FUNDAMENTAL OF ALGORITHUM PROJECT

"PacMan "

BATCH : B -5

VRISHTI GAHLAUT (12103528)
DHRUV SINGH (12103533)
SUMIT BANSAL (12103530)
HAMZA KHAN (12103540)

INDEX

Certificate Of Acknowledgment

Jaypee Institute Of Information And Technology

This is to certify that ,on this day of 7 May,2014 , Sumit Bansal(12103530), Dhruv Singh(12103533) , Hamza Khan(12103540), Vrishti Gahlaut (12103528) have successfully completed their project on topic “Pacman0.0”.

Acknowledgment

We take this opportunity to express our profound gratitude and deep regards to our guide and professor MS. ANKITA WADHWA for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her has played a major role in making our project a success.

We also take this opportunity to express a deep sense of gratitude to Company Mentors for their cordial support, valuable information and guidance, which helped me in completing this task through various stages.

Lastly, We thank almighty, my parents, brother, sisters and friends for their constant encouragement without which this assignment would not be possible.

Introduction:

PacMAn:

The player controls Pac-Man through a maze , eating pac-dots (also called pellets).However a monster always keeps following him wherever it goes. When all pac-dots are eaten, Pac-Man is taken to the next stage. Enemy roams around the maze, trying to catch Pac-Man. If an enemy touches Pac-Man, a life is lost and the Pac-Man itself withers and dies.
The game ends when Pac-man collects all the dots without being eaten by the monster!!

ALGORITHM USED..

The basic target is to find the shortest path to our target (Pac-man). In our project we use Dijkstra's algorithm and backtracking (Greedy)to find shortest path .Originally the game uses A* algorithm , however both algorithms serve our purpose.

