INFORMATION SEARCH AND ANALYSIS SKILLS (ISAS)

CEP-CCIT

Continuing Education Program
Center for Computing and Information Technology
Fakultas Teknik Universitas Indonesia

"Minesweeper Game in Java"

Written by;

Ade Kurniawan	(2120010004)
Angga Pranidiya S	(2120010231)
Rafi Fajar Sulaiman	(2120010269)

Faculty;

M. Riza Iqbal Latief, ST.

Class;

2 SE1

PROJECT ON

Minesweeper Game in java

Developed by

- **Ade Kurniawan**
- **Rafi Fajar Sulaiman**
- **Angga Pranindiya**

DATA ANALYSIS

Batch Code	: 2SE1
Start Date	: April 15th, 2022
End Date	: April 18th, 2022

Name Of Faculty : Mr. M. Riza Iqbal

Names of Developer :

- Ade Kurniawan
- Rafi Fajar Sulaiman
- Angga Pranindiya

NIIT

CERTIFICATE

This is to certify that report titled “Minesweeper game in java”. Embodies the original work done by Ade Kurniawan, Rafi Fajar Sulaiman, Angga Pranindiya. Project in partial fulfillment of their course requirement at NIIT.

Coordinator :

Mr. M. Riza Iqbal

ACKNOWLEDGEMENT

Thank you, the writer wishes to Allah SWT, the Almighty God for His blessings and mercy, that we were able to complete this ISAS task, both in the form of paper and in the form of presentations in a timely manner.

Thank you especially for our faculty, M. Riza Iqbal Latief, ST. who always help us to finish this paper. Thank you to the all fellow students who have supported and also thank you for being a partner in this education at CCIT-FTUI.

The Project Paper entitled “**Minesweeper game in Java**” which the authors propose as Project 2022 task requirements. Author hopes this paper can be useful for the readers, so that it can add knowledge and insight for the readers.

The authors realize that this paper is far from perfect. Therefore, the authors really all constructive suggestions and criticisms from the reader to improve this paper. Finally, hopefully this paper can provide many benefits for the readers.

Depok, April 2022

SYSTEM ANALYSIS

System summary :

Minesweeper is a computer game for one player. The object of the game is to clear the playing field without hitting mines. The game is played by opening squares on the grid, usually by clicking on the mouse. If the opened box contains mines, the player loses. A square containing a number indicates the number of mines around it. Players can right click to mark the box with mines. in this game the first step in starting the game is to choose a grid size, after that in the next step we are asked to choose the level of difficulty that exists such as easy, medium, and hard

CLASS DIAGRAM

```
package minesweeper.minesweeper;

import javax.swing.JOptionPane;

public class Minesweeper extends javax.swing.JFrame {

    public void start(Minesweeper minesweeper) {
        Input = new input(minesweeper);
        Input.main(Input);
    }

    public void proceed(int size) {
        int toughness = 1;
        Object[] options = {"Easy", "Moderate", "Hard"};
        toughness = JOptionPane.showOptionDialog(null,
                "What's your difficulty level ?", "Difficulty",
                JOptionPane.YES_NO_CANCEL_OPTION,
                JOptionPane.QUESTION_MESSAGE,
                null,
                options,
                options[1]);
        if(toughness == -1)
            System.exit(0);
        newGame = new game(size, toughness);
        newGame.main(newGame, size);
    }
```

CLASS DIAGRAM

```
public static void restart() {
        int toughness = 1;
        Object[] options = {"Play Again", "Exit"};
        toughness = JOptionPane.showOptionDialog(null,
                    "Do you want to play again ?", "New Game",
                    JOptionPane.YES_NO_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE,null,
                    options, options[0]);
        if(toughness == 1) {
            System.exit(0);
        }
        minesweeper = new Minesweeper();
        minesweeper.start(minesweeper);
    }

    public static void main(String[] args) {
        minesweeper = new Minesweeper();
        minesweeper.start(minesweeper);
    }

    private static Minesweeper minesweeper;
    private static game newGame;
    private static input Input;
}
```

