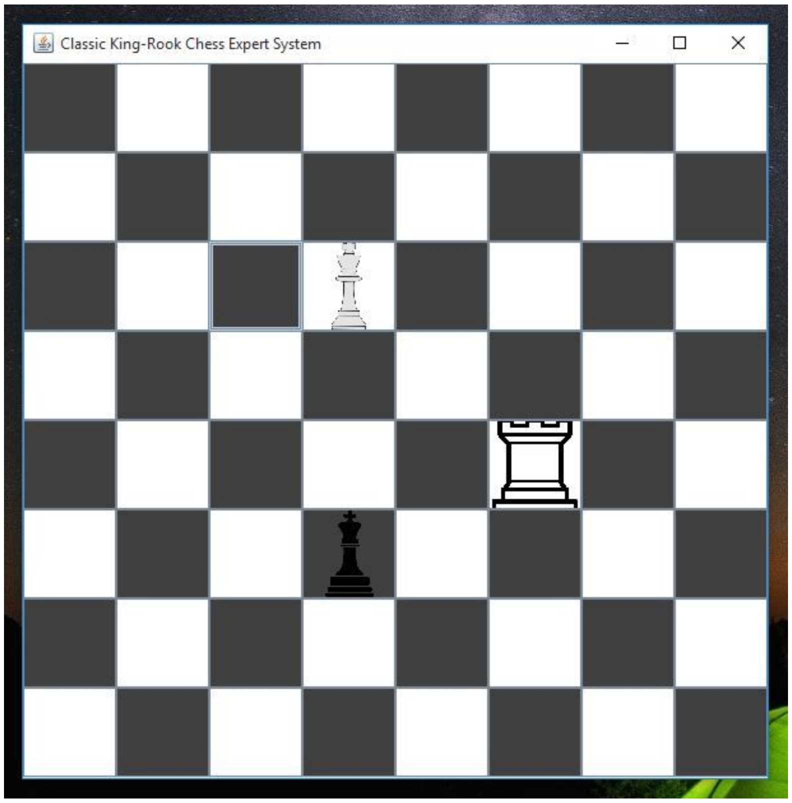
Adversarial Expert System

Classic King and Rook Chess Endgame



Satyam Poddar -	IIT2014505
Rajat Saxena-	IIT2014503
Naveen Kumar Shukla -	IIT2014154
Abhinav Mishra -	IIM2014003
Lakshay Gupta-	IIT2014044

CANDIDATE’S DECLARATION

We certify that the work embodied in this project is our own bonafide work carried out by us under the supervision of Dr. Anupam Agarwal during the fifth semester at the Indian Institute of Information Technology, Allahabad. The matter embodied in this work has not been submitted for the award of any other degree/ diploma. We declare that we have faithfully acknowledged, given credit to and referred to the research workers wherever their works have been cited in the text and the body of the project. We further certify that we have not wilfully lifted up some other’s work, paragraph, text, data, results, etc. reported in the journals, books, magazines, reports, dissertations, thesis etc. or material available at web-sites and included them in this project and cited as our own work.

Satyam Poddar -	IIT2014505
Rajat Saxena-	IIT2014503
Naveen Kumar Shukla -	IIT2014154
Abhinav Mishra -	IIM2014003
Lakshay Gupta-	IIT2014044

Dated: 23rd of November, 2016.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Dr. Anupam Agarwal at the Indian Institute of Information Technology, Allahabad for sharing their pearls of wisdom with us during the course of this research.

We are also immensely grateful to our friends for their comments on an earlier version of the manuscript and providing insight and expertise that greatly assisted the project.

We are also thankful to the institute for providing us with the necessary resources required for the timely completion of the project.

INTRODUCTION

The **Classic King and Rook chess endgame** is the stage of the game when few pieces are left on the board. This report will illustrate the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and complete declaration of the system. It will also tell how the system will respond to external stimuli and its behaviour when interacting with external application main purpose of this document is to be used as a reference for developing the initial edition of the system for the development team.

