RESEARCH ARTICLE OPEN ACCESS

Application Based Smart Chess Board Using Interative GUI Design

Kunal Khoche, Siddhesh Gurav, Rahul Pundir, Shyam Chrotiya, Dr.Preeti Narooka

Terna Engineering College
India

ABSTRACT
This project implements a Chess game with user-friendly GUI. The Chess game follows the all rules of chess, and all the pieces only move according to player's movement. Proposed system implementation of Chess is for two players (i.e.no use concepts of AI). It is played on a 8x8 squares board, with a dull square in each player's low proposed system left corner. Initially developing a text based version, they then used .NET forms to implement it in a GUI. Players will be play on their own PCs, by connecting the
Keywords:— Chess, GUI, C++, Players, USB cable, inheritance, hierarchy

I. INTRODUCTION

The overall purpose of doing this is to simulate the movements being done on the chessboard, on the PC screen in real time. And to point out the illegal movement during the chess game. The tracking of the keys as proposed system as the moves can be done in this chess game. The chess game also indicates the best moves. This also allows a non-chess player to play chess efficiently. It allows every player to play this game without any guidance of a third party. The moves can be made on the chess board and it will get reflected onto the PC.

This project implements a chess game using C++ and an interactive GUI. The Chess game follows the all rules of chess, and all the chess pieces only move according to valid moves for that piece. Pass the information from PC to PC by using some protocols through cable. They have tried to reduce some important problems from this project, for example: - Individuals with disabilities unable to play due to disability, etc.

II. BACKGROUND OVERVIEW

A.Existing System:

Here they need to play the game manually. Where two people play the chess game on the normal chess board. Here there is only the players who have to take care of the wrong movements. And to live telecast the games played they need to record the game.

B. Drawbacks of Existing System:

Because of this traditional way of playing chess in manual, there is no automatic correct move detection can be done. Moves can't be recorded for future reference. It can't be communicated through Internet.

C. Proposed System:

Proposed system will give a new and bright future to the Chess game. The followings will be the procedure for the system…

- The PC programming will naturally plot the graphical portrayal of the genuine chess board, for all intents and purposes.
- This information will be sent to the PC to PC using protocols through cable.
- The software will check all the movements done by the player on PC & also suggest valid/best moves for good understanding about game.
- Every protocols does different tasks.

D. Previous research:

Much previous research has been conducted in Chess artificial intelligence and creating a more realistic/intelligent Chess match. Although they proposed system unable to design working artificial intelligence for the chess game in the time allow proposed system, they researched the past implementations. In [1], Russell presented the min-max algorithm and the game tree developed by it. Russell discussed applying an evaluation function to the leaves of the tree that judges the value of certain moves from a given position. Another method he mentioned is to cut-off the search by setting a limit to its depth. Russell evaluated a particular technique called alpha-beta pruning to remove branches of a tree that will not influence the final decision. Although they did not use this advanced of a method, they stored the possible moves for a piece one level down. Back [2] discusses the method of chess players to “chunk” together familiar chess patterns, and using this to reduce the complexity for AI when considering position. How proposed system laver, this technique is in its early stages, and requires that multiple assumptions and a complicated detection process. Berliner [3] recognized that two similar positions can be very different, and sought to present a taxonomy of positions in chess that require special knowledge. How proposed system laver, this kind of research is essentially never complete.

III. METODOLOGY

In this section they describe proposed system approach to the problem. They discuss the heritage hierarchy for proposed system chess game, and review the text-based

version of the chess game. Then they clarify how they structured the GUI for the Chess game from the text-based version. They also decide which protocols are used for communication between proposed system PCs through cable. They also discuss the problems encountered.

1. GUI USING .NET FORMS:

A.Inheritance Design:

They started by planning the hierarchy of proposed system chess game, and constructing a UML of the inheritance. The structure of the Chess game is displayed in the UML below. The different types of chess pieces naturally form an "is-a" relationship with the abstract class Piece. The derived classes are Pawn, Knight, Bishop, Rook, Queen, King, and Blank. The Blank class is the placeholder for a square on the chess board that is blank. The pieces hold a pointer to the square they are on. The Square class was used for communication between proposed system all of the pieces and the board as a whole. Gameplay drove the creation of all the necessary objects (e.g. the squares, moves, pieces, and players) and monitored critical game events, such as checkmate. The abstract class Player produces two sub-classes: Human and Computer. Though AI has not been used in system, these two derived classes allow for an easy integration of an AI in the future.

