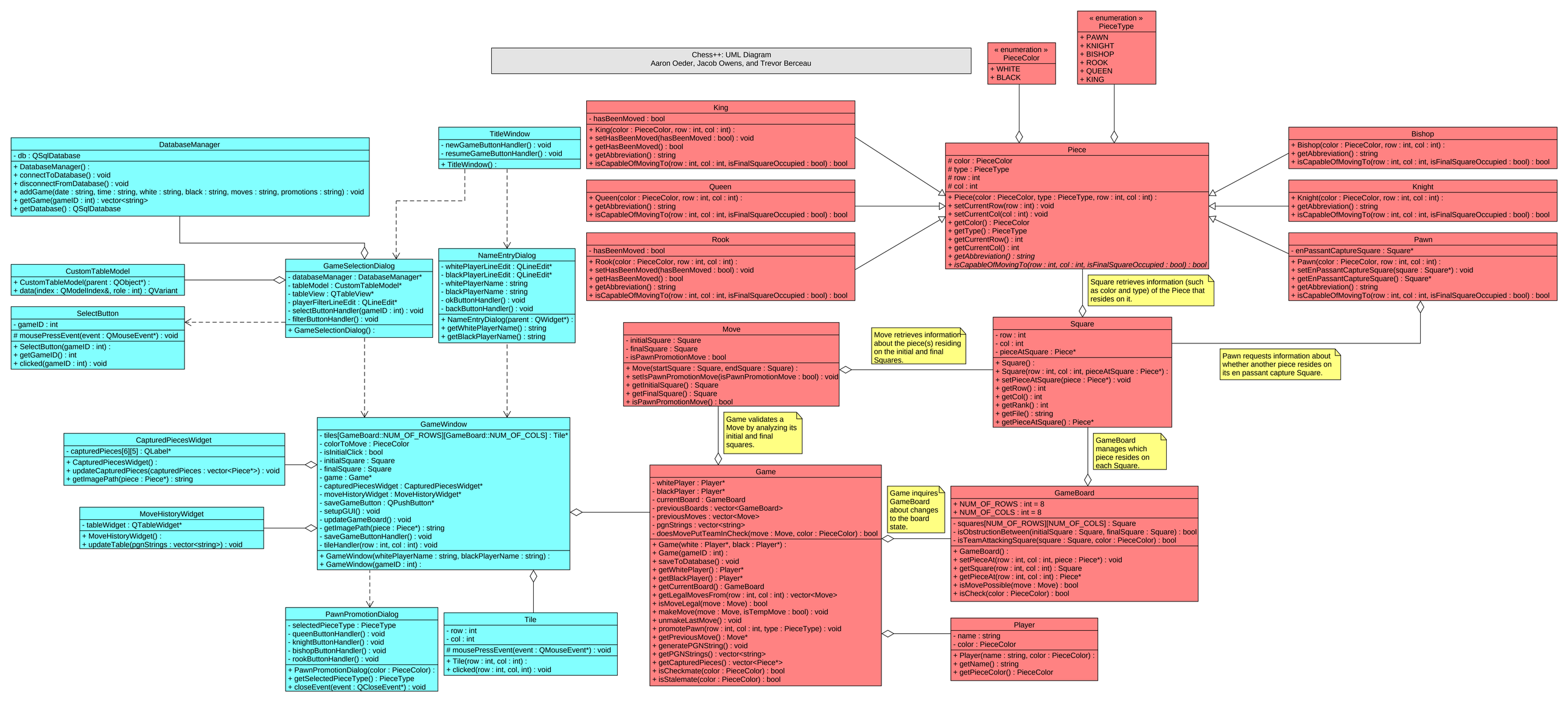
Chess++: UML Diagram
Aaron Oeder, Jacob Owens, and Trevor Berceau
« enumeration »
PieceColor
+ WHITE
+ BLACK
« enumeration »
PieceType
+ PAWN
+ KNIGHT
+ BISHOP
+ ROOK
+ QUEEN
+ KING
King
- hasBeenMoved : bool
+ King(color : PieceColor, row : int, col : int) :
+ setHasBeenMoved(hasBeenMoved : bool) : void
+ getHasBeenMoved() : bool
+ getAbbreviation() : string
+ isCapableOfMovingTo(row : int, col : int, isFinalSquareOccupied : bool) : bool
Queen
+ Queen(color : PieceColor, row : int, col : int) :
+ getAbbreviation() : string
+ isCapableOfMovingTo(row : int, col : int, isFinalSquareOccupied : bool) : bool
Rook
- hasBeenMoved : bool
+ Rook(color : PieceColor, row : int, col : int) :
+ setHasBeenMoved(hasBeenMoved : bool) : void
+ getHasBeenMoved() : bool
+ getAbbreviation() : string
+ isCapableOfMovingTo(row : int, col : int, isFinalSquareOccupied : bool) : bool
Piece
color : PieceColor
type : PieceType
row : int
col : int
+ Piece(color : PieceColor, type : PieceType, row : int, col : int) :
+ setCurrentRow(row : int) : void
+ setCurrentCol(col : int) : void
+ getColor() : PieceColor
+ getType() : PieceType
+ getCurrentRow() : int
+ getCurrentCol() : int
+ getAbbreviation() : string
+ isCapableOfMovingTo(row : int, col : int, isFinalSquareOccupied : bool) : bool
Square retrieves information (such as color and type) of the Piece that resides on it.
Bishop
+ Bishop(color : PieceColor, row : int, col : int) :
+ getAbbreviation() : string
+ isCapableOfMovingTo(row : int, col : int, isFinalSquareOccupied : bool) : bool
Knight
+ Knight(color : PieceColor, row : int, col : int) :
+ getAbbreviation() : string
+ isCapableOfMovingTo(row : int, col : int, isFinalSquareOccupied : bool) : bool
Pawn
- enPassantCaptureSquare : Square*
+ Pawn(color : PieceColor, row : int, col : int) :
