**Introduction:** Pacman is an arcade game originally developed by a Japanese company Namco in 1980. Pacman became one of the most popular arcade games ever created. The main goal of this game is to collect all the points (seen as white dots) in the maze and avoid the enemies which appear as little ghosts. We animated his body with four GIFs, according to directions. The animation is used to create the moving picture of Pacman opening and closing his mouth frame by frame. The maze consists of 15x15 blocks. The structure of the maze is based on a simple array of integers. Pacman has more than one chances (By default it is three but it is changeable). It also counts the score.

We made this game using Java language and application for Java is Netbeans.

This project consists of Java files, Pacman.java and Model.java. The Model.java contains all the stuff which constructs the maze, pac-man, ghosts and white dots (Food). The Pacman.java initiates the game and uses the code written in Model.java.

The Pac-man is controlled by cursor keys which is left, right, up and down. The SPACE is used to pause and start the game.

The code on line 32:

private int pacman\_x, pacman\_y, pacmand\_x, pacmand\_y;

The first two variables pacman\_x and pacman\_y is used for X and Y co-ordinates of Pacman. The last variables pacmand\_x and pacmand\_y is used for vertical and horizontal directions.

The code on line 35:

private final short levelData[] =

{

19, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 22,

17, 16, 24, 24, 24, 24, 16, 16, 16, 24, 24, 24, 24, 16, 20,

17, 20, 0, 0, 0, 0, 17, 16, 20, 0, 0, 0, 0, 17, 20,

17, 20, 0, 19, 18, 18, 16, 16, 16, 18, 18, 22, 0, 17, 20,

17, 20, 0, 17, 16, 16, 16, 16, 16, 16, 16, 20, 0, 17, 20,

17, 20, 0, 17, 16, 16, 24, 24, 24, 16, 16, 20, 0, 17, 20,

17, 16, 18, 16, 16, 20, 0, 0, 0, 17, 16, 16, 18, 16, 20,

17, 16, 16, 16, 16, 20, 0, 0, 0, 17, 16, 16, 16, 16, 20,

17, 16, 24, 16, 16, 20, 0, 0, 0, 17, 16, 16, 24, 16, 20,

17, 20, 0, 17, 16, 16, 18, 18, 18, 16, 16, 20, 0, 17, 20,

17, 20, 0, 17, 16, 16, 16, 16, 16, 16, 16, 20, 0, 17, 20,

17, 20, 0, 25, 24, 24, 16, 16, 16, 24, 24, 28, 0, 17, 20,

17, 20, 0, 0, 0, 0, 17, 16, 20, 0, 0, 0, 0, 17, 20,

17, 16, 18, 18, 18, 18, 16, 16, 16, 18, 18, 18, 18, 16, 20,

25, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 28,

};

These collection of numbers build up the maze. Number 1 represents left. Number 2, 4 and 8 represents up, right and down. Whereas Number 16 is a point. To make a border you need to add these numbers up, such as for left and top border we add up 1 + 2 + 16 = 19. The 19 puts borders on left and top for the block. You can design your maze according to your likings.

The code on line 80:

private void initVariables() {

screenData = new short[N\_BLOCKS \* N\_BLOCKS];

d = new Dimension(400, 400);

ghost\_x = new int[MAX\_GHOSTS];

ghost\_dx = new int[MAX\_GHOSTS];

ghost\_y = new int[MAX\_GHOSTS];

ghost\_dy = new int[MAX\_GHOSTS];

ghostSpeed = new int[MAX\_GHOSTS];

dx = new int[4];

dy = new int[4];

timer = new Timer(40, this);

timer.start();

}

This code is used to initiate variables for screen data, screen resolution and ghost. ghost\_x and ghost\_y is for directions x-coordinates and y-coordinates whereas ghost\_dx and ghost\_dy is for vertical and horizontal directions.

timer = new Timer(40, this);

timer.start();

These two lines are used for refresh rate of animations and characters. The value 40 is actually 40 mili-seconds, the refresh rate by default is 40 mili-seconds. It can increased or decreased but some values may give an error or do some trouble with FPS.