CLASS DIAGRAM

```
  package minesweeper;

import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.util.Random;
import java.awt.event.*;
import javax.imageio.ImageIO;

public class game extends JFrame {

    public game(int size, int toughness) {
        noOfMines = size*(1 + toughness/2);
        this.setSize(size*MAGIC_SIZE, size*MAGIC_SIZE + 50);
        this.setTitle("Minesweeper");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    }

    private void setMines(int size) {
        Random rand = new Random();
```

CLASS DIAGRAM

```
  mineLand = new int[size][size];
      for (int i = 0; i < size; i++) {
          for (int j = 0; j < size; j++) {
              mineLand[i][j] = 0;
          }
      }

      int count = 0;
      int xPoint;
      int yPoint;
      while(count<noOfMines) {
          xPoint = rand.nextInt(size);
          yPoint = rand.nextInt(size);
          if (mineLand[xPoint][yPoint]!=-1) {
              mineLand[xPoint][yPoint]=-1;  // -1 represents bomb
              count++;
          }
      }
// Fill boxes adjacent to mines with numbers
      for (int i = 0; i < size; i++) {
      for (int j = 0; j < size; j++) {
          if (mineLand[i][j]==-1) {
                  for (int k = -1; k <= 1 ; k++) {
                  for (int l = -1; l <= 1; l++) {
```

CLASS DIAGRAM

```
// In boundary cases
                 try {
                    if (mineLand[i+k][j+l]!=-1) {
                        mineLand[i+k][j+l] += 1;
                    }
                 }
                 catch (Exception e) {
                    // Do nothing
                 }
              }
              }
        }
     }
     }
  }

  public void main(game frame, int size) {

     // Some instantiation
     GameEngine gameEngine = new GameEngine(frame);
     MyMouseListener myMouseListener = new MyMouseListener(frame);
     JPanel mainPanel = new JPanel();

     panel1 = new JPanel();
      panel2 = new JPanel();
```

CLASS DIAGRAM

```
this.noOfRevealed = 0;
        revealed = new boolean[size][size];
        flagged = new boolean[size][size];
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                revealed[i][j] = false;
                flagged[i][j] = false;
            }
        }
        // Images
        try {
            smiley = ImageIO.read(getClass().getResource("images/Smiley.png"));
            newSmiley = smiley.getScaledInstance(MAGIC_SIZE, MAGIC_SIZE,
java.awt.Image.SCALE_SMOOTH);
            dead = ImageIO.read(getClass().getResource("images/dead.png"));
            newDead = dead.getScaledInstance(MAGIC_SIZE, MAGIC_SIZE,
java.awt.Image.SCALE_SMOOTH);
            flag = ImageIO.read(getClass().getResource("images/flag.png"));
            newFlag = flag.getScaledInstance(MAGIC_SIZE, MAGIC_SIZE,
java.awt.Image.SCALE_SMOOTH);
            mine = ImageIO.read(getClass().getResource("images/mine.png"));
            newMine = mine.getScaledInstance(20, 20, java.awt.Image.SCALE_SMOOTH);
         }
         catch (Exception e){
         }
```

CLASS DIAGRAM

```
mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.Y_AXIS));

    BoxLayout g1 = new BoxLayout(panel1, BoxLayout.X_AXIS);
    panel1.setLayout(g1);

    JLabel jLabel1 = new JLabel(" Flags = ");
    jLabel1.setAlignmentX(Component.LEFT_ALIGNMENT);
    jLabel1.setHorizontalAlignment(JLabel.LEFT);
    flagsLabel = new JLabel(""+this.noOfMines);

    smiling = true;
    smileButton = new JButton(new ImageIcon(newSmiley));
    smileButton.setPreferredSize(new Dimension(MAGIC_SIZE, MAGIC_SIZE));
    smileButton.setMaximumSize(new Dimension(MAGIC_SIZE, MAGIC_SIZE));
    smileButton.setBorderPainted(true);
    smileButton.setName("smileButton");
    smileButton.addActionListener(gameEngine);

    JLabel jLabel2 = new JLabel(" Time :");
    timeLabel = new JLabel("0");
    timeLabel.setAlignmentX(Component.RIGHT_ALIGNMENT);
    timeLabel.setHorizontalAlignment(JLabel.RIGHT);
```

