OOM PROJECT

N SQUARE

GROUP MEMEBERS

Name	Roll Number
Challa Lahari	IIT2020083
Bhupathi Smaran	IIT2020168
Vaishnav Kamarti	IIT2020192
Nitheesh	IIT2020234

Abstract of the project

- We had to create a 9-box puzzle game Software
- A nine box puzzle game consists of nine boxes that are labeled from one to eight. Player needs to arrange the labeled boxes serially.
- The game counts the number of attempts by the player. A player with the minimum number of attempts is the winner of the game
- The puzzle game also accepts the name of the player and keeps track of the Players performance.

Our Approach to the PROJECT

- Home Screen will redirect the Player to desired -> **MainView** extends JPanel.
- Game board will have 8 buttons -> **Box** extends JButton.
- Box will be clicked , so to listening that event -> **EventHandler** extends action Listener (with Sound / Other Animation).
- We needed a panel to hold boxes and display swaps -> **PlayArea** extends JPanel.
- We needed a Brain for the Game, that will keep track of algorithm , it will initialise , it will validate moves (right or not) -> **GameCPU**
- We also had to track Player name and score, we stored it in an Arraylist -> **ScoreBoard** extends JPanel
- But each player data is unique -> **PlayerData**
- What if someone does not know how to play game -> **Rules** extends JPanel

What if we wants to extend the project

- We can increase the number of boxes to N^2 (16,25,36.....)

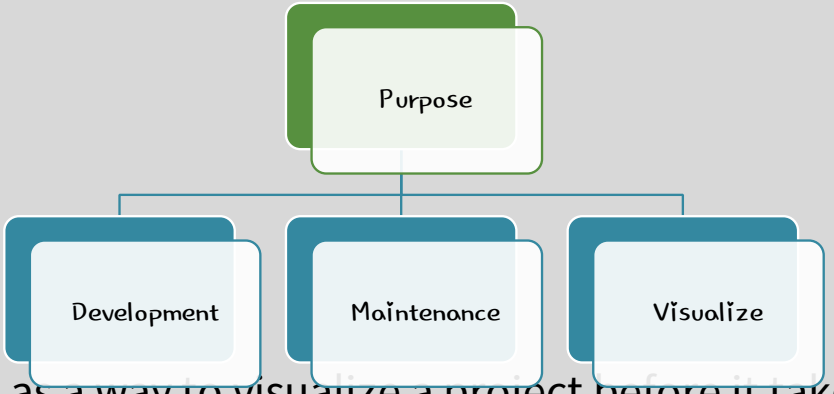
8	1	3
4		7
5	2	6

11	5	13	10
6	1		3
15	4	8	14
7	12	2	9

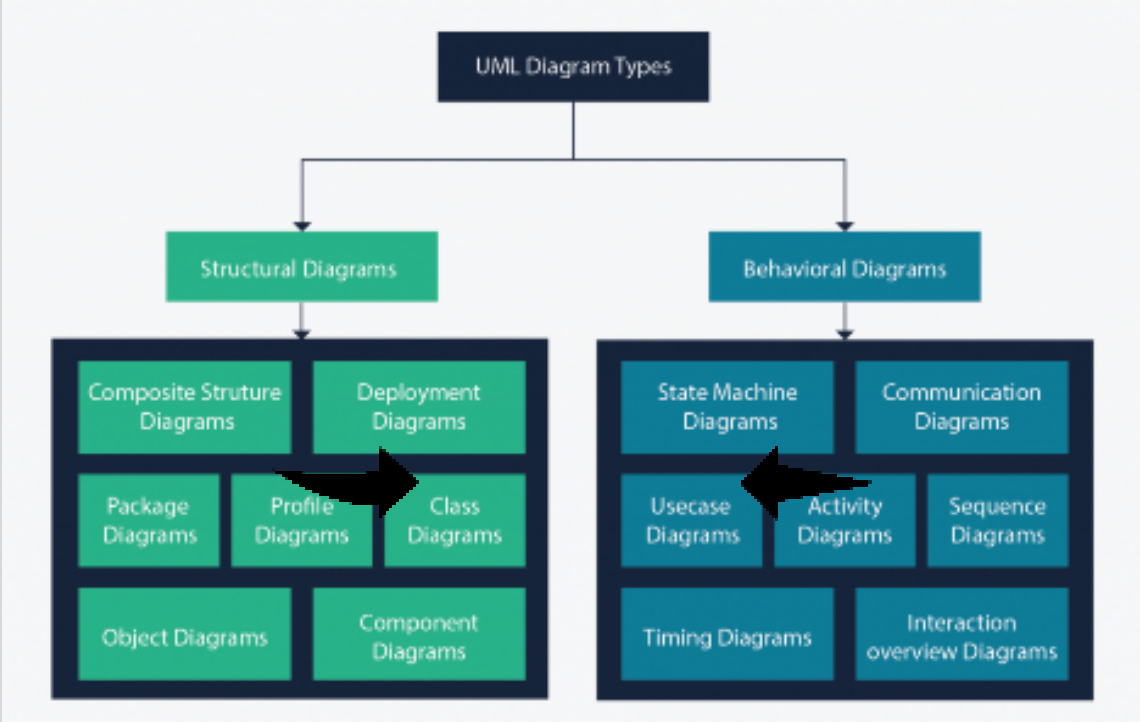
We can extend number puzzle to picture puzzle.

