PL_PIG_CHESS_ENGINE Package Body

Method Signatures:

CREATE OR REPLACE PACKAGE BODY PL_PIG_CHESS_ENGINE

This is the main package body declaration. It contains various procedures and functions related to the chess engine.

FUNCTION UPPER_n(n SIMPLE_INTEGER) RETURN SIMPLE_INTEGER

This function takes an integer `n` as input and returns its uppercase equivalent. It is used to convert piece representations to uppercase for comparison.

FUNCTION pdN(brik_n SIMPLE_INTEGER, felt SIMPLE_INTEGER) RETURN SIMPLE_INTEGER

This function calculates the index for a two-dimensional array based on the piece representation (`brik_n`) and the square number (`felt`). It is used to access the positional data arrays.

FUNCTION pdX(brik CHAR, felt SIMPLE_INTEGER) RETURN SIMPLE_INTEGER

Similar to `pdN`, this function calculates the index for a two-dimensional array but takes a character (`brik`) and an integer (`felt`) as input.

PROCEDURE WRT(s VARCHAR2)

This procedure writes a string `s` to the output, presumably for debugging purposes.

PROCEDURE GetNextTil(stilling, fra, til, retning, MoveTyp)

The `GetNextTil` procedure finds the next legal move for a specific piece in the position. It takes the game position (`stilling`), the current piece (`fra`), the target square (`til`), the direction (`retning`), and the move type (`MoveTyp`) as input. It returns the next legal move or sets `til` to 89 if no more moves are available.

PROCEDURE DoMove(stilling, fra, til, MoveTyp)

The `DoMove` procedure executes a move in the game position. It takes the game position (`stilling`), the source square (`fra`), the target square (`til`), and the move

type (MoveTyp) as input. It updates the game position based on the move and handles promotions, castling, and en-passant captures.

PROCEDURE DoMoveC(stilling, fra, til)

The DoMoveC procedure is similar to DoMove, but it does not require the MoveTyp parameter. It is used when the move is already validated and the move type is not needed.

FUNCTION CheckSkak(stilling, n, hvid) RETURN BOOLEAN

This function checks if a specific square (n) is threatened by the opponent's pieces. It takes the game position (stilling), the square number (n), and a boolean flag (hvid) indicating the player's turn as input. It returns TRUE if the square is threatened, and FALSE otherwise.

FUNCTION IkkeSkak(stilling, fra, til, MoveTyp) RETURN BOOLEAN

The IkkeSkak function checks if a move is illegal due to the king being in check or if the rook is moving through a threatened square. It takes the game position (stilling), the source square (fra), the target square (til), and the move type (MoveTyp) as input. It returns TRUE if the move is legal, and FALSE otherwise.

PROCEDURE GetNext(stilling, fra, til, retning, MoveTyp)

The GetNext procedure finds the next legal move in the game position. It takes the game position (stilling), the current piece (fra), the target square (til), the direction (retning), and the move type (MoveTyp) as input. It returns the next legal move or sets til to 89 if no more moves are available.

PROCEDURE Mirror(stilling)

The Mirror procedure mirrors the game position, swapping the white and black pieces. It takes the game position (stilling) as input and updates it accordingly.

FUNCTION DoMoveOk(stilling, fra, til, MoveTyp) RETURN BOOLEAN

The DoMoveOk function checks if a move is valid by generating all possible moves and checking if the move is among them. It takes the game position (stilling), the source square (fra), the target square (til), and the move type (MoveTyp) as input. It returns TRUE if the move is valid, and FALSE otherwise.

PROCEDURE ShellSort_(Trk, Upto)

The `ShellSort_` procedure sorts a portion of the `Trk` array using the Shellsort algorithm. It takes the `Trk` array and the upper bound (`Upto`) as input.

PROCEDURE QSortTrk(Trk, Fromm, Upto)

The `QSortTrk` procedure sorts a portion of the `Trk` array using the QuickSort algorithm. It takes the `Trk` array, the lower bound (`Fromm`), and the upper bound (`Upto`) as input.

FUNCTION Egain(stilling, fr, ti, OwnCount, FirstCountAttacker) RETURN SIMPLE_INTEGER

This function calculates the material gain for a specific move. It takes the game position (`stilling`), the source square (`fr`), the target square (`ti`), the count of own pieces (`OwnCount`), and the count of the first attacking piece (`FirstCountAttacker`) as input. It returns the material gain for the move.

FUNCTION QFind(stilling, Activityy, far, farfar, cf, Qdepth, Chess, farFra, farTil, farfarFra, farfarTil) RETURN SIMPLE_INTEGER

The `QFind` function is the main search function. It performs a quiescence search to evaluate the game position. It takes the game position (`stilling`), the activity level (`Activityy`), the alpha and beta values (`far`, `farfar`), the constant factor (`cf`), the search depth (`Qdepth`), a boolean flag indicating if it's a chess move (`Chess`), and the source and target squares for the previous move (`farFra`, `farTil`, `farfarFra`, `farfarTil`) as input. It returns the evaluation score.