Project Code:

```
import java.lang.*;
import java.awt.EventQueue;
import javax.swing.JFrame;
import java.awt.*;
import javax.imageio.ImageIO;
import java.io.*;
import javax.swing.WindowConstants;
import java.awt.image.*;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Event;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

import javax.swing.ImageIcon;
import javax.swing.JPanel;
import javax.swing.Timer;
import javax.swing.*;

 class Board extends JPanel implements ActionListener {
```

```
private Dimension d;
private final Font smallfont = new Font("Test-1", Font.BOLD, 14);
      static int flag=0;
private Image ii;
private final Color dotcolor = new Color(192, 192, 0);
private Color mazecolor;

private boolean ingame = false;
private boolean dying = false;

private final int blocksize = 24;
private final int nrofblocks = 15;
private final int scrsize = nrofblocks * blocksize;
private final int pacanimdelay = 2;
private final int pacmananimcount = 4;
private final int maxghosts = 6;
private final int pacmanspeed = 6;

private int pacanimcount = pacanimdelay;
private int pacanimdir = 1;
private int pacmananimpos = 0;
private int nrofghosts = 0;
private int pacsleft, score;
private int[] dx, dy;
private int[] ghostx, ghosty, ghostdx, ghostdy, ghostspeed;

private Image ghost,logo;
private Image pacman1, pacman2up, pacman2left, pacman2right, pacman2down;
private Image pacman3up, pacman3down, pacman3left, pacman3right;
private Image pacman4up, pacman4down, pacman4left, pacman4right;

private int pacmanx, pacmany, pacmandx, pacmandy;
private int reqdx, reqdy, viewdx, viewdy;

private final short leveldata[] = {
   19, 26, 26, 26, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 22,
   21, 0, 0, 0, 17, 16, 16, 16, 16, 16, 16, 16, 16, 16, 20,
   21, 0, 0, 0, 17, 16, 16, 16, 16, 16, 16, 16, 16, 16, 20,
   21, 0, 0, 0, 17, 16, 16, 24, 16, 16, 16, 16, 16, 16, 20,
   17, 18, 18, 18, 16, 16, 20, 0, 17, 16, 16, 16, 16, 16, 20,
   17, 16, 16, 16, 16, 16, 20, 0, 17, 16, 16, 16, 16, 24, 20,
   25, 16, 16, 16, 24, 24, 28, 0, 25, 24, 24, 16, 20, 0, 21,
   1, 17, 16, 20, 0, 0, 0, 0, 0, 0, 0, 17, 20, 0, 21,
   1, 17, 16, 16, 18, 18, 22, 0, 19, 18, 18, 16, 20, 0, 21,
   1, 17, 16, 16, 16, 16, 20, 0, 17, 16, 16, 16, 20, 0, 21,
   1, 17, 16, 16, 16, 16, 20, 0, 17, 16, 16, 16, 20, 0, 21,
   1, 17, 16, 16, 16, 16, 16, 18, 16, 16, 16, 16, 20, 0, 21,
   1, 17, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 20, 0, 21,
```

```
    1, 25, 24, 24, 24, 24, 24, 24, 24, 24, 16, 16, 16, 18, 20,
    9, 8, 8, 8, 8, 8, 8, 8, 8, 8, 25, 24, 24, 24, 28
};

private final int validspeeds[] = {1, 2, 3, 4, 6, 8};
private final int maxspeed = 6;

private int currentspeed = 3;
private short[] screendata;
private Timer timer;

public Board() {

    loadImages();
    initVariables();

    addKeyListener(new TAdapter());

    setFocusable(true);

    setBackground(Color.black);
    setDoubleBuffered(true);
}


private void initVariables() {

    screendata = new short[nrofblocks * nrofblocks];
    mazecolor = new Color(5, 100, 5);
    d = new Dimension(400, 400);
    ghostx = new int[maxghosts];
    ghostdx = new int[maxghosts];
    ghosty = new int[maxghosts];
    ghostdy = new int[maxghosts];
    ghostspeed = new int[maxghosts];
    dx = new int[4];
    dy = new int[4];

    timer = new Timer(40, this);
    timer.start();
}

@Override
public void addNotify() {
    super.addNotify();

    initGame();
}
```

```
private void doAnim() {

    pacanimcount--;

    if (pacanimcount <= 0) {
        pacanimcount = pacanimdelay;
        pacmananimpos = pacmananimpos + pacanimdir;

        if (pacmananimpos == (pacmananimcount - 1) || pacmananimpos == 0) {
            pacanimdir = -pacanimdir;
        }
    }
}

private void playGame(Graphics2D g2d) {

    if (dying) {

        death();

    } else {

        movePacman();
        drawPacman(g2d);
        moveGhosts(g2d);
        checkMaze();
    }
}

private void showIntroScreen(Graphics2D g2d) {

    g2d.setColor(new Color(0, 32, 48));