What is UML diagram?

It is based on **diagrammatic representations** of software components.



UML diagrams can be used as a way to visualize a project before it takes place or as documentation for a project afterward. But the overall goal of UML diagrams is to allow teams to visualize how a project is or will be working, and they can be used in any field, not just software engineering.



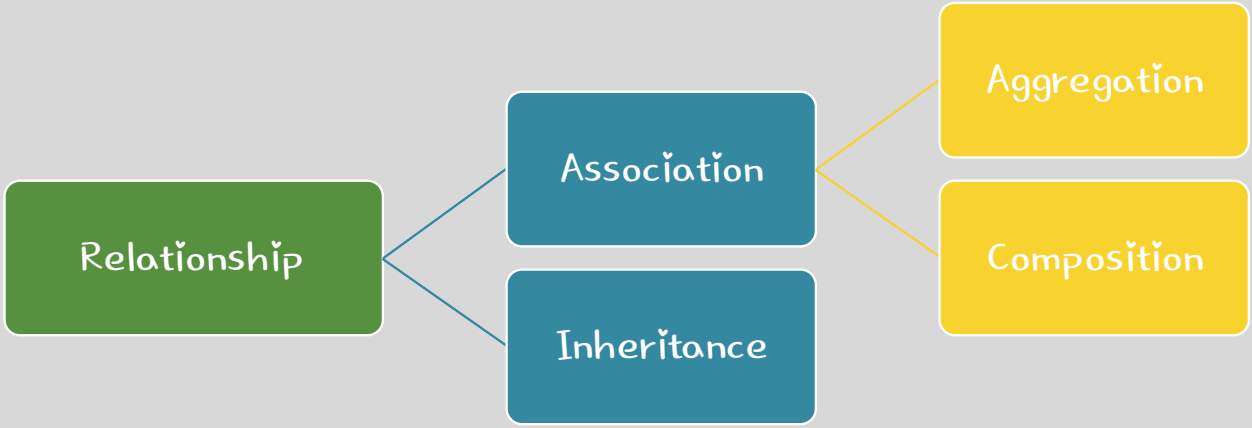
Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other.

Behavioral diagrams show what should happen in a system. They describe how the objects interact with each other to create a functioning system.