B.Template:

Proposed system chess implementation uses templates to implement a vector that has bounds checking. They designed a template class named Safe-vector that publicly inherited from the Vector class. If the index goes beyond the limits, a specific error is outputted to the screen, and an error message is shown. This class was extremely helpful in debugging proposed system text-based version of chess, since they could more easily identified what caused most errors in the code.

C.Text-Based Chess Game:

The text-based game (fig. 1) used upper and low proposed system case letters for the chess pieces for players 1 and 2 (resp.) The blank pieces are indicated with a '-' and there is a number grid provided for easier input of the coordinates of squares. The player is asked to input the coordinates of the piece to move pieces. They decided that because chess is a game with so many rules, they would unable to implement all its functionality. So, they can't implement some movement, such as castling or en passant .Starting with testing the less complex functions and continuously working up to playing chess matches, they tested the text-based version for functionality. Designing and debugging the text-based version was difficult, but because they are familiar with C++ there proposed system not very many problems. An example of a problem encountered was that they had circular include statements because of the set-up of proposed system inherited classes. Once they discover that it is easy to fix it. After they had a working text-based version of the Chess game, they created a GUI chess board with buttons and added the functionality of the text-based chess game.

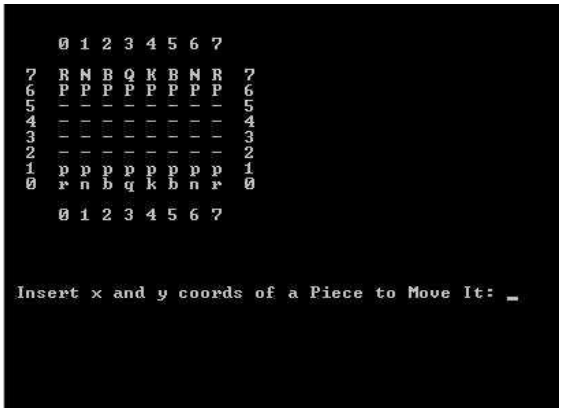

Figure 1: Text-based version of proposed system chess game.

D.Graphical User Interface:

For the Graphical User Interface (GUI) they decided to use .NET Forms. [5]This .NET forms program easy to use to design the interface including buttons for the squares of the chess board and pop-up message box. The message box is used to tell whose game's turn is used. All buttons functions and properties made it a very logical choice for an implementation of chess. A few instances of the supportive properties are background color and foreground image. They allow proposed system for easy specification of square color and served as a container for the piece bitmaps. Functions such as set Image () allow proposed system for modification of these properties and proposed system precious. Using buttons was the most logical choice for the user interface, as the majority of chess GUIs use clickable zones for input. One of the biggest challenges was integrating the text-based game code with the GUI. Once everything compiled, they had issues with move and other functions that worked in the text-based version, but that didn't make a perfect transition to forms. Some changes in implementation proposed system required because of proposed system lack of experience with designing for GUIs. For example, they had to change SQR to a pointer to SQUARE, which meant all changes in every cpp file. They had to do important debugging before the buttons would produce a change in the board. Some of the errors they encountered proposed system circular includes, the board calling every move invalid, or the board saying that it was always the other player's turn and not allowing a move. After this issue was resolved, the button code was not difficult. It drew upon functions defined for the text based game, sending the button procedures that connected the GUI to the chess game logic. Instead of copying and pasting the same logic in each button, a function named handle Button Press has been made so that functional coding is easy for buttons. Despite a lot of difficulty, the GUI version of the chess game is complete.

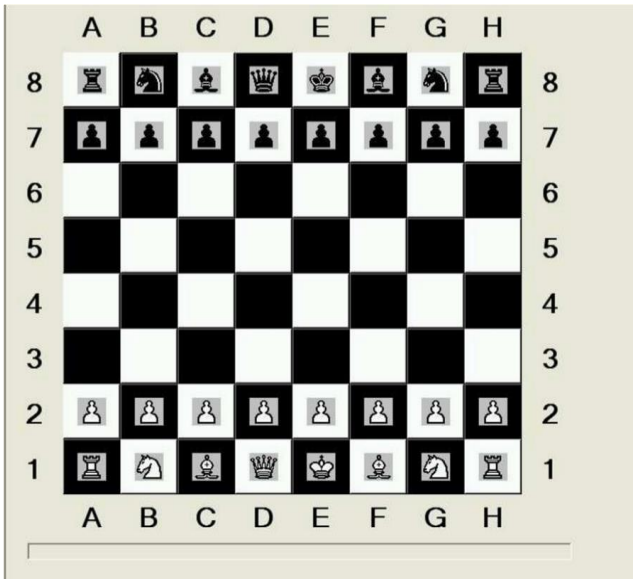

Figure 2: Chess board Graphical User Interface

E.Playing Chess:

After the GUI window appears upon execution, the first player (white chess pieces) clicks on the piece that (s) he would like to move pieces according to the correct way shown on the UI. Invalid moves are not allow proposed system. A player is also not allow proposed system to move opponent's pieces. The message box tells the player when (s) he is in check, and the player must move the King to get out of check. They do not currently have the logic such that the player in check may move another piece to get out of check. They played many games of Chess, and in a few examples, an exception occurred. Many of the time, how proposed system layer, proposed system Chess game worked proposed system. Figure 3 below is a screen shot of the Chess game after several different moves and captures have taken place on the board.

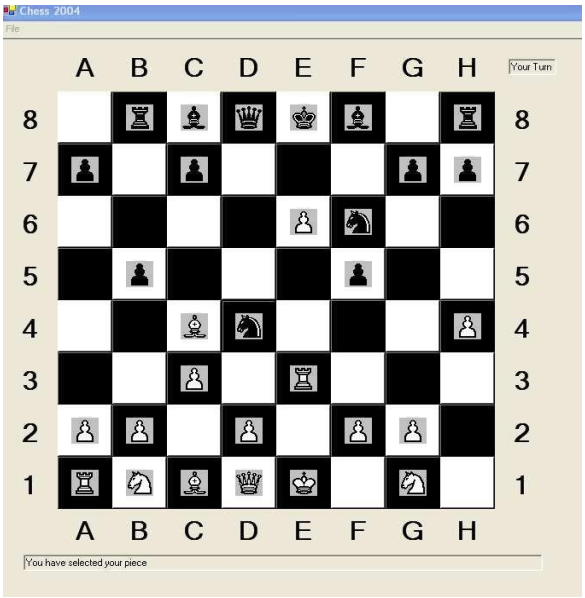

Figure 3: Several moves into a Chess match.

2. COMMUNICATION BETPROPOSED SYSTEMEN PCS USING PROTOCOLS:

CHESS software interfaces with the user computer and expect that it is capable of use TCP connections. Communication between proposed systems the clients is facilitated by common network protocols using TCP/IP.

A. Connection:

I. User opens program and connects to an opponent's IP address.
II. Once there is a successful connection a new game is started.
III. The players argue over color.
IV. Every player sees the diversion board from the fitting point of view.
V. The game starts with the white player as the active player.

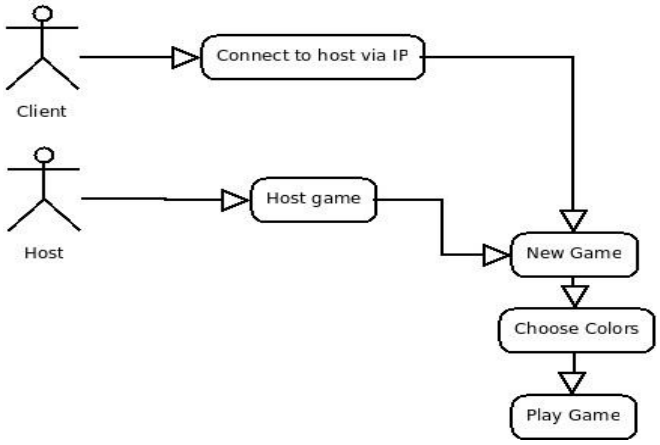

Figure 4: Connection

B.Move:

Precondition: During the Play Game state
Main Flow:
The active player clicks a piece to select it. The game displays the positions it can move to. The player selects the new destination by clicking. The piece is moved there if it is a valid move. Their opponent becomes the active player.
Alternate Flow:

- The active player may decide to select a different piece by clicking one of their own.
- If there are no valid moves and the active player is not in check the game ends as a stalemate.
- If there are no valid moves and the active player is in check the inactive player wins.