PROBLEM DEFINITION

This software system will be “Classic King and Rook Chess Endgame” that will be able to stimulate the chess game when Rook and King of opponent is against the only left King of the player. It is a classic chess state where the objective of the player is to avoid Check-Mate for as many moves as possible.

FEATURES

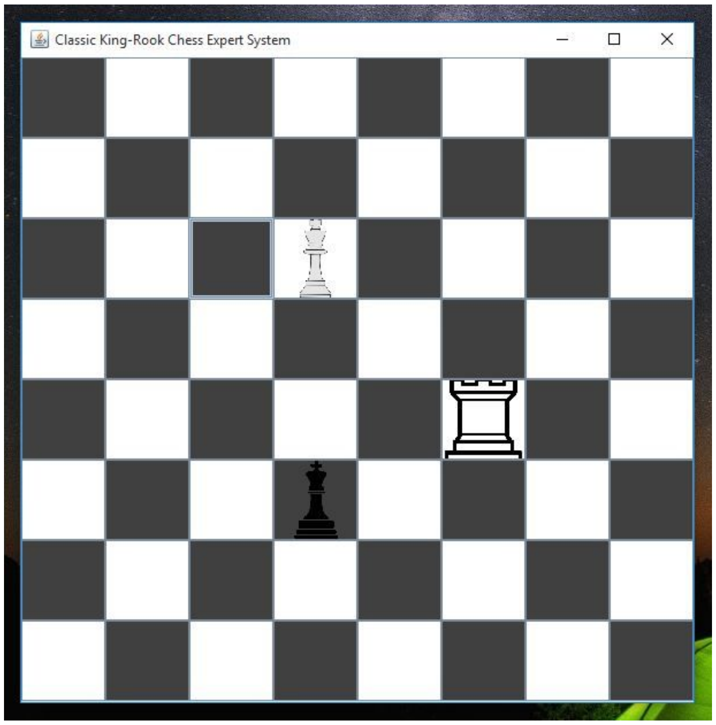
The aim of this project is to design a system for implementing Classic King and Rook Chess Endgame problem.

● It will use CLIPS for predefining rule based system which will be responsible for making sure that the game is played following the rules of chess only.

● The software in java uses CLIPSJNI which is a CLIPS library designed for java.

● It uses CLIPSJNI for getting the output for next move and software uses Java Swing for implementing the GUI .