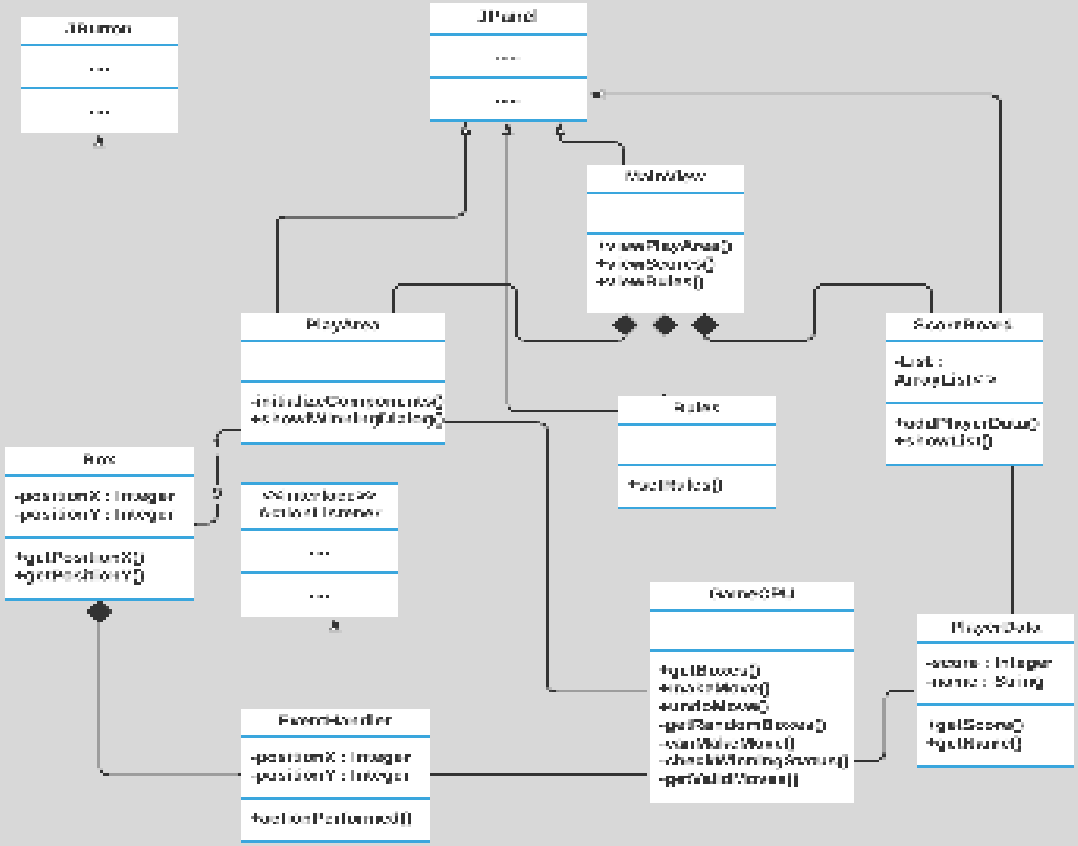
Class diagram

The class diagram is a central modeling technique that runs through nearly all object-oriented methods. This diagram describes the types of objects in the system and various kinds of static relationships which exist between them. Class diagram also shows the attributes and operations of a class

There are three principal kinds of relationships which are important :



