# PICTURE PUZZLE GAME

**BACHELOR OF TECHNOLOGY**

### IN

**INFORMATION TECHNOLOGY**

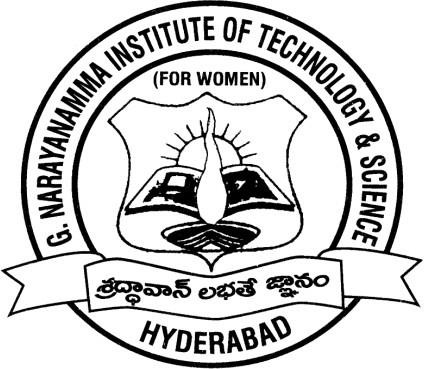
Submitted By

1. T. Hari Nandini:18251A1289
2. E. Charitha : 18251A12A0
3. Deepika : 19255A1211
4. G. Neha : 18251A12A2

Under the esteemed guidance of

**Ms. K. Sridevi**,

Assistant Professor, IT, GNITS



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**G. NARAYANAMMA INSTITUTE OF TECHNOLOGY & SCIENCE**

(For Women)

## AUTONOMOUS

**AN ISO 9001-2008 CERTIFIED INSTITUTION**

### (Accredited by NBA)

2020-2021

**CONTENTS**

|  |  |
| --- | --- |
| **TOPIC** | **Page No.** |
| Abstract | 2 |
| Introduction | 3 |
| Design Overview | 4-6 |
| Flow Chart | 7 |
| Code | 8-18 |
| Conclusion | 19 |
| Result | 20 |

**ABSTRACT**

The purpose of this project is to design and build a sliding puzzle game that will give the user the experience of solving various tricky slide puzzle through gaming.

**GAME RULES**: The picture puzzle game, contains a grid which is divided into 9 blocks, which are a part of image and one block (**default block**) is given power to swap itself any other block. The puzzle is provided to the user with the blocks shuffled. So, the user should use the default block to swap it with other blocks (according to the necessity) and arrange the given picture blocks in order to get it like the picture provided to the right (**Sample picture**). Here the main rule is that the user can only swap the default block with normal block and not two usual blocks. As and when the user completes his task to arrange the blocks in required order (as given in sample picture) the user wins the game. There is no time limit and no limit for number of swaps taken by the user. Each time a new randomly placed blocks of puzzle is provided to the user.

**Signature of HOD**

**INTRODUCTION**

The 9 Puzzle Game consists of a square board that contains 9 parts of picture divided into squares. The game is played by moving the numbered squares on the board until the image is in the order. These puzzles are also very often made with a number broken into 9 squares, with the bottom right square having the blank space (so that we can use it to swap it with other to arrange the numbers in an order).

This puzzle can be played on a computer or a hand-held electronic device. The game is greatly enhanced on an electronic device. The picture puzzle game, contains a grid which is divided into 9 blocks, which are a part of image and one block (default block) is given power to swap itself any other block. The puzzle is provided to the user with the blocks shuffled. So, the user should use the default block to swap it with other blocks (according to the necessity) and arrange the given picture blocks in order to get it like the picture provided to the right (Sample picture). Here the main rule is that the user can only swap the default block with normal block and not two usual blocks. As and when the user completes his task to arrange the blocks in required order (as given in sample picture) the user wins the game. There is no time limit and no limit for number of swaps taken by the user. Each time a new randomly placed blocks of puzzle is provided to the user.

In this project a 15 Puzzle game is implemented in java with the help of AWT/Swing with event handling.

**1.1 SCOPE**

In this document an overview is given of the 15 Puzzle game design which is based upon requirements outlined in the project proposal. Included in this document are the explanations of the methods and techniques in which the project was designed, implemented and tested in order to meet the requirements that are outlined in the project proposal. Also included are digital images to help the reader more fully understand the description of the design.

Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java. Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using there sources of OS.

The java.awt package provides classes for AWT API such as TextField,Label,TextArea,RadioButton,CheckBox,Choice,List etc.