INSTRUCTIONS

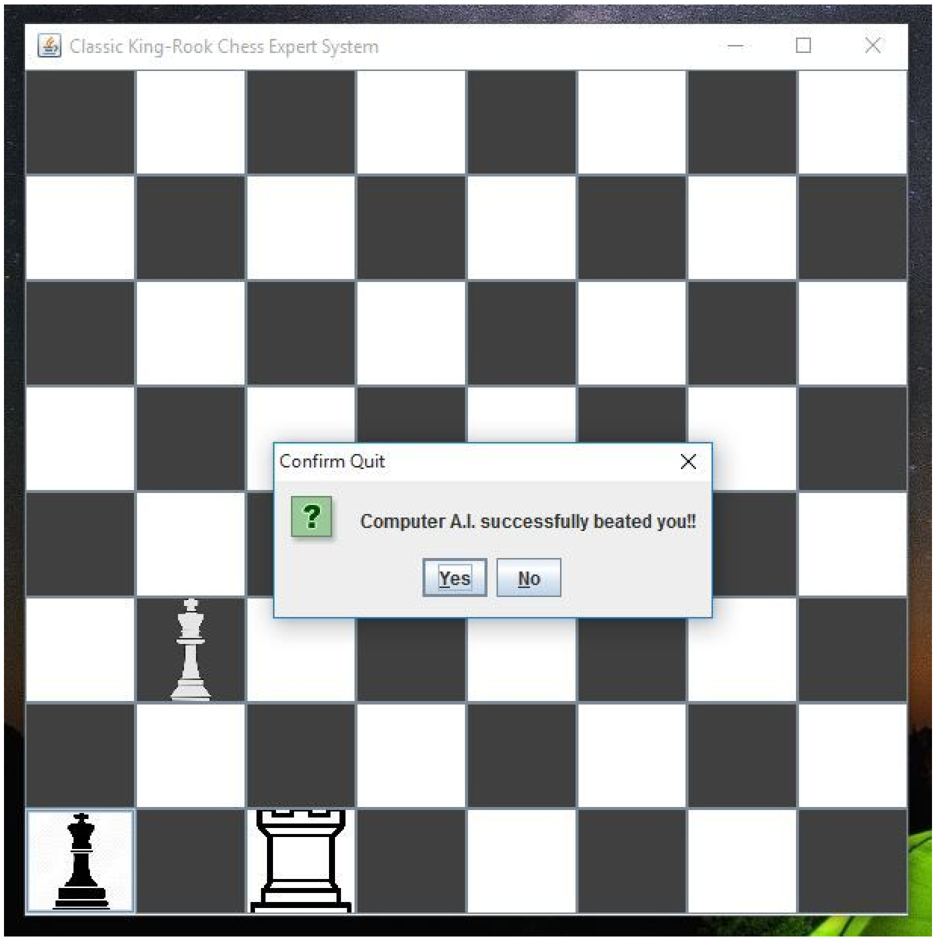


This screen is loaded as soon as the software is run.

1. Move starts from the opponent player and they place the first move (White has already made a move as soon as game starts but it always remains the same).

2. The opponent responds to the every move by us and try to avoid checkmate as long as   possible.

3. When the game finishes (Check-Mate) a dialogue box appears on the screen with the message that Computer AI successfully beat you.



Computer A.I. successfully beated you!!

SYSTEM FEATURES

The system has the following features –

- The game starts from the state when the opponent has a Rook and a King left on the grid with a King left of the player.
- The black player has the first move.
- The initial position of pieces can be decided by changing the value of initial position stored in file 1.txt, 2 .txt, 3.txt, 4.txt, 5.txt and 6.txt.
- Next move by the software is given as soon as we make our move.
- When the system beats you a dialogue box appears stating the ending of the game.
- When the game finishes , to start new game we have to run the software again.

TOOLS AND LANGUAGES USED

The Chess system uses Netbeans 8.02 as a programming tool. The software contains an in-built set of facts and rules which decides how the next moves will be decides by the software. It also makes sure that the moves should be valid.

The operating system requirement of the software is very simple. Any version of windows operating system can be used as the operating environment for this software. Ubuntu can also be used with a installation of NetBeans. It is a Java application which requires CLIPSJNI(CLIPS library in Java). The application is not resource or graphics intensive, so, there are no practical hardware constraints.

Following are the softwares required for the correct functioning of the system:

- The software designed will work correctly with the Windows operating system .
- Java version used is Java JDK 8.0
- Netbeans IDE 8.0.2 is used for writing the code.
- The input during the gameplay is by using mouse.

RESULTS AND CONCLUSIONS

Successfully implemented the game with several additional features apart from the basic functionalities mentioned in features. The graphical user interface of **Classic King and Rook Chess Endgame** is to be designed with usability as the first priority. The system will be presented and organized in a manner that is both visually appealing and easy for the user to use. The system is flexible enough to incorporate the changes leading to improvement in the working of the software.

RULES FOR EXPERT SYSTEM

1. IF (initial-fact) AND user has called function (start-game) THEN IF black king has legal coordinates as user input THEN status "white move" ELSE reset game.

2. IF using current co-ordinates of pieces checkmate can be done THEN game resets ELSE status "black move".

3. IF back king has only 1 move to play THEN move black king's only move AND status is "black move done".

4. IF a move is done AND piece moved is black THEN modify coordinates of black king.

5. IF status "black move" AND black king has more than 1 move to play THEN ask for user input AND status "black move done".