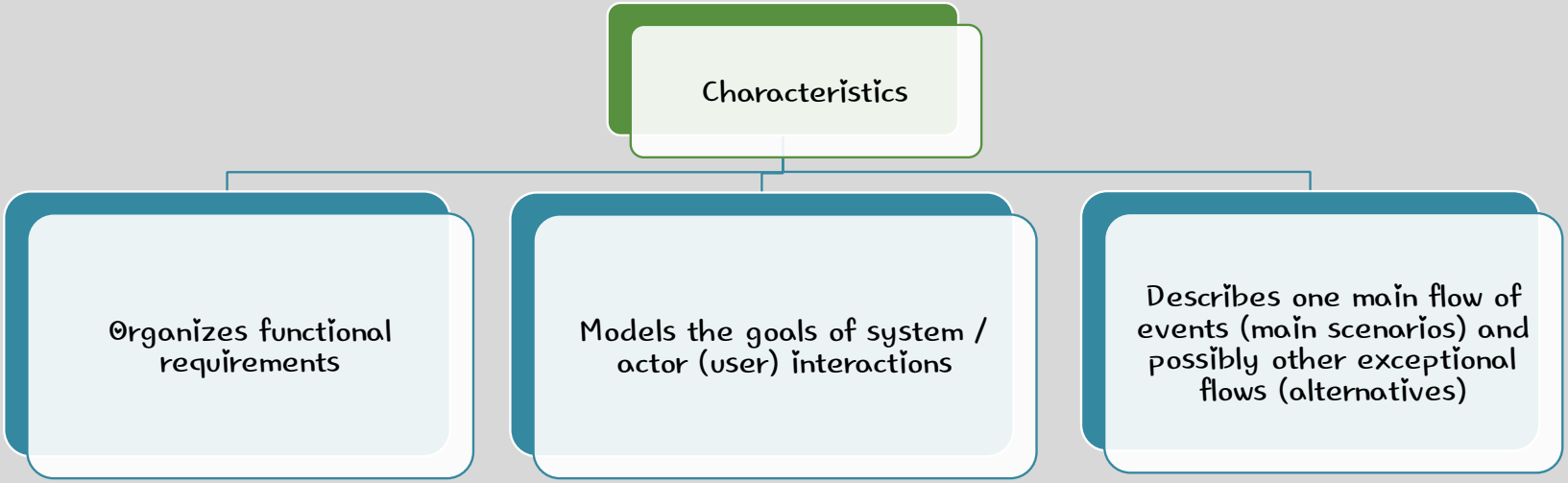
The Puzzle Box class diagram



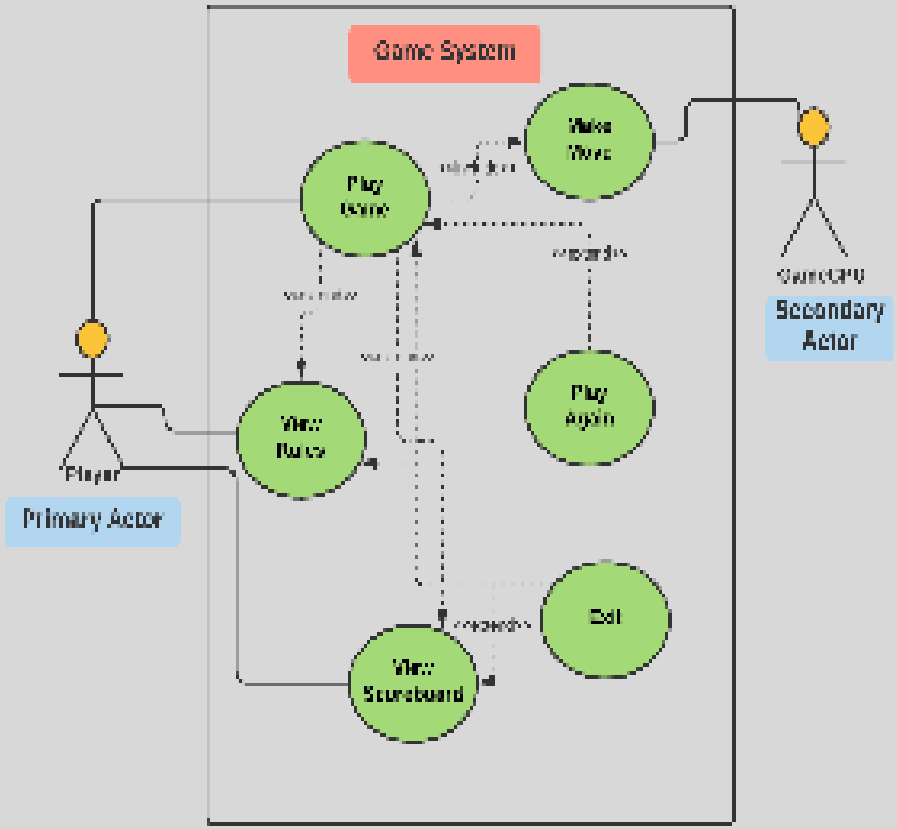
USE CASE diagram

- Use case diagram can summarize the details of your system's users and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent the following :

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve

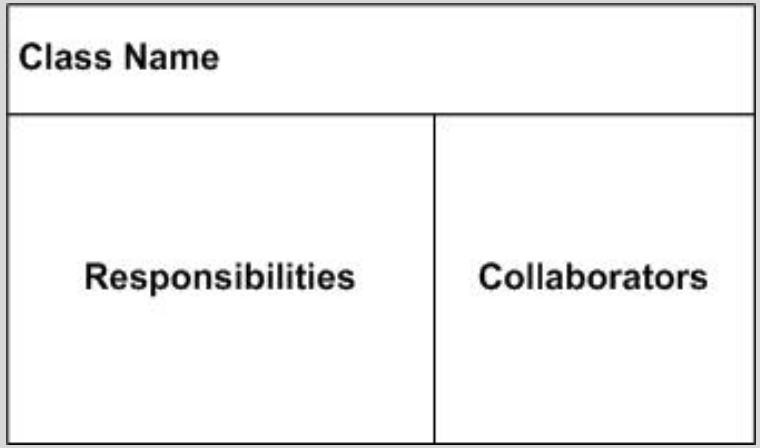


The Puzzle Box Use- Case diagram :



Class Responsibility Collaborator CRC DIAGRAM

- A Class Responsibility Collaborator (CRC) model is a collection of standard index cards that have been divided into three sections, A class represents a collection of similar objects, a responsibility is something that a class knows or does, and a collaborator is another class that a class interacts with to fulfill its responsibilities.