    g2d.fillRect(50, scrsize / 2 - 30, scrsize - 100, 50);
    g2d.setColor(Color.white);
    g2d.drawRect(50, scrsize / 2 - 30, scrsize - 100, 50);

    String s = "Press s to start.";
    Font small = new Font("Helvetica", Font.BOLD, 14);
    FontMetrics metr = this.getFontMetrics(small);

    g2d.setColor(Color.white);
    g2d.setFont(small);
    g2d.drawString(s, (scrsize - metr.stringWidth(s)) / 2, scrsize / 2);
}

private void drawScore(Graphics2D g) {

    int i;
    String s;
```

```
        g.setFont(smallfont);
        g.setColor(new Color(96, 128, 255));
        s = "Score: " + score;
        g.drawString(s, scrsize / 2 + 96, scrsize + 16);

        for (i = 0; i < pacsleft; i++) {
            g.drawImage(pacman3left, i * 28 + 8, scrsize + 1, this);
        }
    }

    private void checkMaze() {

        short i = 0;
        boolean finished = true;

        while (i < nrofblocks * nrofblocks && finished) {

            if ((screendata[i] ) != 0) {
                finished = false;
            }

            i++;
        }

        if (finished) {

            score += 50;

            if (nrofghosts < 2) {
                nrofghosts++;
            }

            if (currentspeed < maxspeed) {
                currentspeed++;
            }

            initLevel();
        }
    }

    private void death() {

        pacsleft--;

        if (pacsleft == 0) {
            ingame = false;
        }
```

```
        continueLevel();
    }

    private void moveGhosts(Graphics2D g2d) {

        short i;
        int pos;
        int count;

        for (i = 0; i < 6; i++) {
            if (ghostx[i] % blocksize == 0 && ghosty[i] % blocksize == 0) {
                pos = ghostx[i] / blocksize + nrofblocks * (int) (ghosty[i] / blocksize);

                count = 0;
                                //      System.out.println(" "+pacmanx+"   "+pacmany);
                                if(i==0&&(screendata[pos] & 1) == 0&&(screendata[pos] & 2) ==
0&&(screendata[pos] & 4) == 0&&(screendata[pos] & 8) == 0)
                                {
                                flag++;
                                if(flag%2==0)
                                {
                                new Dijkstra();
                                }
                                else
                                {

                                if(Math.abs(pacmanx-ghostx[i])>Math.abs(pacmany-ghosty[i]))
                                {
                                        if(pacmanx>ghostx[i])
                                        {

                                        System.out.println(" right");
                                                dx[count] = 1;
                        dy[count] = 0;
                                                count++;
                                        }
                                        else
                                        {

                                        System.out.println(" left");
                                                dx[count] = -1;
                        dy[count] = 0;
                                                count++;
                                        }

                                }
                                else
                                {
```

```
                        if(pacmany>ghosty[i])
                        {

                        System.out.println(" down");
                                dx[count] = 0;
        dy[count] = 1;
                                count++;
                        }
                        else
                        {

                        System.out.println(" up");
                                dx[count] = 0;
        dy[count] = -1;
                                count++;
                        }

                }

                }

/*              if(pacmanx<ghostx[i])
                {
                        new Dijkstra();
                        System.out.println(" left");