Java Swing is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. Unlike AWT, Java Swing provides platform- independent and lightweight components. The javax.Swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

**Design Overview**

**Requirements**

1. It shall display a picture on the LCD screen.
2. It shall scramble the picture into different blocks.
3. It shall recognize user touches on the touch screen.
4. The game shall make corresponding action based on user touch location. If user touches a button the program shall access the button, if the user touches a block, it shall move the block accordingly.
5. It shall output a dialogue box on the user win.

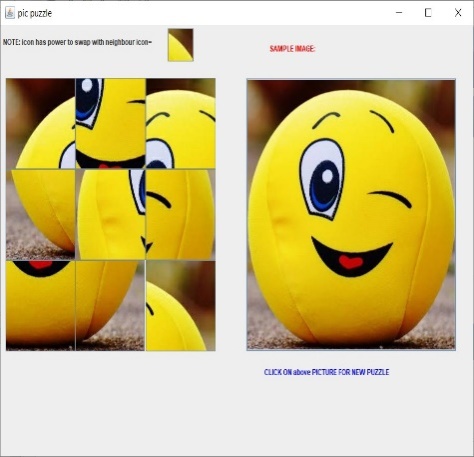
**Theory of operation**

Upon power on or reset, the main menu is displayed on the LCD as shown in Figure 1. The program then waits for a user selection and displays the corresponding screen. If the game play screen is chosen, the puzzle is shown for one second as seen and then the puzzle is scrambled with the 9th square deleted. The user is then required to press the screen over a block adjacent to the empty square. Nothing occurs if the user presses on a block not adjacent to the empty square. After the user makes a move the timer is started and the number of moves is displayed and incremented after each move. Included in the game play screen are two buttons displayed at the bottom of the screen. One is the ‘menu’ button, to return to the main menu and the other is a ‘solve’ button. If the user presses the solve button, the program proceeds to solve the puzzle from the current state and will continue until the puzzle is solved. After each user move, the program checks to see if it is in the winning state. If it is, then the timer is stopped and the time, as well as the number of moves, is displayed. The number of moves and time are then compared with the current high score values and if necessary are loaded into the Flash memory as the new high scores. In the winning state as the screen also displays a flag with “Congratulations!! YOU WON!!” . The user may then select to go back to the menu by pressing the menu button or choose the shuffle button (now in place of the solve button) to reshuffle the squares and restart the game. The completed puzzle is shown for a moment. The main menu of a figure that the LCD screen showing the winning state of the game.

**Design alternatives**

Design Alternatives As more puzzles and sounds were created it was found that the microcontroller did not have sufficient memory to fit the code. Figure 2 shows the second puzzle which consists of a picture rather than numbered blocks.

Using a picture adds difficulty as well as variety and colour to the game. It was discovered, however, that due to the limited memory of the board both levels could not be loaded to the board at the same time. The second level of the game is in a separate header file that can be loaded to the board, so both levels can be played depending on the header file that is added. An alternative method that would allow multiple levels would be to create a function to draw the separate squares rather than have a saved table of pixel values. This alternative however, would create bad graphics and require a total rewrite of the code. Another alternative method that was considered was to have the pictures used for the puzzle formatted differently. Lower quality pictures were implemented but the results were very undesirable, causing discoloration and difficulty in distinguishing each square. A method using the same pixel table to draw the blocks and then the numbers on top of the blocks separately was also considered. This method would save memory because only one pixel table instead of fifteen would be needed; however, this required lots of new code and would reduce the quality of the graphics. These methods were not chosen due to the poor graphic quality and the amount of time it would take to alter the code to allow these new methods. A method to save the high score in non-volatile memory in the LCD as a pixel was considered, however, the better option of saving to Flash memory was chosen. Flash memory was chosen because saving the high score to a pixel would create an odd pixel on the screen and would require code changes to ensure that it never got erased. Instead of using the ASCII text functions provided in lab 6 for the menu screen and high score screen, it was considered to create buttons similar to those used in the current game play screen. This would create a much better-looking interface as well as provide for a greater uniformity between the different game screens.