GameCPU

Responsibilities	Collaborators
Make moves Validate moves Generate Box Increase count on valid moves Check if win	Box

ScoreBoard

Responsibilities	Collaborators
Displays high scores of player for every level of difficulty Keep Track of Highscore	PlayerData PlayArea ScoreBoard Box Theme

Box

Responsibilities	Collaborators
Make Box	Theme

Rules

Responsibilities	Collaborators
Displays rules of the game	

SimpleAudioPlayer

Responsibilities	Collaborators
Plays audio music in the background	

Theme

Responsibilities	Collaborators
Changes the theme of the game (dark/light)	

PlayArea

Responsibilities	Collaborators
ShowCount Initialize GameCPU DrawBoxes Initialize EventHandler	GameCpu Theme AudioPlayer

Rules

Responsibilities	Collaborators
Displays rules of the game	NineBoxPuzzle ScoreBoard SimpleAudioPlayer

BoxPuzzle

Responsibilities	Collaborators
Start Game Show rules Show ScoreBoard	PlayArea Rules ScoreBoard Theme AudioPlayer

EventHandler

Responsibilities	Collaborators
Function call on catching event	PlayerData

MyBorder

Responsibilities	Collaborators
Class for Setting Borders of Button	Theme

PuzzleBox Class

JMenuBar

```
JMenuBar menubar = new JMenuBar();
```

JMenuOptions

```
JMenu options = new JMenu("Theme");
        JMenuItem darkTheme = new JMenuItem("Dark Theme");
        darkTheme.addActionListener(
            new ActionListener(){
                @Override
                public void actionPerformed(ActionEvent e)
                {
                    //Change Theme
                }});
```

GridBagLayout

```
setLayout(new GridBagLayout());//Setting layout
    GridBagConstraints gridBagLayout = new GridBagConstraints();// we have to do this to add it to container
    gridBagLayout.fill = GridBagConstraints.BOTH;
    gridBagLayout.anchor = GridBagConstraints.CENTER;
    gridBagLayout.ipady = 20;
    gridBagLayout.ipadx = 40;
    gridBagLayout.gridwidth = 1;
    gridBagLayout.gridheight = 1;
    gridBagLayout.insets = new Insets(10,10,10,10);
```

PuzzleBox Class

JButtons

```
JButton ruleButton = new JButton("rule");   //Rules or PlayGame or Scores or Exit
    ruleButton.setOpaque(false);
ruleButton.setContentAreaFilled(false);
ruleButton.setBorderPainted(false);
    JLabel l7 = new JLabel("");

    l7.setIcon(imageIconPath("Rules"+theme));      //Setting Theme
    l7.setPreferredSize(new Dimension(100, 100));
    ruleButton.add(l7);
    ruleButton.addActionListener(new ActionListener()
 {
      //Change Panel to Rules or PlayGame or Scores or Exit
    }});

   ruleButton.setFont(new Font("Dialog", Font.PLAIN, 40));
gridBagLayout.gridx = 0;
gridBagLayout.gridy = 2;
add(ruleButton, gridBagLayout);
```

