Chess Game

Do Nguyen Huy Hoang (Charlie)
0726845

Abstract

This paper looks at a model of a chess game created in Eden using EM principles. Chess can be a game difficult to understand depending on perceptions. Therefore this exercise provides an insight into the processes behind the creation and manipulation of the chess game in Eden. First EM is introduced briefly and the aims of the projects are stated. Then more details of the implementation with design decisions are looked at. The strengths of EM for this exercise are highlighted and all weaknesses and potential for further work discussed.

1 Introduction

Empirical Modelling (EM) is a modelling approach which is developed in University of Warwick. EM focuses on how the user can think with the computer and model their understanding. It provides open interpretation of concepts, which means the user can still change their program while it is running. Also EM contains the visualisation module Donald which can help the user build 2D models. These features of EM provide a significant degree of assistance for the implementation of a game.

Chess is a board game with 2 players. Each player has 16 chess pieces with the same colours (8 pawns, 2 rooks, 2 bishops, 2 knights, 1 queen and 1 king), so there are a total of 32 chess pieces on a board with 64 squares arranged in an eight-by-eight grid. Each player can move 1 piece each turn and can only move their pieces. The main point of the game is to capture the enemy's king or to perform a checkmate move (that is what they call in Chess).

2 Aims and Objectives

Creating a Chess game which can be played by 2 players is a main aim of this model. EM tools (Eden, Donald, Scout, Eddi) are used to build the game.
EM is based on 3 key concepts: observables, dependencies and agents. These concepts are also the key points in building a Chess game.

- Observables will be applied on the Board, the Chess pieces.
- Dependencies will be applied on the gameplay, how the chess pieces related to each other…
- Agents are the players.

The model will be divided into several parts; however, they can be summarized into 2 main parts:

- Eden part (gameplay and game theory)
- Scout and Donald part (graphics)

3 Model Design

3.1 Eden

Eden is the main tool for this project. It is used to model the gameplay and the chess theory in the project. The main point of Eden is to create a relationship between the chess pieces and the chess board. So every time one piece on the chess board is moved, the chess board will automatically update and show the result on the screen for the user (player).

The chess game is based on the chess pieces and the chess board, so observables about them must be created in order to create the game.

3.2 Chess Board

The real chess board is an 8x8 grid which contains 64 squares but inside Eden, we cannot create a matrix. So a list of 64 variables is used in this case. So if a square is changed, the list will be updated too.

ChessBoard = [square1, square2… square64];

The square number will increase from left to right and from the bottom to the top. With this model, the possible moves of the chess pieces can be calculated by maths.

3.3 Chess Pieces

The chess pieces is the square variable, when 1 chess piece is selected and moved, the chess board will change too because the chess piece now is an entity of the board. If the chess piece return "rook" in 10, so we know that there is a string rook in ChessBoard[10].

3.3 Possible Moves

As the board has been numbered, the possible moves will be calculated by mathematic functions. This solution may not be the best one, however, because the function only needs to calculate numbers, so the computation time is fast. The Eden function works well in this case.

57	58	59	60	61	62	63	64
49	50	51	52	53	54	55	56
41	42	43	44	45	46	47	48
33	34	35	36	37	38	39	40
25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24
9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8

Figure 1. Numbered Positions on Chess Board

4 Dependencies

This is the best part of EM. All the variables can be changed automatically without calling recursive function. Procedures are used to update the score, update the moves and chess board. It is also used to move the chess piece by just clicking into the screen.

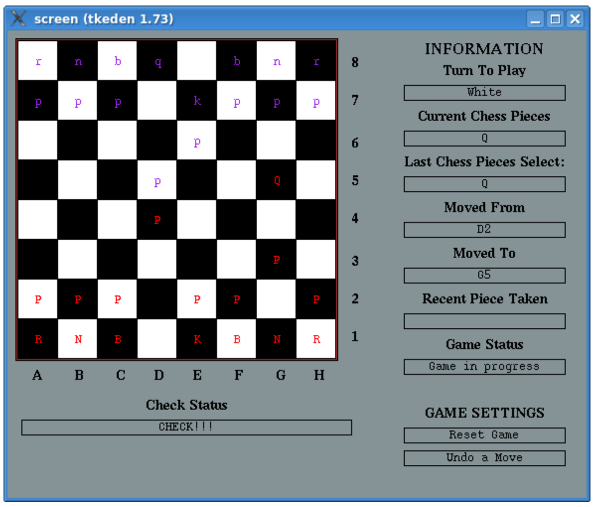
```
proc square88_makemove: square88_mouse {
        if (square88_mouse[1] == 1 ) {
                pieceSelected = "64";
        }
        if (square88_mouse[1] == 3) {
                pieceDecided = "64";
                if (pieceSelected != pieceDecided)
                {
                        make-
                move(pieceSelected, pieceDecid-
                ed);
                }
        }
}
```

Procedure of updating the value of pieceSelected and pieceDecided by the click of the mouse (square88_mouse).

5 Scout and Donald: Chess Board and Interface

Scout is used to create several windows which can display the chess squares, the text and the output box (piece selected, piece taken…). The Donald windows are linked into Eden values so that they can be updated from Eden code.

The chess board is created by 64 squares and each of them needs 2 points to form up, which is a significant task. The bad thing about Donald and Scout is the user needs to make everything from scratch (apart from built-in particle like circle).



INFORMATION

Turn To Play: White

Current Chess Pieces: Q

Last Chess Pieces Select: Q

Moved From: D2

Moved To: G5

Recent Piece Taken:

Game Status: Game in progress

Check Status: CHECK!!!

GAME SETTINGS

Reset Game

Undo a Move

Figure 2. Chess Game User Interface

The interface will contain 2 main parts: The chess board and the information boxes which can be updated automatically for the players.

6 Evaluations

This model focuses on a new game which is not currently inside the project archive of the EM webpage. Also, this game can be a good experience for people who want to learn about AI in EM, because creating a bot which can play chess in EM is also a big challenge. This model can be used to teach people how to play chess (by typing the function to find the possible moves of chess pieces into Eden input box) or to know more about EM.
This model also contains some weaknesses. First, the "undo" function is a good one which can really track up the game state and call back previous one. "Undo" function links with a database built by Eddi, but because there is no function to delete the newest

data inside the database it is still not working properly.

The chess pieces are represented by string which is not good for an interface; however, Eden keeps crashing when the images are changed so the author has decided to get rid of the pictures. Its hoped that this problem can be fixed in the future. One solution to solve this problem is using Dasm to create the chess board and inserting the picture. However, this solution is unviable due to the time required to design the chess piece models.

7 Further Works

This model can be rebuilt to display 3D graphics with the help of Dasm and Cadence. Also, it can be used to become a teaching model by making some demo games of chess. Last but not least, this model can be run on web Eden so that players now can play only through a server.

Acknowledgements

Thanks to Dr Meurig Beynon for his guidance in this project. Also, thanks to Dr Steve Russ for his idea and support in creating the game.

References

M. Beynon. Visualisation using Empirical Modelling principles and tools. *AHRC ICT Methods Network Expert Worksho.* 2007

N.Pope, M.Beynon. Empirical Modelling as an unconventional approach to software development. *Proc. SPLASH 2010 Workshop on Flexible Modeling Tools.* 2010

Draughts [online]. University of Warwick Empirical Modelling Project Archive http://empublic.dcs.warwick.ac.uk/projects/draughtsRawles1997/

Music Program [online]. University of Warwick Empirical Modelling Project Archive http://empublic.dcs.warwick.ac.uk/projects/musicWai2000/