**Design Details:**

Software Design:

The 3\*3 Puzzle software design has two main states: The Game Play Screen, and the Winning Screen. The Game Play Screen displays the puzzle and allows the user to play, select to use the solver, or go back to the menu. The Game Play state also keeps track of the number of moves made, and the time it takes to win. Once the game has been won the user has the option to shuffle the puzzle and start over at the Game Play Screen, or go back to the menu. The winning state displays a congratulations message on the screen, outputs a sound, outputs the time it took to win on the screen, and checks if the score was a high score. If the score was a high score it is saved to the flash memory on the microcontroller. Menu Screen Output Time Shuffle Button Save High Score Solve Game Play High Score Output Sound Menu Button Check User Push Count Moves Start Time Winning State Play as shown in flowchart of Software Design 9 P.

**Flowchart**

Select image

Img1 1e

**True False**

Img2222

**True False**

Img3

**True false**

Select the empty square

Swap the empty square with another

**No**

If squares are in order

**yes**

Game over

**No**

Play again

**Yes**

Continue

Exit

**Code**

import java.awt.event.\*;

import java.awt.\*;

import javax.swing.\*;

public class picpuzzle2 extends JFrame implements ActionListener

{   JButton b1,b2,b3,b4,b5,b6,b7,b8,b9,sample,starB;

    Icon star;

    Icon ic0=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\starB0.jpg");

    Icon ic10=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\starB10.jpg");

    Icon ic20=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\starB20.jpg");

    Icon samicon1=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\main.jpg");

    Icon samicon2=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\main2.jpg");

    Icon samicon3=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\main3.jpg");

    Icon ic1=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\1.jpg");

    Icon ic2=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\5.jpg");

    Icon ic3=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\2.jpg");

    Icon ic4=new ImageIcon("C:\\ NetBeansProjects\\demoprojectjava\\src\\pic\\7.jpg");

    Icon ic5=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\4.jpg");

    Icon ic6=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\6.jpg");

    Icon ic7=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\8.jpg");

    Icon ic8=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\9.jpg");

Icon ic9=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\3.jpg");

    Icon ic11=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\12.jpg");

    Icon ic12=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\13.jpg");

    Icon ic13=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\16.jpg");

    Icon ic14=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\11.jpg");

    Icon ic15=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\14.jpg");

    Icon ic16=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\19.jpg");

    Icon ic17=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\17.jpg");

    Icon ic18=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\15.jpg");

    Icon ic19=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\18.jpg");

    Icon ic21=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\24.jpg");

    Icon ic22=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\25.jpg");

    Icon ic23=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\21.jpg");

    Icon ic24=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\27.jpg");

    Icon ic25=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\23.jpg");

    Icon ic26=new ImageIcon("C:\\ NetBeansProjects\\demoprojectjava\\src\\pic\\29.jpg");

    Icon ic27=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\28.jpg");

    Icon ic28=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\22.jpg");

    Icon ic29=new ImageIcon("C:\\NetBeansProjects\\demoprojectjava\\src\\pic\\26.jpg");

    picpuzzle2()

    {  // super("pic puzzle");

        b1=new JButton(ic1);

        b1.setBounds(10,80,138,138);

        b2=new JButton(ic2);

        b2.setBounds(110,80,138,138);

        b3=new JButton(ic3);

        b3.setBounds(210,80,138,138);

        b4=new JButton(ic4);

        b4.setBounds(10,180,138,138);

        b5=new JButton(ic5);

        b5.setBounds(110,180,138,138);

        b6=new JButton(ic6);

        b6.setBounds(210,180,138,138);

        b7=new JButton(ic7);

        b7.setBounds(10,280,138,138);

        b8=new JButton(ic8);

        b8.setBounds(110,280,138,138);

        b9=new JButton(ic9);

        b9.setBounds(210,280,138,138);

        sample=new JButton(samicon1);

        sample.setBounds(380,100,414,414);

        JLabel l1=new JLabel("Sample:");

        l1.setBounds(530,200,70,20);

        JLabel l2=new JLabel("NOTE: icon has power to swap with neighbour icon=");

        l2.setBounds(5,15,600,20);