PlayArea Class

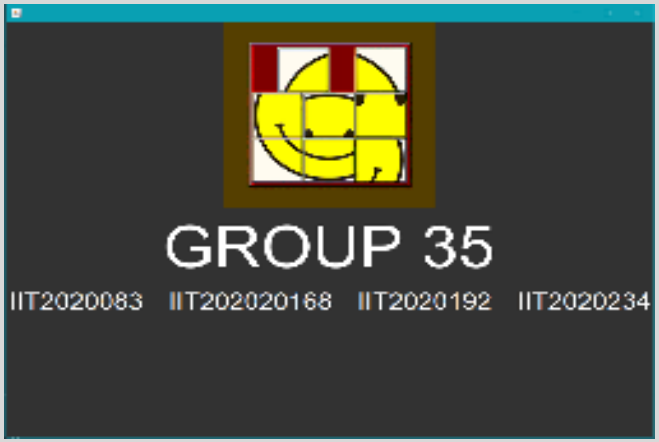
Initialise

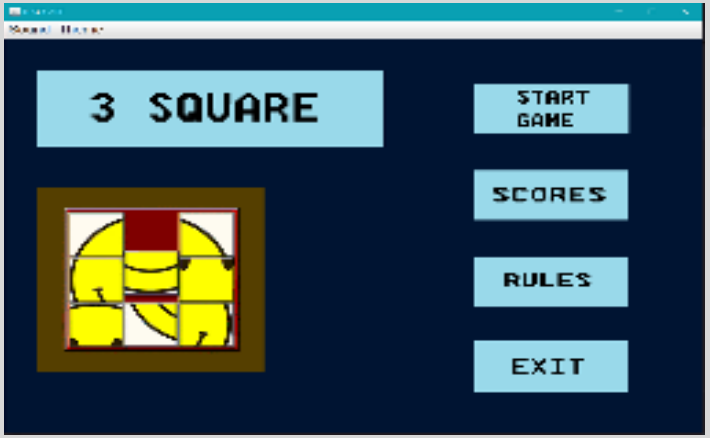
```
private void initializeComponents(int n) {
        setLayout(new GridLayout(n, n));
        Box[][] boxes = mygamecpu.getBoxes();

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++){
                add(boxes[i][j]);
                boxes[i][j].addActionListener(new EventHandler(mygamecpu, i, j));
            }
        }
        setVisible(true);
    }
```

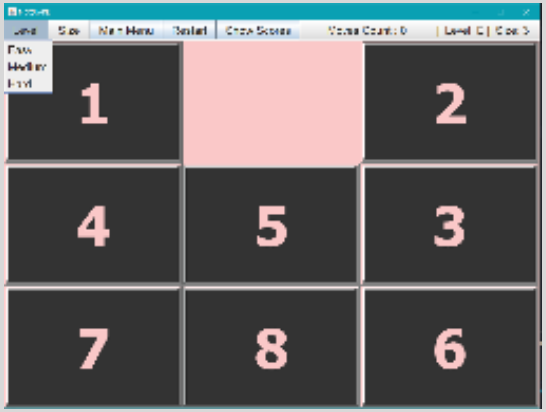
Moves Setter

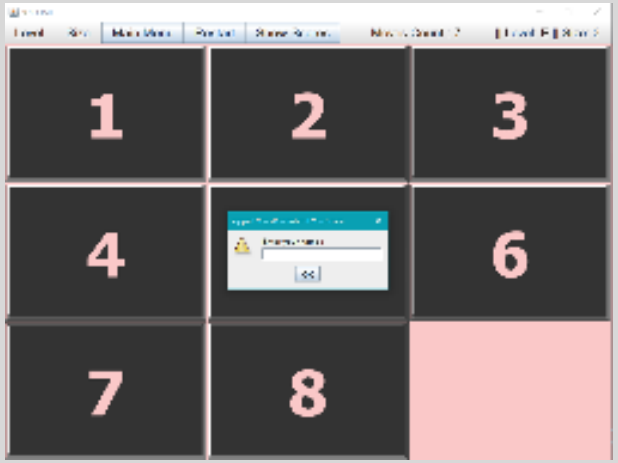
```
protected void setCount(int movesCount) {
        score.setText("Moves Count : " + movesCount);
```