+ setEnPassantCaptureSquare(square : Square*) : void
+ getEnPassantCaptureSquare() : Square*
+ getAbbreviation() : string
+ isCapableOfMovingTo(row : int, col : int, isFinalSquareOccupied : bool) : bool
Pawn requests information about whether another piece resides on its en passant capture Square.
TitleWindow
- newGameButtonHandler() : void
- resumeGameButtonHandler() : void
+ TitleWindow() :
DatabaseManager
- db : QSqlDatabase
+ DatabaseManager() :
+ connectToDatabase() : void
+ disconnectFromDatabase() : void
+ addGame(date : string, time : string, white : string, black : string, moves : string, promotions : string) : void
+ getGame(gameID : int) : vector<string>
+ getDatabase() : QSqlDatabase
CustomTableModel
+ CustomTableModel(parent : QObject*) :
+ data(index : QModelIndex&, role : int) : QVariant
SelectButton
- gameID : int
mousePressEvent(event : QMouseEvent*) : void
+ SelectButton(gameID : int) :
+ getGameID() : int
+ clicked(gameID : int) : void
GameSelectionDialog
- databaseManager : DatabaseManager*
- tableModel : CustomTableModel*
- tableView : QTableView*
- playerFilterLineEdit : QLineEdit*
- selectButtonHandler(gameID : int) : void
- filterButtonHandler() : void
+ GameSelectionDialog() :
NameEntryDialog
- whitePlayerLineEdit : QLineEdit*
- blackPlayerLineEdit : QLineEdit*
- whitePlayerName : string
- blackPlayerName : string
- okButtonHandler() : void
- backButtonHandler() : void
+ NameEntryDialog(parent : QWidget*) :
+ getWhitePlayerName() : string
+ getBlackPlayerName() : string
Move
- initialSquare : Square
- finalSquare : Square
- isPawnPromotionMove : bool
+ Move(startSquare : Square, endSquare : Square) :
+ setIsPawnPromotionMove(isPawnPromotionMove : bool) : void
+ getInitialSquare() : Square
+ getFinalSquare() : Square
+ isPawnPromotionMove() : bool
Move retrieves information about the piece(s) residing on the initial and final Squares.