        JLabel l3=new JLabel("CLICK ON above PICTURE FOR NEW PUZZLE");

        l3.setBounds(450,530,300,50);

        l3.setForeground(Color.blue);

        starB=new JButton(ic0);

        starB.setBounds(330,5,50,50);

        star=b9.getIcon();

        add(b1);add(b2);add(b3);add(b4);add(b5);add(b6);add(b7);add(b8);add(b9);add(sample);add(l1);add(l2);add(starB);add(l3);

        b1.addActionListener(this); b2.addActionListener(this);  b3.addActionListener(this); b4.addActionListener(this);b5.addActionListener(this);

        b6.addActionListener(this); b7.addActionListener(this); b8.addActionListener(this); b9.addActionListener(this);

        sample.addActionListener(this);

        setLayout(null);

        setSize(950,700);

        setVisible(true);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

    }

    public void actionPerformed(ActionEvent e)

    {   if(e.getSource()==b1)

        {    Icon s1=b1.getIcon();

              if(b2.getIcon()==star)

              {  b2.setIcon(s1);

                b1.setIcon(star);

              }

              else if(b4.getIcon()==star)

              { b4.setIcon(s1);

                b1.setIcon(star);

              }

        }//end of if

        if(e.getSource()==b2)

        {    Icon s1=b2.getIcon();

              if(b1.getIcon()==star)

              {  b1.setIcon(s1);

                 b2.setIcon(star);

              }

              else if(b5.getIcon()==star)

              {  b5.setIcon(s1);

                 b2.setIcon(star);

              }

              else if(b3.getIcon()==star)

              {  b3.setIcon(s1);

                 b2.setIcon(star);

              }

        }//end of if

        if(e.getSource()==b3)

        {   Icon s1=b3.getIcon();

            if(b2.getIcon()==star)

            {   b2.setIcon(s1);

                b3.setIcon(star);

            }

            else if(b6.getIcon()==star)

            {   b6.setIcon(s1);

                b3.setIcon(star);

            }

        }//end of if

         if(e.getSource()==b4)

        {    Icon s1=b4.getIcon();

              if(b1.getIcon()==star)

              {  b1.setIcon(s1);

                 b4.setIcon(star);

              }

              else if(b5.getIcon()==star)

              {  b5.setIcon(s1);

                 b4.setIcon(star);

              }

               else if(b7.getIcon()==star)

              {  b7.setIcon(s1);

                 b4.setIcon(star);

              }

        }//end of if

        if(e.getSource()==b5){

            Icon s1=b5.getIcon();

              if(b2.getIcon()==star){

                b2.setIcon(s1);

                b5.setIcon(star);

              } else if(b4.getIcon()==star){

                            b4.setIcon(s1);

                            b5.setIcon(star);

                           }

                 else if(b6.getIcon()==star){

                            b6.setIcon(s1);

                            b5.setIcon(star);

                           }

                  else if(b8.getIcon()==star){

                            b8.setIcon(s1);

                            b5.setIcon(star);

                           }

          }//end of if

        if(e.getSource()==b6){

            Icon s1=b6.getIcon();

              if(b3.getIcon()==star){

                b3.setIcon(s1);

                b6.setIcon(star);

              } else if(b5.getIcon()==star){

                            b5.setIcon(s1);

                            b6.setIcon(star);

                           }

                 else if(b9.getIcon()==star){

                            b9.setIcon(s1);

                            b6.setIcon(star);

                           }

        }//end of if

        if(e.getSource()==b7){

            Icon s1=b7.getIcon();

              if(b4.getIcon()==star){

                b4.setIcon(s1);

                b7.setIcon(star);

              } else if(b8.getIcon()==star){

                            b8.setIcon(s1);

                            b7.setIcon(star);

                           }

         }//end of if

           if(e.getSource()==b8){

            Icon s1=b8.getIcon();

              if(b7.getIcon()==star){

                b7.setIcon(s1);

                b8.setIcon(star);

              } else if(b5.getIcon()==star){

                            b5.setIcon(s1);

                            b8.setIcon(star);

                           }

                 else if(b9.getIcon()==star){

                            b9.setIcon(s1);

                            b8.setIcon(star);

                           }

          }//end of if

         if(e.getSource()==b9){

            Icon s1=b9.getIcon();

              if(b8.getIcon()==star){

                b8.setIcon(s1);

                b9.setIcon(star);

              } else if(b6.getIcon()==star){

                            b6.setIcon(s1);

                            b9.setIcon(star);