                                dx[count] = -1;
        dy[count] = 0;
                                count++;
                }
                else if(pacmanx>ghostx[i])
                {
                        new Dijkstra();
                        System.out.println(" right");
                                dx[count] = 1;
        dy[count] = 0;
                                count++;
                }
                else if(pacmany>ghosty[i])
                {
                        new Dijkstra();
                        System.out.println(" down");
                                dx[count] = 0;
        dy[count] = 1;
                                count++;
                }
                else
                {
                        new Dijkstra();
                        System.out.println(" up");
```

```
                                dx[count] = 0;
    dy[count] = -1;
                                count++;
                }*/
                                ghostdx[i] = dx[count];
    ghostdy[i] = dy[count];

                }

                else     {
if ((screendata[pos] & 1) == 0 && ghostdx[i] != 1) {
    dx[count] = -1;
    dy[count] = 0;
    count++;
}

if ((screendata[pos] & 2) == 0 && ghostdy[i] != 1) {
    dx[count] = 0;
    dy[count] = -1;
    count++;
}

if ((screendata[pos] & 4) == 0 && ghostdx[i] != -1) {
    dx[count] = 1;
    dy[count] = 0;
    count++;
}

if ((screendata[pos] & 8) == 0 && ghostdy[i] != -1) {
    dx[count] = 0;
    dy[count] = 1;
    count++;
}

if (count == 0) {

    if ((screendata[pos] & 15) == 15) {
        ghostdx[i] = 0;
        ghostdy[i] = 0;
    } else {
        ghostdx[i] = -ghostdx[i];
        ghostdy[i] = -ghostdy[i];
    }

} else {

    count = (int) (Math.random() * count);

    if (count > 3) {
```

```
                count = 3;
            }

            ghostdx[i] = dx[count];
            ghostdy[i] = dy[count];
        }

    }
                        }
    ghostx[i] = ghostx[i] + (ghostdx[i] * ghostspeed[i]);
    ghosty[i] = ghosty[i] + (ghostdy[i] * ghostspeed[i]);
    drawGhost(g2d, ghostx[i] + 1, ghosty[i] + 1);

    if (pacmanx > (ghostx[i] - 12) && pacmanx < (ghostx[i] + 12)
        && pacmany > (ghosty[i] - 12) && pacmany < (ghosty[i] + 12)
        && ingame) {

      dying = true;
                                System.out.println("bhoot!!");
    }
  }
}

private void drawGhost(Graphics2D g2d, int x, int y) {

  g2d.drawImage(ghost, x, y, this);
}

private void movePacman() {

  int pos;
  short ch;

  if (reqdx == -pacmandx && reqdy == -pacmandy) {
    pacmandx = reqdx;
    pacmandy = reqdy;
    viewdx = pacmandx;
    viewdy = pacmandy;
  }

  if (pacmanx % blocksize == 0 && pacmany % blocksize == 0) {
    pos = pacmanx / blocksize + nrofblocks * (int) (pacmany / blocksize);
    ch = screendata[pos];

    if ((ch & 16) != 0) {
      screendata[pos] = (short) (ch & 15);
      score++;
                                if(score == 179)
                                {
```

```
                                gameEnd();
                                }
        }

        if (reqdx != 0 || reqdy != 0) {
           if (!((reqdx == -1 && reqdy == 0 && (ch & 1) != 0)
                 || (reqdx == 1 && reqdy == 0 && (ch & 4) != 0)
                 || (reqdx == 0 && reqdy == -1 && (ch & 2) != 0)
                 || (reqdx == 0 && reqdy == 1 && (ch & 8) != 0))) {
              pacmandx = reqdx;
              pacmandy = reqdy;
              viewdx = pacmandx;
              viewdy = pacmandy;
           }
        }

        // Check for standstill
        if ((pacmandx == -1 && pacmandy == 0 && (ch & 1) != 0)
              || (pacmandx == 1 && pacmandy == 0 && (ch & 4) != 0)
              || (pacmandx == 0 && pacmandy == -1 && (ch & 2) != 0)
              || (pacmandx == 0 && pacmandy == 1 && (ch & 8) != 0)) {
           pacmandx = 0;
           pacmandy = 0;
        }
     }
     pacmanx = pacmanx + pacmanspeed * pacmandx;
     pacmany = pacmany + pacmanspeed * pacmandy;
}