PROCEDURE ClearHistory(cnt, black)

The `ClearHistory` procedure clears the engine's history data for a specific color. It takes the number of positions to clear (`cnt`) and a boolean flag indicating the color (`black`) as input.

PROCEDURE AddHistory(stilling, fra, til, vlu)

The `AddHistory` procedure adds a position to the engine's history data. It takes the game position (`stilling`), the source square (`fra`), the target square (`til`), and the evaluation score (`vlu`) as input.

FUNCTION Equal(stilling, still2) RETURN BOOLEAN

This function compares two game positions for equality. It takes two game positions (`stilling`, `still2`) as input and returns `TRUE` if they are equal, and `FALSE` otherwise.

PROCEDURE Find(stilling, dybde, far, farfar, cf, traek, farFra, farTil, farfarFra, farfarTil)

The `Find` procedure is the main search function for the engine. It performs a minimax search to find the best move. It takes the game position (`stilling`), the search depth (`dybde`), the alpha and beta values (`far`, `farfar`), the constant factor (`cf`), the move data (`traek`), and the source and target squares for the previous move (`farFra`, `farTil`, `farfarFra`, `farfarTil`) as input.

FUNCTION NEq(s1, s2) RETURN BOOLEAN

This function compares two game positions for inequality. It takes two game positions (`s1`, `s2`) as input and returns `TRUE` if they are not equal, and `FALSE` otherwise.

PROCEDURE FindTrk(stilling, dybde, ekstra, Traek)

The `FindTrk` procedure is the entry point for the engine's search. It initiates the search and returns the best move. It takes the game position (`stilling`), the search depth (`dybde`), an extra parameter (`ekstra`), and the move data (`Traek`) as input.

PROCEDURE GetNextQ(stilling, fra, til, retning, MoveTyp)

The `GetNextQ` procedure is similar to `GetNext`, but it does not respect check. It is used for quick move generation.

PROCEDURE GetMove(stilling, t, MoveNr, Quick)

The `GetMove` procedure retrieves the `MoveNr`-th move from the game position. It takes the game position (`stilling`), the move data (`t`), the move number (`MoveNr`), and a boolean flag indicating if it's a quick move (`Quick`) as input.

PROCEDURE GetMoveNr(stilling, fra, til, MoveNr, Quick)

The `GetMoveNr` procedure calculates the move number for a specific move in the game position. It takes the game position (`stilling`), the source square (`fra`), the target square (`til`), the move number (`MoveNr`), and a boolean flag indicating if it's a quick move (`Quick`) as input.

PROCEDURE InitValueCalc

The `InitValueCalc` procedure initializes the `ValueCalc` array, which is used to calculate the material value of pieces.

PROCEDURE InitTeo

The `InitTeo` procedure initializes the engine's opening book. It loads predefined opening moves and their frequencies.

PROCEDURE AddTeo(fromtostr, frequency, maxmoves)

The `AddTeo` procedure adds a new opening move to the engine's opening book. It takes the move in algebraic notation (`fromtostr`), the frequency of the move, and the maximum number of moves to add (`maxmoves`) as input.

PROCEDURE InitRetn

The `InitRetn` procedure initializes the `retn` array, which is used to store the move directions for each piece.

PROCEDURE Initialize

The `Initialize` procedure initializes the engine's data structures and arrays. It is called once at the beginning of the engine's execution.

Code Behavior:

The PL_PIG_CHESS_ENGINE package body contains the core logic of the chess engine. It defines various procedures and functions to generate legal moves, evaluate positions, perform searches, and maintain the engine's history and opening book.

The `GetNextTil` and `GetNext` procedures are used to generate legal moves for a specific piece or for all pieces, respectively. The `DoMove` and `DoMoveC` procedures are used to execute moves in the game position, handling promotions, castling, and en-passant captures.

The `CheckSkak` and `IkkeSkak` functions are used to check if a move is illegal due to the king being in check or if the rook is moving through a threatened square. The `Mirror` procedure mirrors the game position, swapping the white and black pieces.

The `DoMoveOk` function is used to validate a move by generating all possible moves and checking if the move is among them. The `QFind` and `Find` functions are the main search functions, performing quiescence and minimax searches, respectively, to evaluate the game position.

The `AddHistory` procedure adds a position to the engine's history data, while the `ClearHistory` procedure clears the history data for a specific color. The `Equal` and `NEq` functions are used to compare game positions for equality and inequality, respectively.

The `InitValueCalc` and `InitRetn` procedures initialize the `ValueCalc` and `retn` arrays, respectively. The `InitTeo` and `AddTeo` procedures are used to initialize and

add moves to the engine's opening book.

The `Initialize` procedure is called once to initialize the engine's data structures and arrays. It sets up the necessary variables and arrays for the engine to function properly.