6. IF conditions for mating move (black king is in corner and there is king opposition) THEN check if such mating move exists IF yes THEN status "white move done" ELSE status "white move ponder".

7. IF conditions for mating move (black king is on edge and there is full king opposition) THEN search for mate IF yes THEN status "white move done" ELSE status "white move ponder".

8. IF no mating conditions THEN think of other continuations AND status "white move ponder".

9. IF king opposition AND rook in between AND status "white move ponder" THEN execute waiting move with white king AND status "white move done".

10. IF king opposition, white king should not leave it AND status "white move ponder" THEN play best rook move AND status "white move done".

11. IF white king is on edge AND under nasty opposition but has a move to leave edge AND status "white move ponder" THEN move white king AND status "white move done".

12. IF rook is attacked AND white king is away AND status "white move ponder" THEN play best rook move AND status "white move done".

13. IF rook is attacked AND white king can defend it AND status "white move ponder" THEN play king move AND status "white move done".

14. IF none of the other pondering rules is applicable AND status "white move ponder"

THEN choose a move among king AND rook best moves AND status "white move done".

APPENDIX

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package expertchess;

import CLIPSJNI.*;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.util.Scanner;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import static java.lang.Math.abs;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.*;
/**
 *
 * @author satyam, Rajat, Naveen, Lakshay, Abhinav
 */
public class Expertchess {

    /**
     * @param args the command line arguments
     */
    public static int x1,y1,x2,y2;
    public static String content;
    public static boolean first = true,flag = false;
    public static JFrame f;
    public static JButton[][] b = new JButton[8][8];
    public static ImageIcon whiteking = new ImageIcon(new
ImageIcon("whiteking.png").getImage().getScaledInstance(75, 75, Image.SCALE_DEFAULT));
    public static ImageIcon whiterook = new ImageIcon(new
ImageIcon("whiterook.png").getImage().getScaledInstance(75, 75, Image.SCALE_DEFAULT));
    public static ImageIcon blackking = new ImageIcon(new
ImageIcon("blackking.png").getImage().getScaledInstance(75, 75, Image.SCALE_DEFAULT));

    public static void main(String[] args) throws IOException
    {
        // TODO code application logic here

        f=new JFrame("Classic King-Rook Chess Expert System");
        f.setSize(600,600);
        f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        GridLayout gl = new GridLayout(8,8);
        f.setLayout(gl);
        for(int i=0; i<8; i++)
        {
            for(int j=0; j< 8; j++)
            {
                b[i][j] = new JButton();
                b[i][j].setBounds(600*i/8,600*j/8,600/8,600/8);
                if((i+j)%2==0)
                {
                    b[i][j].setBackground(Color.darkGray);
                }
                else
                {
                    b[i][j].setBackground(Color.white);
                }
                b[i][j].addActionListener(new ActionListener()
                {
                    @Override
                    public void actionPerformed(ActionEvent e)
                    {
                       JButton jb = (JButton) e.getSource();
                       for(int i=0;i<8;i++)
                        {
                             for(int j=0;j<8;j++)
                             {
                                 if(b[i][j] == jb)
                                 {
                                     x2 = i;
                                     y2 = j;
                                 }
                             }
                         }
                       String prevy = "", prevx = "";
                       String rookx = "", rooky = "";
                       String kingx = "", kingy = "";
                        try {
                            prevy = new String(Files.readAllBytes(Paths.get("5.txt")));
                            prevx = new String(Files.readAllBytes(Paths.get("6.txt")));
                            kingy = new String(Files.readAllBytes(Paths.get("1.txt")));
                            kingx = new String(Files.readAllBytes(Paths.get("2.txt")));
                            rooky = new String(Files.readAllBytes(Paths.get("3.txt")));
                            rookx = new String(Files.readAllBytes(Paths.get("4.txt")));
                        } catch (IOException ex) {
                            Logger.getLogger(Expertchess.class.getName()).log(Level.SEVERE,
null, ex);
                        }