private void drawPacman(Graphics2D g2d) {

     if (viewdx == -1) {
        drawPacnanLeft(g2d);
     } else if (viewdx == 1) {
        drawPacmanRight(g2d);
     } else if (viewdy == -1) {
        drawPacmanUp(g2d);
     } else {
        drawPacmanDown(g2d);
     }
}

        private void gameEnd()
        {
        Frame frm = new Frame("");
        frm.setSize(700,380);

        try{
```

```
        final BufferedImage end = ImageIO.read(new File("images/end.png"));
                JPanel pane1 = new JPanel() {
        @Override
        protected void paintComponent(Graphics g) {
           super.paintComponent(g);
           g.drawImage(end, 0, 0, null);

        }
     };


                                frm.add(pane1);
                }catch( Exception e){}

        frm.setVisible(true);



        //System.exit(0);
        }


private void drawPacmanUp(Graphics2D g2d) {

    switch (pacmananimpos) {
        case 1:
            g2d.drawImage(pacman2up, pacmanx + 1, pacmany + 1, this);
            break;
        case 2:
            g2d.drawImage(pacman3up, pacmanx + 1, pacmany + 1, this);
            break;
        case 3:
            g2d.drawImage(pacman4up, pacmanx + 1, pacmany + 1, this);
            break;
        default:
            g2d.drawImage(pacman1, pacmanx + 1, pacmany + 1, this);
            break;
    }
}

private void drawPacmanDown(Graphics2D g2d) {

    switch (pacmananimpos) {
        case 1:
            g2d.drawImage(pacman2down, pacmanx + 1, pacmany + 1, this);
            break;
        case 2:
            g2d.drawImage(pacman3down, pacmanx + 1, pacmany + 1, this);
            break;
```

```
            case 3:
                g2d.drawImage(pacman4down, pacmanx + 1, pacmany + 1, this);
                break;
            default:
                g2d.drawImage(pacman1, pacmanx + 1, pacmany + 1, this);
                break;
        }
    }

    private void drawPacnanLeft(Graphics2D g2d) {

        switch (pacmananimpos) {
            case 1:
                g2d.drawImage(pacman2left, pacmanx + 1, pacmany + 1, this);
                break;
            case 2:
                g2d.drawImage(pacman3left, pacmanx + 1, pacmany + 1, this);
                break;
            case 3:
                g2d.drawImage(pacman4left, pacmanx + 1, pacmany + 1, this);
                break;
            default:
                g2d.drawImage(pacman1, pacmanx + 1, pacmany + 1, this);
                break;
        }
    }

    private void drawPacmanRight(Graphics2D g2d) {

        switch (pacmananimpos) {
            case 1:
                g2d.drawImage(pacman2right, pacmanx + 1, pacmany + 1, this);
                break;
            case 2:
                g2d.drawImage(pacman3right, pacmanx + 1, pacmany + 1, this);
                break;
            case 3:
                g2d.drawImage(pacman4right, pacmanx + 1, pacmany + 1, this);
                break;
            default:
                g2d.drawImage(pacman1, pacmanx + 1, pacmany + 1, this);
                break;
        }
    }

    private void drawMaze(Graphics2D g2d) {

        short i = 0;
        int x, y;
```

```
    for (y = 0; y < scrsize; y += blocksize) {
        for (x = 0; x < scrsize; x += blocksize) {

            g2d.setColor(mazecolor);
            g2d.setStroke(new BasicStroke(2));

            if ((screendata[i] & 1) != 0) {
                g2d.drawLine(x, y, x, y + blocksize - 1);
            }

            if ((screendata[i] & 2) != 0) {
                g2d.drawLine(x, y, x + blocksize - 1, y);
            }

            if ((screendata[i] & 4) != 0) {
                g2d.drawLine(x + blocksize - 1, y, x + blocksize - 1,
                        y + blocksize - 1);
            }

            if ((screendata[i] & 8) != 0) {
                g2d.drawLine(x, y + blocksize - 1, x + blocksize - 1,
                        y + blocksize - 1);
            }

            if ((screendata[i] & 16) != 0) {
                g2d.setColor(dotcolor);
                g2d.fillRect(x + 11, y + 11, 2, 2);