CLASS DIAGRAM

```
panel1.add(jLabel1);
    panel1.add(flagsLabel);
    panel1.add(Box.createRigidArea(new Dimension((size-1)*15 - 80,50)));
    panel1.add(smileButton, BorderLayout.PAGE_START);
    panel1.add(Box.createRigidArea(new Dimension((size-1)*15 - 85,50)));
    panel1.add(jLabel2);
    panel1.add(timeLabel);

    GridLayout g2 = new GridLayout(size, size);
    panel2.setLayout(g2);

    buttons = new JButton[size][size];

    for (int i=0; i<size; i++) {
       for (int j=0; j<size ; j++ ) {
          buttons[i][j] = new JButton();
          buttons[i][j].setPreferredSize(new Dimension(12, 12));
          buttons[i][j].setBorder(new LineBorder(Color.BLACK));
          buttons[i][j].setBorderPainted(true);
          buttons[i][j].setName(i + " " + j);
          buttons[i][j].addActionListener(gameEngine);
          buttons[i][j].addMouseListener(myMouseListener);
          panel2.add(buttons[i][j]);
        }
     }
```

CLASS DIAGRAM

```
 // Both panels done

    mainPanel.add(panel1);
    mainPanel.add(panel2);
    frame.setContentPane(mainPanel);
    this.setVisible(true);

    // Algorithms
    setMines(size);

    // The timer
    timeThread timer = new timeThread(this);
    timer.start();

}

// Increase timer every second
public void timer() {
    String[] time = this.timeLabel.getText().split(" ");
    int time0 = Integer.parseInt(time[0]);
    ++time0;
    this.timeLabel.setText(Integer.toString(time0) + " s");
}
```

CLASS DIAGRAM

```
    // Change icon upon clicking smile Button
    public void changeSmile() {
        if (smiling) {
            smiling=false;
            smileButton.setIcon(new ImageIcon(newDead));
        } else {
            smiling=true;
            smileButton.setIcon(new ImageIcon(newSmiley));
        }
    }
    // If a block is right clicked
public void buttonClicked(int x, int y) {
        if(!revealed[x][y] && !flagged[x][y]) {
            revealed[x][y] = true;

            switch (mineLand[x][y]) {
                case -1:
                    try {
                        buttons[x][y].setIcon(new ImageIcon(newMine));
                    } catch (Exception e1) {
                    }
                    buttons[x][y].setBackground(Color.RED);
                    try {
                        smileButton.setIcon(new ImageIcon(newDead));
                    } catch (Exception e2) {
                    }

                    JOptionPane.showMessageDialog(this, "Game Over !", null,
                            JOptionPane.ERROR_MESSAGE);    }
```

CLASS DIAGRAM

```
dispose();
            Minesweeper.restart();

            break;

         case 0:
            buttons[x][y].setBackground(Color.lightGray);
            ++this.noOfRevealed;

            if (gameWon()) {
               JOptionPane.showMessageDialog(rootPane,
                  "Congratulations! You've Won");

               dispose();
               Minesweeper.restart();
            } // Winning condition

            // Else simply recurse around
            for (int i = -1; i <= 1; i++) {
            for (int j = -1; j <= 1; j++) {
               try {
                  buttonClicked(x + i, y + j);
               }
               catch (Exception e3) {
                  // Do nothing
               }
            }
            }
```

CLASS DIAGRAM

```
case 0:
            buttons[x][y].setBackground(Color.lightGray);
            ++this.noOfRevealed;

            if (gameWon()) {
               JOptionPane.showMessageDialog(rootPane,
                  "Congratulations! You've Won");

               System.exit(0);
            } // Winning condition

            // Else simply recurse around
            for (int i = -1; i <= 1; i++) {
            for (int j = -1; j <= 1; j++) {
               try {
                  buttonClicked(x + i, y + j);
               }
               catch (Exception e3) {
                  // Do nothing
               }
            }
            }

            break;
```