                        if(((abs(Integer.parseInt(prevx)-1 - x2) <=1
)&&(abs(Integer.parseInt(prevy)-1 - y2) <= 1))&&
                                ((abs(Integer.parseInt(prevx)-1 - x2) !=
0)||(abs(Integer.parseInt(prevy)-1 - y2) != 0))
                               &&((abs(Integer.parseInt(kingx)-1 - x2) >1
)||(abs(Integer.parseInt(kingy)-1 - y2) >1))
                                &&((abs(Integer.parseInt(rookx)-1 - x2) !=
0)&&(abs(Integer.parseInt(rooky)-1 - y2) != 0)) ){
                            move(Integer.parseInt(prevx)-1,Integer.parseInt(prevy)-1,x2,y2);
                            try {
                               //System.out.println("Enter black king vertical : ");
                               //bky = in.nextLine();

Files.write(Paths.get("5.txt"),Integer.toString(y2+1).getBytes(), StandardOpenOption.WRITE);
                           } catch (IOException ex) {
                               Logger.getLogger(Expertchess.class.getName()).log(Level.SEVERE,
null, ex);
                           }
                           try {
                               // System.out.println("Enter black king horizontal : ");
                               // bkx = in.nextLine();
                               Files.write(Paths.get("6.txt"),
Integer.toString(x2+1).getBytes(), StandardOpenOption.WRITE);
                           } catch (IOException ex) {
                               Logger.getLogger(Expertchess.class.getName()).log(Level.SEVERE,
null, ex);
                           }
                            try {

                                runclip();
                                if(!issafe()){
                                    System.out.println("Winner");
                                    //JOptionPane.showMessageDialog(new JFrame("Game Over"),
"Computer A.I. successfully beated you!!");
                                    int result = JOptionPane.showConfirmDialog(new
JFrame("Game Over"),
                                         "Computer A.I. successfully beated you!!",
                                         "Confirm Quit", JOptionPane.YES_NO_OPTION);
                                    if (result == JOptionPane.YES_OPTION) System.exit(0);
                                }
                                System.out.println("safe");


                           } catch (IOException ex) {
                               Logger.getLogger(Expertchess.class.getName()).log(Level.SEVERE,
null, ex);
                           }
                        }else{
                            x2 = Integer.parseInt(prevx);
                            y2 = Integer.parseInt(prevy);
                        }
                    }

                    private boolean issafe() {
                       String prevy = "", prevx = "";
                       String rookx = "", rooky = "";
                       String kingx = "", kingy = "";
                        try {
                            prevy = new String(Files.readAllBytes(Paths.get("5.txt")));
                            prevx = new String(Files.readAllBytes(Paths.get("6.txt")));
                            kingy = new String(Files.readAllBytes(Paths.get("1.txt")));
                            kingx = new String(Files.readAllBytes(Paths.get("2.txt")));
                            rooky = new String(Files.readAllBytes(Paths.get("3.txt")));
                            rookx = new String(Files.readAllBytes(Paths.get("4.txt")));
                        } catch (IOException ex) {
                            Logger.getLogger(Expertchess.class.getName()).log(Level.SEVERE,
null, ex);
                        }
                        System.out.println(prevx + " " + prevy + " " + kingx + " " + kingy + "
" + rookx + " " + rooky + " " );
                        int movex[] = {0, 1, 1, 1,  0, -1, -1, -1};
                        int movey[] = {1, 1, 0, -1,-1, -1,  0,  1};

