,int> initial,tuple<int,int> dest): int
+ createPiece(int row, int col, char c, bool color, bool moved): ChessPiece*
+ getWhitePieces(): vector<ChessPiece *>
+ getBlackPieces(): vector<ChessPiece *>
+ setWhitePieces(ChessPiece *): void
+ setBlackPieces(ChessPiece *): void
+ removePieceFromVector(const int, const int): void
+ getWhiteKing(): ChessPiece *
+ getBlackKing(): ChessPiece *
+ setWhiteKing(ChessPiece *king): void
+ setBlackKing(ChessPiece *king): void
+ getBlackCheck(): Bool
+ getWhiteCheck(): Bool

Game

- chessBoard: Board
- x: Xwindow
- player: ComputerPlayer

+ initGame(): void
+ printGraphics(): void
+ updateGraphics(int row, int col);
+ drawPiece(int row, int col): void
+ makeMove(tuple<int,int> initial,tuple<int,int> dest, bool compPlayer): int
+ setupGame(char, char, tuple<int,int> pos, bool): void
+ validateConfig(): Bool
+ gameStatus(bool): int

ComputerPlayer

+ levelOne(Board*, int):Moves
+ levelTwo(Board*, int):Moves
+ levelThree(Board*, int):Moves
+ minmaxMove(Board*, int, int): Moves

Move

- initialPosition:std::tuple<int, int>
- finalPosition:std::tuple<int, int>
- board: *ChessBoard

ChessPiece

position: std::tuple<int, int>
color: Bool
type: char
hasMoved: Bool
possibleMoves: vector<Moves>
legalMoves: vector<Moves>

+ getPosition():Coordinates
+ getColor(): bool
+ getType(): char
+ getHasMoved(): bool
+ getLegalMoves() : std::vector<Moves>
+ getPossibleMoves() : std::vector<Moves>
+ setPossibleMoves(Board *):void
+ setLegalMoves():void
+ setPosition(std::tuple<int,int>)
+ checkForCheck(): void;

Pawn

- enPassant: Bool
- enPassantPosition: std::tuple<int, int>

+ setPossibleMoves(Board *): void
+ setEnPassant(std::tuple<int, int>): void
+ setNoEnPassant(): void

Rook

+ setPossibleMoves(Board *): void

Bishop

+ setPossibleMoves(Board *): void

Knight

+ setPossibleMoves(Board *): void

Queen

+ setPossibleMoves(Board *): void

King

- checked: Bool
- kingSideCastle: Bool
- queenSideCastle: Bool

+ setPossibleMoves(Board *): void
+ checkKing(): void
+ getCheck(): bool
+ uncheckKing(): void
+ getKingSideCastle() : bool
+ getQueenSideCastle() : bool