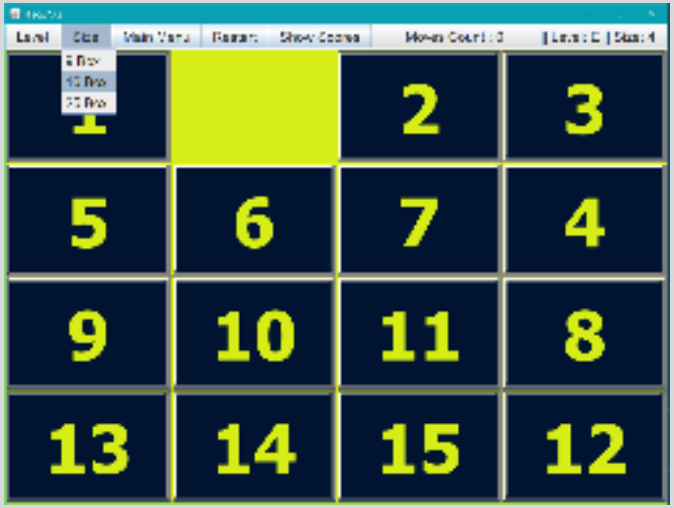

GROUP 35
IIT2020083 IIT202020168 IIT2020192 IIT2020234

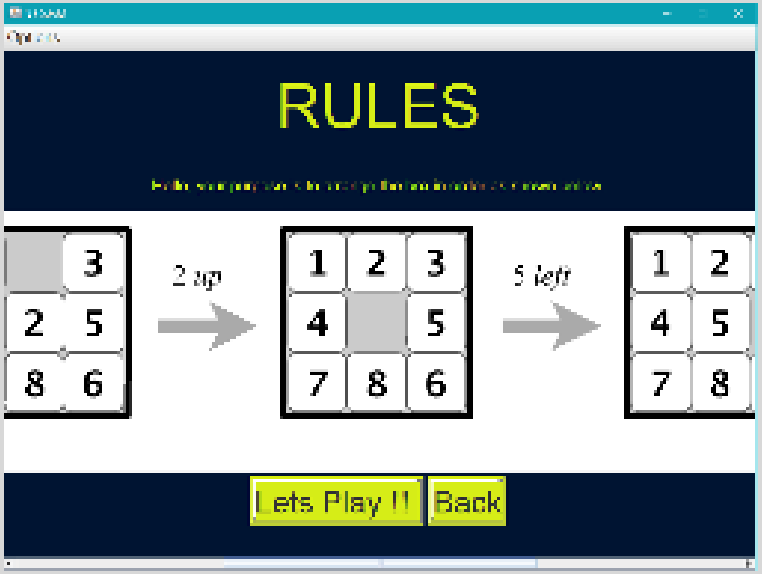

3 SQUARE
START GAME
SCORES
RULES
EXIT

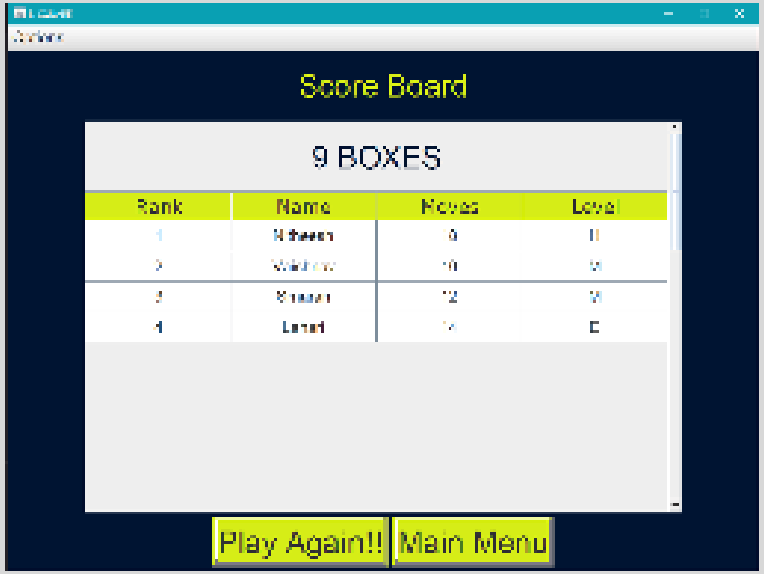

3 SQUARE
START GAME
SCORES
RULES
EXIT


1
2
4
5
3
7
8
6


1
2
3
4
6
7
8


2
3
5
6
7
4
9
10
11
8
13
14
15
12


RULES
3
2 5
8 6
2 up
1 2 3
4 5
7 8 6
5 left
1 2
4 5
7 8
Lets Play !!
Back


Score Board
9 BOXES
Rank
Name
Moves
Level
Play Again!!
Main Menu

Group-35