CLASS DIAGRAM

```
break;


            default:
                buttons[x][y].setText(Integer.toString(mineLand[x][y]));
                buttons[x][y].setBackground(Color.LIGHT_GRAY);
                ++this.noOfRevealed;
                if (gameWon()) {
                    JOptionPane.showMessageDialog(rootPane, "You Won !");


                    dispose();
                    Minesweeper.restart();
                }


                break;
        }
    }


}
private JButton[][] buttons;  // The Grid buttons
private JPanel panel1;  // Top panel containing labels and a smile button
private JPanel panel2;  // Bottom panel containing the grid of buttons
private JLabel flagsLabel;  // Number of flags remaining to be used
private JButton smileButton;  // The smile button ;-)
private JLabel timeLabel;  // Label showing time elapsed
```

CLASS DIAGRAM

```
   private int noOfMines = 0;  // The no. of mines in the field
    private int[][] mineLand;  // 2-D array containing info for each block
    private boolean[][] revealed;  // Whether the button has been clicked
    private int noOfRevealed;  // How many of them?
    private boolean[][] flagged;  // Or the got flagged?

    private Image smiley;
    private Image newSmiley;
    private Image flag;
    private Image newFlag;
    private Image mine;
    private Image newMine;
    private Image dead;
    private Image newDead;

    private boolean smiling;  // Is he? Or is he not?

    public static final int MAGIC_SIZE = 30;

}

class GameEngine implements ActionListener {
    game parent;

    GameEngine(game parent) {
        this.parent = parent;
    }
```

CLASS DIAGRAM

```
    @Override
    public void actionPerformed(ActionEvent e) {
        Object eventSource = e.getSource();
        JButton clickedButton = (JButton) eventSource;
        String name = clickedButton.getName();
        if (name.equals("smileButton")) {
            parent.changeSmile();
        }
        else {
            String[] xy = clickedButton.getName().split(" ", 2);
            int x = Integer.parseInt(xy[0]);
            int y = Integer.parseInt(xy[1]);
            parent.buttonClicked(x, y);
        }
    }
}
class MyMouseListener implements MouseListener {
    game parent;
    MyMouseListener(game parent) {
        this.parent = parent;
    }
    public void mouseExited(MouseEvent arg0){
    }
    public void mouseEntered(MouseEvent arg0){
    }
    public void mousePressed(MouseEvent arg0){
    }
    public void mouseClicked(MouseEvent arg0){
    }
```

CLASS DIAGRAM

```
@Override
    public void mouseReleased(MouseEvent arg0) {
        if(SwingUtilities.isRightMouseButton(arg0)){
            Object eventSource = arg0.getSource();
            JButton clickedButton = (JButton) eventSource;
            String[] xy = clickedButton.getName().split(" ", 2);
            int x = Integer.parseInt(xy[0]);
            int y = Integer.parseInt(xy[1]);
            parent.buttonRightClicked(x, y);
        }
    }
}

class timeThread implements Runnable {
    private Thread t;
    private game newGame;

    timeThread(game newGame) {
        this.newGame = newGame;
    }
```

CLASS DIAGRAM

```
public void run() {
        while(true) {
            try {
                Thread.sleep(1000);
                newGame.timer();
            }
            catch (InterruptedException e) {
                System.exit(0);
            }
        }
    }

    public void start() {
        if (t==null) {
            t = new Thread(this);
            t.start();
        }
    }
}
```

CLASS DIAGRAM

```
package minesweeper;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class input extends JFrame {

    public input(Minesweeper minesweeper) {
        this.iMinesweeper = minesweeper;
        this.setSize(400, 100);
        this.setTitle("Input");
        setLocationRelativeTo(null);
        this.setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    }

    // Setter and Getter
    public void set(int n) {
        size = n;
        iMinesweeper.proceed(size);
    }
  public int get() {
        return size;
    }
```

CLASS DIAGRAM

```
public void main(input Input) {
        inputEngine = new InputEngine(Input);

        size=0;

        panel = new JPanel();

        label = new JLabel("Enter grid size : ");
        panel.add(label);

        text = new JTextField(30);
        text.addActionListener(inputEngine);
        panel.add(text);

        Input.setContentPane(panel);
        this.setVisible(true);
    }

    final private Minesweeper iMinesweeper;  // A reference to the original game
    private InputEngine inputEngine;  // The ActionListener

    private int size;  // size given
    private JPanel panel;
    private JLabel label;
    private JTextField text;
```