            }

            i++;
        }
    }
}

private void initGame() {

    pacsleft = 3;
    score = 0;
    initLevel();
    nrofghosts = 6;
    currentspeed = 3;
}

private void initLevel() {

    int i;
    for (i = 0; i < nrofblocks * nrofblocks; i++) {
        screendata[i] = leveldata[i];
```

```
        }

        continueLevel();
    }

    private void continueLevel() {

        short i;
        int dx = 1;
        int random;

        for (i = 0; i < nrofghosts; i++) {

            ghosty[i] = 4 * blocksize;
            ghostx[i] = 4 * blocksize;
            ghostdy[i] = 0;
            ghostdx[i] = dx;
            dx = -dx;
            random = (int) (Math.random() * (currentspeed + 1));

            if (random > currentspeed) {
                random = currentspeed;
            }

            ghostspeed[i] = validspeeds[random];
        }

        pacmanx = 7 * blocksize;
        pacmany = 11 * blocksize;
        pacmandx = 0;
        pacmandy = 0;
        reqdx = 0;
        reqdy = 0;
        viewdx = -1;
        viewdy = 0;
        dying = false;
    }

    private void loadImages() {

        ghost = new ImageIcon("images/ghost.png").getImage();
        pacman1 = new ImageIcon("images/pacman.png").getImage();
        pacman2up = new ImageIcon("images/up1.png").getImage();
        pacman3up = new ImageIcon("images/up2.png").getImage();
        pacman4up = new ImageIcon("images/up3.png").getImage();
        pacman2down = new ImageIcon("images/down1.png").getImage();
        pacman3down = new ImageIcon("images/down2.png").getImage();
        pacman4down = new ImageIcon("images/down3.png").getImage();
        pacman2left = new ImageIcon("images/left1.png").getImage();
```

```
    pacman3left = new ImageIcon("images/left2.png").getImage();
    pacman4left = new ImageIcon("images/left3.png").getImage();
    pacman2right = new ImageIcon("images/right1.png").getImage();
    pacman3right = new ImageIcon("images/right2.png").getImage();
    pacman4right = new ImageIcon("images/right3.png").getImage();

}

@Override
public void paintComponent(Graphics g) {
    super.paintComponent(g);

    doDrawing(g);
}

private void doDrawing(Graphics g) {

    Graphics2D g2d = (Graphics2D) g;

    g2d.setColor(Color.black);
    g2d.fillRect(0, 0, d.width, d.height);

    drawMaze(g2d);
    drawScore(g2d);
    doAnim();

    if (ingame) {
        playGame(g2d);
    } else {
        showIntroScreen(g2d);
    }

    g2d.drawImage(ii, 5, 5, this);
    Toolkit.getDefaultToolkit().sync();
    g2d.dispose();
}

class TAdapter extends KeyAdapter {

    @Override
    public void keyPressed(KeyEvent e) {

        int key = e.getKeyCode();

        if (ingame) {
            if (key == KeyEvent.VK_LEFT) {
                reqdx = -1;
                reqdy = 0;
            } else if (key == KeyEvent.VK_RIGHT) {
```

```
                reqdx = 1;
                reqdy = 0;
            } else if (key == KeyEvent.VK_UP) {
                reqdx = 0;
                reqdy = -1;
            } else if (key == KeyEvent.VK_DOWN) {
                reqdx = 0;
                reqdy = 1;
            } else if (key == KeyEvent.VK_ESCAPE && timer.isRunning()) {
                ingame = false;
            } else if (key == KeyEvent.VK_PAUSE) {
                if (timer.isRunning()) {
                    timer.stop();
                } else {
                    timer.start();
                }
            }
        } else {
            if (key == 's' || key == 'S') {
                ingame = true;
                initGame();
            }
        }
    }

    @Override
    public void keyReleased(KeyEvent e) {

        int key = e.getKeyCode();

        if (key == Event.LEFT || key == Event.RIGHT
                || key == Event.UP || key == Event.DOWN) {
            reqdx = 0;
            reqdy = 0;
        }
    }
}

@Override
public void actionPerformed(ActionEvent e) {

    repaint();
}
}

public class Pacman extends JFrame {

    public Pacman() {
```

