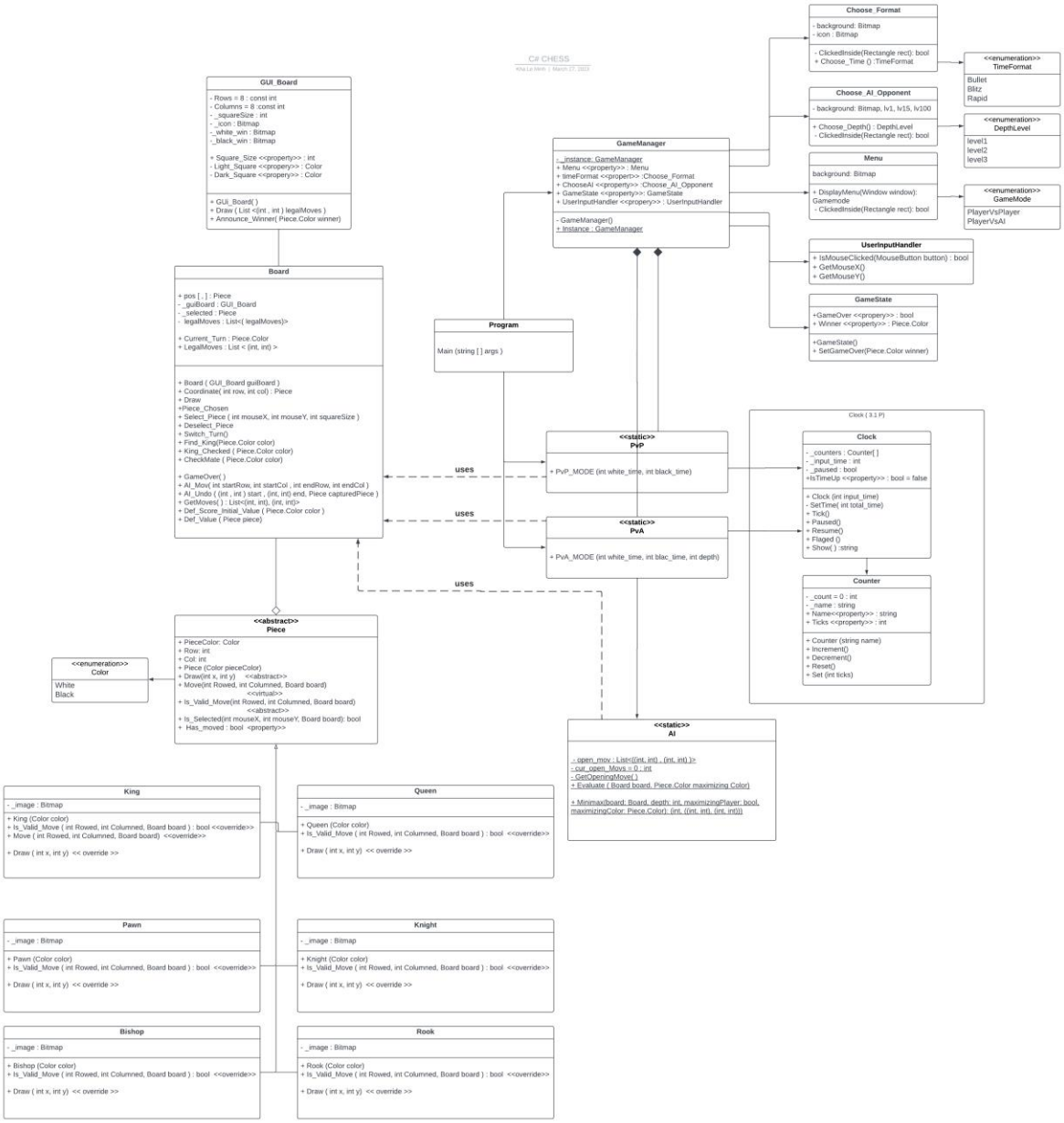
C# CHESS
Khoa Le Minh | March 27, 2023
GUI_Board
- Rows = 8 : const int
- Columns = 8 :const int
- _squareSize : int
- _icon : Bitmap
- _white_win : Bitmap
- _black_win : Bitmap
+ Square_Size <<property>> : int
- Light_Square <<property>> : Color
- Dark_Square <<property>> : Color
+ GUI_Board( )
+ Draw (List <(int , int) legalMoves)
+ Announce_Winner(Piece.Color winner)
Board
+ pos [,] : Piece
- _guiBoard : GUI_Board
- _selected : Piece
- legalMoves : List<(legalMoves)>
+ Current_Turn : Piece.Color
+ LegalMoves : List < (int, int) >
+ Board (GUI_Board guiBoard)
+ Coordinate(int row, int col) : Piece
+ Draw
+Piece_Chosen
+ Select_Piece (int mouseX, int mouseY, int squareSize)
+ Deselect_Piece
+ Switch_Turn()
+ Find_King(Piece.Color color)
+ King_Checked (Piece.Color color)
+ CheckMate (Piece.Color color)
+ GameOver( )
+ AI_Mov(int startRow, int startCol , int endRow, int endCol)
+ AI_Undo ((int , int) start , (int, int) end, Piece capturedPiece)
+ GetMoves( ) : List<(int, int), (int, int)>
+ Def_Score_Initial_Value (Piece.Color color)
+ Def_Value (Piece piece)
Program
Main (string [ ] args)
GameManager
- _instance: GameManager
+ Menu <<property>> : Menu
+ timeFormat <<property>> :Choose_Format
+ ChooseAI <<property>> :Choose_AI_Opponent
+ GameState <<property>>: GameState
+ UserInputHandler <<propery>> : UserInputHandler
- GameManager()
+ Instance : GameManager
Choose_Format
- background: Bitmap
- icon : Bitmap
- ClickedInside(Rectangle rect): bool
+ Choose_Time () :TimeFormat
<<enumeration>>
TimeFormat
Bullet
Blitz
Rapid
Choose_AI_Opponent
- background: Bitmap, lv1, lv15, lv100
+ Choose_Depth() : DepthLevel
- ClickedInside(Rectangle rect): bool
<<enumeration>>
DepthLevel
level1
level2
level3
Menu
background: Bitmap
+ DisplayMenu(Window window): Gamemode
- ClickedInside(Rectangle rect): bool
<<enumeration>>
GameMode
PlayerVsPlayer
PlayerVsAI
UserInputHandler
+ IsMouseClicked(MouseButton button) : bool
+ GetMouseX()
+ GetMouseY()
GameState
+GameOver <<propery>> : bool
+ Winner <<property>> : Piece.Color
+GameState()
+ SetGameOver(Piece.Color winner)
<<static>>
PvP
+ PvP_MODE (int white_time, int black_time)