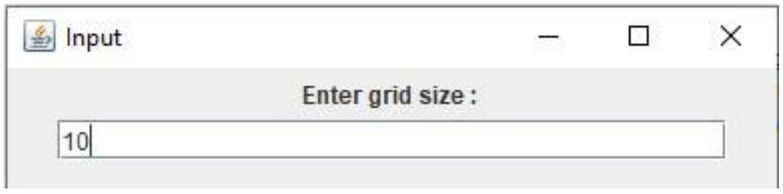
CLASS DIAGRAM

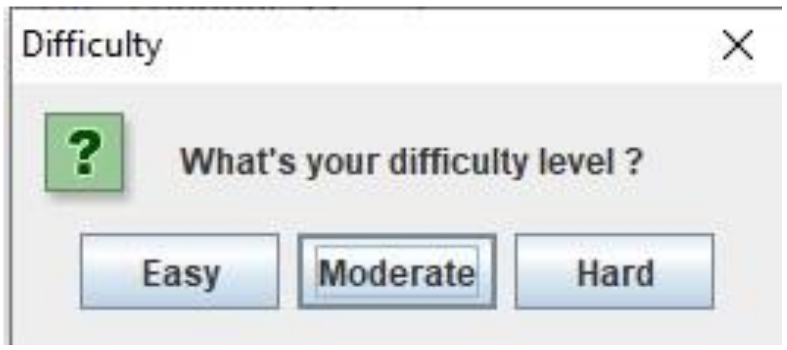
```
class InputEngine implements ActionListener {
   input parent;

   InputEngine(input parent) {
      this.parent = parent;
   }

   @Override
   public void actionPerformed(ActionEvent evt) {
      Object eventSource = evt.getSource();
      JTextField text = (JTextField) eventSource;
      String input =  "0";
      int size = 0;
      while(true) {
         try {
            input = text.getText();
            size = Integer.parseInt(input);
            if (size<=6) {
               JOptionPane.showMessageDialog(parent,
                     "Enter an integer greater than 6", "Invalid Input!",
                     JOptionPane.ERROR_MESSAGE);
               text.setText("");
               break;
              } else {
               parent.setVisible(false);
               parent.set(size);
            }
            break;
```

CLASS DIAGRAM

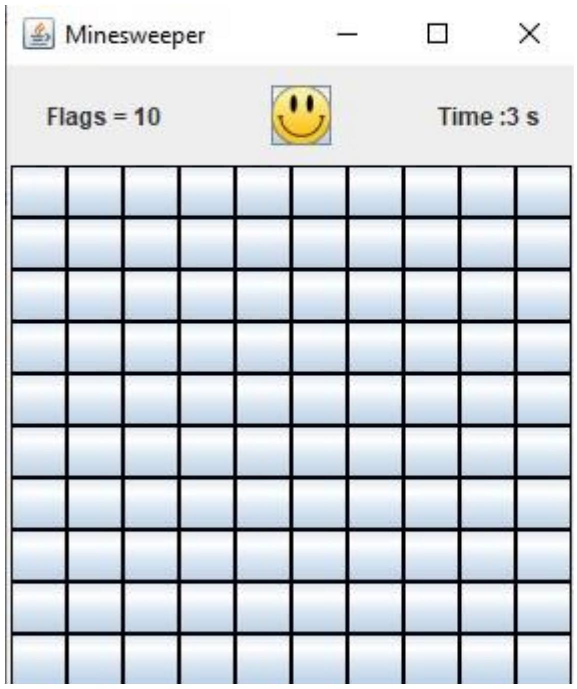
```
    }
        catch (NumberFormatException | HeadlessException e) {
          JOptionPane.showMessageDialog(parent,
               "Enter a valid integer!", "Invalid Input",
               JOptionPane.ERROR_MESSAGE);
          text.setText("");
          break;
        }
      }
    }
}
```