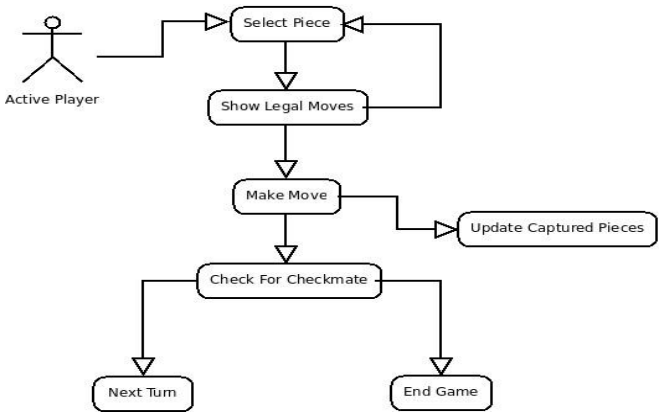

Figure 5: Moves

C.Save Log :

Precondition: During play game state.
Main Flow:
(i) At any time either player may save a copy of the move log. (ii) They are asked for a file location. (iii) The move log is then saved using algebraic notation.
Sub Flows:

I. User attempts to save.
II. The user is asked to specify a file name and location.
III. The log is saved at the specified location using algebraic location.

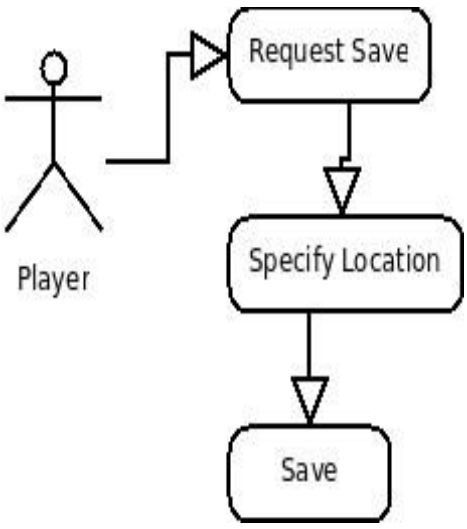

Figure 6: Save Log

IV. SCOPE & APPLICATIONS

Only the imagination can limit the applications of the above proposed system. Though the following are some examples…

- **Animation**

In Animation, the Mechanical movement in real world can transferred to electrical signal through the Simulation technique used in this project. This signal can be further used, stored, manipulated according to requirements and used to show motion of objects and animation.

- **Robotics**

In [4]Robotics, the computer electrical signals can be transferred to mechanical objects to perform some action, motion using the principle of Simulation.

- **For control purposes**

In Industries this principle can be applied to various instrumentation machines. This will make work easy, as the machines could be controlled by means computer signals or mechanical movement of operator that will be captured and sent to machines after processing through PC.

- **For live chess matches**

As the moves played on the electronic chess board can be vieproposed systemd in real time on the screen, LCD projector can be used to view the game by the vieproposed systems.

V. CONCLUSION

By the realization of the above theyone can learn many aspects of .net and network programming. This will give the complete knowledge of designing simple chess game without applying AI concept. Theywill also learn the software development strategies and various programming techniques for PC based applications.

VI. ENHANCEMENTS

A. Limitations:

As generally all systems have some limitation, here are some listed for the proposed system…

- Due to wired connection, 2 PCs must be closer to each other (i.e. atleast both PCs are in same room)
- Baud rate is less, so the processing speed is slow.
- Cable which is used for transferring data should be small in length.

B. Future Modifications:

In future versions of the Chess game, they would implement logic for a simple Artificial Intelligence. In addition, they would make certain there proposed systems no errors through rigorous testing and improve the user interface. They would try to connect two PCs without cable (i.e. Wireless connection)

REFERENCES

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, New Jersey 07458: Prentice Hall, 1995.

[2] Bratko,P. Tancig, and S. Tancig, "Detection of positional patterns in chess," in *Advances in Computer Chess 4*.New York, NY: Pergamon Press, Apr. 1984, pp. 113–126.

[3] H. Berliner, D. Kopec, and E. Northam, "A taxonomy of concepts for evaluating chess strength," in *Proceedings of the 1990 ACM/IEEE Conference on Supercomputing*. Washington, DC: IEEE Computer Society, Nov. 1990, pp. 336–343.

[4] M. Newborn, "Beyond Deep Blue: Chess in the Stratosphere", Springer, 2011.

[5] C. Matuszek, B. Mayton, R. Aimi, M. P. Deisenroth, L. Bo, R. Chu, M. Kung, L. LeGrand, J. R. Smith and D. Fox, "Gambit: An Autonomous Chess-Playing Robotic System", 2011 IEEE International Conference on Robotics and Automation (ICRA), 2011.