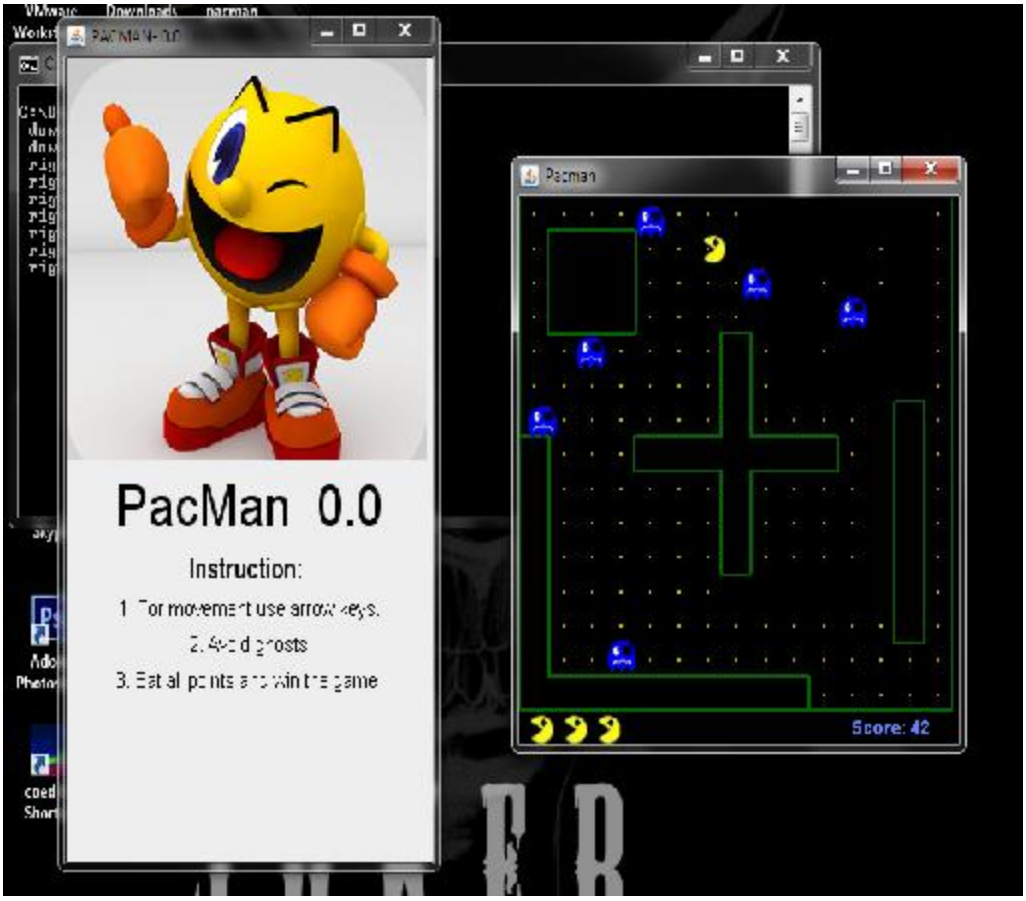
```
        initUI();
    }

    private void initUI() {

        add(new Board());
        setTitle("Pacman");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(380, 420);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    public static void main(String[] args) {
                    new Logo();
        EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                Pacman ex = new Pacman();
                ex.setVisible(true);
            }
        });
    }
}



class Logo      {
            public Logo()
            {
                    JFrame f = buildFrame();
                    f.setSize(320,600);
                    f.setLayout(new GridLayout(2,1));
            //      f.setLayout(new GridBagLayout());
                    f.setBackground(Color.black);
                    try{
                    final BufferedImage logo = ImageIO.read(new File("images/pacman_logo.png"));
                    final BufferedImage sco = ImageIO.read(new File("images/sco.png"));
                    JPanel pane = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.drawImage(logo, 0, 0, null);
                                    //g.drawImage(sco,600, 0, null);
            }
        };
```

```
                                        f.add(pane);
                    }catch( Exception e){}

      // JLabel label1 = new JLabel("Test");
             // label1.setText("Label Text");
            //         f.add(label1);

            JPanel panel = new JPanel();
    JLabel jlabel = new JLabel("");
    jlabel.setFont(new Font("SANS_SERIF",40,40));
            jlabel.setForeground(Color.black);
            jlabel.setText("PacMan  0.0");
    panel.add(jlabel);
            JLabel ins1 = new JLabel("Instruction:");
            ins1.setFont(new Font("SANS_SERIF",0,20));
            ins1.setForeground(Color.black);
            JLabel inss = new JLabel("");
            JLabel ins2 = new JLabel("1. For movement use arrow keys.");
            JLabel ins3 = new JLabel("2. Avoid ghosts");
            JLabel ins4 = new JLabel("3. Eat all points and win the game.");
            ins1.setFont(new Font("SANS_SERIF",0,20));
            ins1.setForeground(Color.black);
            ins2.setFont(new Font("SANS_SERIF",0,15));
            ins2.setForeground(Color.black);
            ins3.setFont(new Font("SANS_SERIF",0,15));
            ins3.setForeground(Color.black);
            ins4.setFont(new Font("SANS_SERIF",0,15));
            ins4.setForeground(Color.black);

            panel.add(ins1);
            panel.add(inss);
            panel.add(ins2);
            panel.add(ins3);
            panel.add(ins4);
   // panel.setBorder(new LineBorder(Color.BLACK)); // make it easy to see
    f.add(panel, new GridBagConstraints());

  //  f.setLocationRelativeTo(null);
    f.setDefaultCloseOperation(f.EXIT_ON_CLOSE);

                    f.setVisible(true);
            }
            private static JFrame buildFrame() {
        JFrame frame = new JFrame("PACMAN- 0.0");
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        return frame;
    }

}
```

```
class Dijkstra
{
        static int[][] cost;
   static int[] dist;
        static int n =255;

        public Dijkstra()
        {

        }
        public Dijkstra(int v)
        {

      cost = new int[n][n];
      dist = new int[n];
      for (int i = 0; i < n; i++) {
         for (int j = 0; j < n; j++) {
            if (i == j) {
               cost[i][j] = 0;
            } else {
               cost[i][j] = 99999;
            }
         }
      }
                  shortestPath(v);
         }

         static void shortestPath(int v) {
      int[] s = new int[n];
      for (int i = 0; i < n; i++) {
         s[i] = 0;
         dist[i] = cost[v][i];
      }
      s[v] = 1;
      dist[v] = 0;
      for (int i = 1; i < n - 1; i++) {
         int u = 0, dis = 0;
         for (int j = 0; j < s.length; j++) {
            if (s[j] == 0) {
               dis = dist[j];
               u = j;
               for (int k = j + 1; k < s.length; k++) {

                  if (dis > dist[k] && s[k] == 0) {
                     dis = dist[k];
                     u = k;
                  }
```

```
                }
                break;
            }
        }
        s[u] = 1;
        for (int j = 1; j < n; j++) {
            if (s[j] == 0) {
                if (dist[j] > (dist[u] + cost[u][j])) {
                    dist[j] = dist[u] + cost[u][j];

                }
            }
        }
    }
}
}
```

SnapShots:


PacMan 0.0
Instruction:
Pacman
Score: 42

BIBLIOGRAPHY

1. Java Doc file
2. Wikipedia
3. Algorithms by Cormen
4. Fundamental Of Algorithms by Sartaj Sahni