                           }

          }//end of if

if(b1.getIcon() == ic1 && b2.getIcon() == ic2 && b3.getIcon() == ic3 && b4.getIcon() == ic4 && b5.getIcon() == ic5 && b6.getIcon() == ic6 && b7.getIcon() == ic7 && b8.getIcon() == ic8&&b9.getIcon() == ic9)

{ JOptionPane.showMessageDialog(picpuzzle2.this,"!!!CONGRATULATIONS!!!YOU WON\nCLICK ON SAMPLE PICTURE FOR NEXT");

}

        if(e.getSource()==sample)

        {    Icon s1=sample.getIcon();

             if(s1==samicon3)

             {  sample.setIcon(samicon1);

                b1.setIcon(ic1);

                b2.setIcon(ic2);

                b3.setIcon(ic3);

                b4.setIcon(ic4);

                b5.setIcon(ic5);

                b6.setIcon(ic6);

                b7.setIcon(ic7);

                b8.setIcon(ic8);

                b9.setIcon(ic9);

                star=b9.getIcon();

                starB.setIcon(ic0);

             }//eof if

            else if(s1==samicon1)

            {   sample.setIcon(samicon2);

                b1.setIcon(ic11);

                b2.setIcon(ic12);

                b3.setIcon(ic13);

                b4.setIcon(ic14);

                b5.setIcon(ic15);

                b6.setIcon(ic16);

                b7.setIcon(ic17);

                b8.setIcon(ic18);

                b9.setIcon(ic19);

                star=b6.getIcon();

                starB.setIcon(ic10);

            }//eof else

            else

            {   sample.setIcon(samicon3);

                b1.setIcon(ic21);

                b2.setIcon(ic22);

                b3.setIcon(ic23);

                b4.setIcon(ic24);

                b5.setIcon(ic25);

                b6.setIcon(ic26);

                b7.setIcon(ic27);

                b8.setIcon(ic28);

                b9.setIcon(ic29);

                star=b6.getIcon();

                starB.setIcon(ic20);

            }//eof else

        }  //eof outer if

//

    }//end of actionPerformed

    public static void main(String args[])

    {  new picpuzzle2();  }//end of main

}//end of class

**Conclusion of sliding puzzle game using java swings:**

The goal of this little game is to form a picture. Buttons containing images are moved by clicking on them. Only buttons adjacent to the empty button can be moved.

We scale the image and cut it into 9 pieces. These pieces are used by JButton components. The last piece is not used. we have an empty button instead. It is a button that does not have an image. Other buttons swap space with this one.

You may only slide squares up, down, left or right and not diagonally.

We use a Grid Layout to store our components. The layout consists of 3 rows and 3 columns.

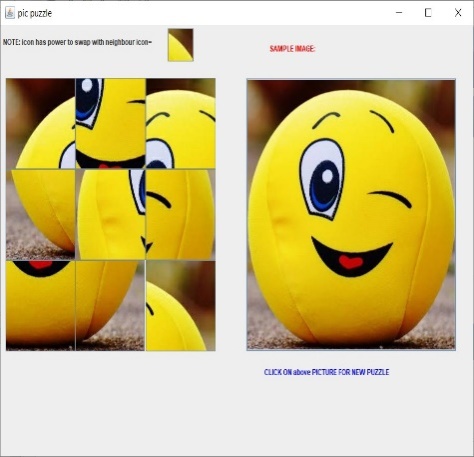
Buttons are identified by their numbers assigned to them. It is a point containing the button's correct row and column position in the picture. These properties are used to find out if we have the correct order of buttons in the window.

 Usually, the goal can only be achieved by moving all of the other pieces in a certain specified order of moves. Each piece may be restricted in the direction that it can move (vertically or horizontally).

The Objective of the game is to set all the squares in order starting from top left corner to bottom right corner such that all the numbers assigned to the squares are arranged sequentially and the bottom right corner box is left emptied. If squares are arranged in this proper manner then a dialog box is received to the player by congratulating for finishing the task.

If the player wants to play picture puzzle then they must make sure that all the broken pictures of respective pictures are fixed properly such that it forms an complete meaningful image. Once if player is able is set all the respective squares sequentially then game is over.

**OUTPUT:**

****