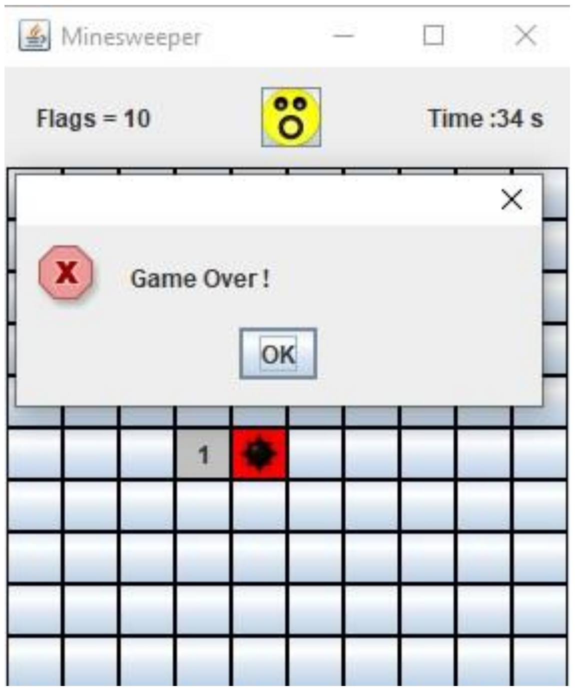
INTERFACE


Input
Enter grid size :
10

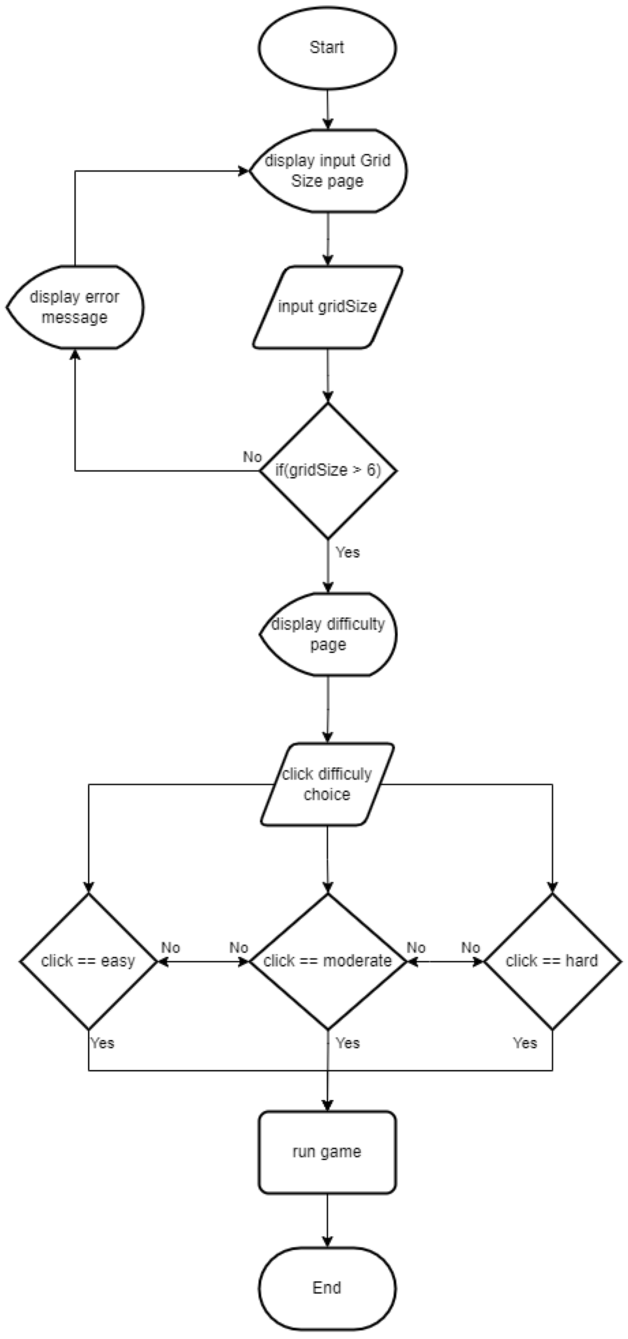

Difficulty
What's your difficulty level ?
Easy
Moderate
Hard

INTERFACE


Minesweeper
Flags = 10
Time :3 s


Minesweeper
Flags = 10
Time :34 s
Game Over !
OK
1

FLOWCHART



Start
display input Grid Size page
display error message
input gridSize
No
if(gridSize > 6)
Yes
display difficulty page
click difficuly choice
click == easy
No
No
click == moderate
No
No
click == hard
Yes
Yes
Yes
run game
End