Square
- row : int
- col : int
- pieceAtSquare : Piece*
+ Square() :
+ Square(row : int, col : int, pieceAtSquare : Piece*) :
+ setPieceAtSquare(piece : Piece*) : void
+ getRow() : int
+ getCol() : int
+ getRank() : int
+ getFile() : string
+ getPieceAtSquare() : Piece*
GameBoard manages which piece resides on each Square.
Game validates a Move by analyzing its initial and final squares.
CapturedPiecesWidget
- capturedPieces[6][5] : QLabel*
+ CapturedPiecesWidget() :
+ updateCapturedPieces(capturedPieces : vector<Piece*>) : void
+ getImagePath(piece : Piece*) : string
MoveHistoryWidget
- tableWidget : QTableWidget*
+ MoveHistoryWidget() :
+ updateTable(pgnStrings : vector<string>) : void
GameWindow
- tiles[GameBoard::NUM_OF_ROWS][GameBoard::NUM_OF_COLS] : Tile*
- colorToMove : PieceColor
- isInitialClick : bool
- initialSquare : Square
- finalSquare : Square
- game : Game*
- capturedPiecesWidget : CapturedPiecesWidget*
- moveHistoryWidget : MoveHistoryWidget*
- saveGameButton : QPushButton*
- setupGUI() : void
- updateGameBoard() : void
- getImagePath(piece : Piece*) : string
- saveGameButtonHandler() : void
- tileHandler(row : int, col : int) : void
+ GameWindow(whitePlayerName : string, blackPlayerName : string) :
+ GameWindow(gameID : int) :
Game
- whitePlayer : Player*
- blackPlayer : Player*
- currentBoard : GameBoard
- previousBoards : vector<GameBoard>
- previousMoves : vector<Move>
- pgnStrings : vector<string>
- doesMovePutTeamInCheck(move : Move, color : PieceColor) : bool
+ Game(white : Player*, black : Player*) :
+ Game(gameID : int) :
+ saveToDatabase() : void
+ getWhitePlayer() : Player*
+ getBlackPlayer() : Player*
+ getCurrentBoard() : GameBoard
+ getLegalMovesFrom(row : int, col : int) : vector<Move>
+ isMoveLegal(move : Move) : bool
+ makeMove(move : Move, isTempMove : bool) : void
+ unmakeLastMove() : void
+ promotePawn(row : int, col : int, type : PieceType) : void
+ getPreviousMove() : Move*
+ generatePGNString() : void
+ getPGNStrings() : vector<string>
+ getCapturedPieces() : vector<Piece*>
+ isCheckmate(color : PieceColor) : bool
+ isStalemate(color : PieceColor) : bool
Game inquires GameBoard about changes to the board state.
GameBoard
+ NUM_OF_ROWS : int = 8
+ NUM_OF_COLS : int = 8
- squares[NUM_OF_ROWS][NUM_OF_COLS] : Square
- isObstructionBetween(initialSquare : Square, finalSquare : Square) : bool
- isTeamAttackingSquare(square : Square, color : PieceColor) : bool
+ GameBoard() :
+ setPieceAt(row : int, col : int, piece : Piece*) : void
+ getSquare(row : int, col : int) : Square
+ getPieceAt(row : int, col : int) : Piece*
+ isMovePossible(move : Move) : bool
+ isCheck(color : PieceColor) : bool
Player
- name : string
- color : PieceColor
+ Player(name : string, color : PieceColor) :
+ getName() : string
+ getPieceColor() : PieceColor
PawnPromotionDialog
- selectedPieceType : PieceType
- queenButtonHandler() : void
- knightButtonHandler() : void
- bishopButtonHandler() : void
- rookButtonHandler() : void
+ PawnPromotionDialog(color : PieceColor) :
+ getSelectedPieceType() : PieceType
+ closeEvent(event : QCloseEvent*) : void
Tile
- row : int
- col : int
mousePressEvent(event : QMouseEvent*) : void
+ Tile(row : int, col : int) :
+ clicked(row : int, col, int) : void