                        for(int i=0;i<8;i++){
                            if(((Integer.parseInt(prevx) -1 + movex[i]) >=
0)&&((Integer.parseInt(prevy) -1 + movey[i]) >= 0)&&
                                    ((Integer.parseInt(prevx) -1 + movex[i]) <
8)&&((Integer.parseInt(prevy) -1 + movey[i]) < 8)){
                                int x2 = (Integer.parseInt(prevx) -1 + movex[i]);
                                int y2 = (Integer.parseInt(prevy) -1 + movey[i]);
                               if( (abs(Integer.parseInt(kingx)-1 - x2) > 1 ) ||
(abs(Integer.parseInt(kingy)-1 - y2) > 1)){
                                 if((abs(Integer.parseInt(rookx)-1 - x2) != 0) &&
(abs(Integer.parseInt(rooky)-1 - y2) != 0)){
                                   System.out.println( Integer.toString(x2) +" "+
Integer.toString(y2));
                                   return true;
                                 }
                               }
                            }
                        }
                        return false;
                    }
                });
                f.add(b[i][j]);

            }
        }
        b[0][4].setIcon(whiteking);
        b[7][4].setIcon(blackking);
        b[0][0].setIcon(whiterook);

        f.setVisible(true);//making the frame visible

        System.out.println("Game Started.......\n");
        //System.out.println("Enter white king vertical : ");
        String wky = "5";
        Files.write(Paths.get("1.txt"), wky.getBytes(), StandardOpenOption.WRITE);
        //System.out.println("Enter white king horizontal : ");
        String wkx = "1";
        Files.write(Paths.get("2.txt"), wkx.getBytes(), StandardOpenOption.WRITE);

        //System.out.println("Enter white rook vertical : ");
        String wry = "1";
        Files.write(Paths.get("3.txt"), wry.getBytes(), StandardOpenOption.WRITE);
        //System.out.println("Enter white rook horizontal : ");
        String wrx = "1";
        Files.write(Paths.get("4.txt"), wrx.getBytes(), StandardOpenOption.WRITE);


        //System.out.println("Enter black king vertical : ");
        String bky = "5";
        Files.write(Paths.get("5.txt"), bky.getBytes(), StandardOpenOption.WRITE);
        //System.out.println("Enter black king horizontal : ");
        String bkx = "8";
        Files.write(Paths.get("6.txt"), bkx.getBytes(), StandardOpenOption.WRITE);

        runclip();
    }
    static void move(int x1, int y1, int x2, int y2)
    {
        ImageIcon icon = (ImageIcon) b[x1][y1].getIcon();
        b[x2][y2].setIcon(icon);
        b[x1][y1].setIcon(null);
    }

    static void runclip() throws IOException{
        Environment clips = new Environment();
        clips = new Environment();
        clips.load("chess.clp");
        clips.reset();
        clips.run();
        clips.eval("(start-game)");
        clips.destroy();
        content = new String(Files.readAllBytes(Paths.get("output.txt")));
        //System.out.println(content);

       if(content.contains("K")){
            int ypos = content.charAt(3) - 'a' + 1;
            int xpos = content.charAt(4) - '0';
            String vert = Integer.toString(ypos);
            String hor = Integer.toString(xpos);
            String prevy = new String(Files.readAllBytes(Paths.get("1.txt")));
```

```
            String prevx = new String(Files.readAllBytes(Paths.get("2.txt")));
            move(Integer.parseInt(prevx)-1,Integer.parseInt(prevy)-1,xpos - 1,ypos - 1);
            Files.write(Paths.get("1.txt"), vert.getBytes(), StandardOpenOption.WRITE);
            Files.write(Paths.get("2.txt"), hor.getBytes(), StandardOpenOption.WRITE);
        }
        else{
            int ypos = content.charAt(3) - 'a' + 1;
            int xpos = content.charAt(4) - '0';
            String vert = Integer.toString(ypos);
            String hor = Integer.toString(xpos);
            String prevy = new String(Files.readAllBytes(Paths.get("3.txt")));
            String prevx = new String(Files.readAllBytes(Paths.get("4.txt")));
            move(Integer.parseInt(prevx)-1,Integer.parseInt(prevy)-1,xpos - 1,ypos - 1);
            Files.write(Paths.get("3.txt"), vert.getBytes(), StandardOpenOption.WRITE);
            Files.write(Paths.get("4.txt"), hor.getBytes(), StandardOpenOption.WRITE);
        }
    }
}
```