uses
<<static>>
PvA
+ PvA_MODE (int white_time, int blac_time, int depth)
uses
uses
Clock (3.1 P)
Clock
- _counters : Counter[ ]
- _input_time : int
- _paused : bool
+IsTimeUp <<property>> : bool = false
+ Clock (int input_time)
- SetTime(int total_time)
+ Tick()
+ Paused()
+ Resume()
+ Flaged ()
+ Show( ) :string
Counter
- _count = 0 : int
- _name : string
+ Name<<property>> : string
+ Ticks <<property>> : int
+ Counter (string name)
+ Increment()
+ Decrement()
+ Reset()
+ Set (int ticks)
<<abstract>>
Piece
+ PieceColor: Color
+ Row: int
+ Col: int
+ Piece (Color pieceColor)
+ Draw(int x, int y) <<abstract>>
+ Move(int Rowed, int Columned, Board board) <<virtual>>
+ Is_Valid_Move(int Rowed, int Columned, Board board) <<abstract>>
+ Is_Selected(int mouseX, int mouseY, Board board): bool
+ Has_moved : bool <property>>
<<enumeration>>
Color
White
Black
<<static>>
AI
- open_mov : List<((int, int) , (int, int))>
- cur_open_Movs = 0 : int
- GetOpeningMove( )
+ Evaluate (Board board, Piece.Color maximizing Color)
+ Minimax(board: Board, depth: int, maximizingPlayer: bool, maximizingColor: Piece.Color): (int, ((int, int), (int, int)))
King
- _image : Bitmap
+ King (Color color)
+ Is_Valid_Move (int Rowed, int Columned, Board board) : bool <<override>>
+ Move (int Rowed, int Columned, Board board) <<override>>
+ Draw (int x, int y) << override >>
Queen
- _image : Bitmap
+ Queen (Color color)
+ Is_Valid_Move (int Rowed, int Columned, Board board) : bool <<override>>
+ Draw (int x, int y) << override >>
Pawn
- _image : Bitmap
+ Pawn (Color color)
+ Is_Valid_Move (int Rowed, int Columned, Board board) : bool <<override>>
+ Draw (int x, int y) << override >>
Knight
- _image : Bitmap
+ Knight (Color color)
+ Is_Valid_Move (int Rowed, int Columned, Board board) : bool <<override>>
+ Draw (int x, int y) << override >>
Bishop
- _image : Bitmap
+ Bishop (Color color)
+ Is_Valid_Move (int Rowed, int Columned, Board board) : bool <<override>>
+ Draw (int x, int y) << override >>
Rook
- _image : Bitmap
+ Rook (Color color)
+ Is_Valid_Move (int Rowed, int Columned, Board board) : bool <<override>>
+ Draw (int x, int y) << override >>