The code on line 96:

private void playGame(Graphics2D g2d) {

if (dying) {

death();

} else {

movePacman();

drawPacman(g2d);

moveGhosts(g2d);

checkMaze();

}

}

If Pac-man loses all of it lives the will be over and the game will be restarted. The other four methods are used to draw Pac-man as in initiate Pac-man’s block and will be used for moving Pac-man. The last methods are used to move ghosts in random directions and building up the maze.

The lines:

private void showIntroScreen(Graphics2D g2d) {

String start = "PRESS SPACE TO PLAY";

g2d.setColor(new Color(255, 223, 0)); //Score color

g2d.drawString(start, (SCREEN\_SIZE)/4, 150);

}

private void drawScore(Graphics2D g) {

g.setFont(smallFont); //Font

g.setColor(new Color(135, 206, 235)); //Score color

String s = "YOUR SCORE: " + score; //Will show your score

g.drawString(s, SCREEN\_SIZE / 2 + 25, SCREEN\_SIZE + 16); //Direction

for (int i = 0; i < lives; i++) {

g.drawImage(heart, i \* 28 + 8, SCREEN\_SIZE + 1, this); //Direction

}

}

The first part is used for intro screen, the font and color can be changed without any risk. The second part is used for score. It will show your score, color and font can be changed as well.

private void moveGhosts(Graphics2D g2d)

This code will move Ghosts in random directions, the Ghosts will move on its own.

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

This code will initiate the Ghosts.

private void movePacman()

This code will be used to move Pac-man around the maze.

private void drawPacman(Graphics2D g2d)

This code is used to initiate Pac-man.

private void drawMaze(Graphics2D g2d)

This code is used to draw the Maze according to the numbers above.

The code on line 339:

private void initGame() {

lives = 5;

score = 0;

initLevel();

N\_GHOSTS = 6;

currentSpeed = 3;

}

This class is used to initiate the game and starting values, we can increase the lives for Pac-man. By default we have set it to 5. Initial Score is always 0 which can be altered. N\_GHOSTS shows numbers of ghosts spawned in the maze. It can be changed but not recommended. Currentspeed shows speed of the game.

private void continueLevel()

Will continue the level once you collect all the white dots, the game will be reset. The score will not be reset instead it will continue. White dots will spawn again.

class TAdapter extends KeyAdapter

This class is for the main controls, without Pac-man will not move. This class assigns cursor keys to move Pac-man around. Left, right, up and down directions are assigned to the cursor keys.

private void loadImages() {

down = new ImageIcon("C:\\Users\\Vortex\\Pictures\\PacManGame\\down.gif").getImage();

up = new ImageIcon("C:\\Users\\Vortex\\Pictures\\PacManGame\\up.gif").getImage();

left = new ImageIcon("C:\\Users\\Vortex\\Pictures\\PacManGame\\left.gif").getImage();

right = new ImageIcon("C:\\Users\\Vortex\\Pictures\\PacManGame\\right.gif").getImage();

ghost = new ImageIcon("C:\\Users\\Vortex\\Pictures\\PacManGame\\ghost.gif").getImage();

heart = new ImageIcon("C:\\Users\\Vortex\\Pictures\\PacManGame\\left.gif").getImage();

}

Will load images from the library or destination of folders given. The image can be any format

such as (.png and .jpeg). GIFs images can also be used. But for different directions, different images are used. You can replace the image to your likings but its width and heights must be 22x22 other than that it won’t load the images correctly. The dimensions are set according to the maze resolution.

**HERE IS THE FULL CODE OF PAC MAN**

**Pacman.java code:**

package pacman;

import javax.swing.JFrame;

public class Pacman extends JFrame{

public Pacman(){

add(new Model());

}

public static void main(String[] args) {

Pacman pac = new Pacman();

pac.setVisible(true);

pac.setTitle("PACMAN ARCADE GAME");

pac.setSize(380,420);

pac.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

pac.setLocationRelativeTo(null);

}

}

**Model.java code:**

package pacman;

import java.awt.\*;

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;

public class Model extends JPanel implements ActionListener {

private Dimension d;

private final Font smallFont = new Font("Calibri", Font.ITALIC, 22);

private boolean inGame = false;

private boolean dying = false;

private final int BLOCK\_SIZE = 24;

private final int N\_BLOCKS = 15;

private final int SCREEN\_SIZE = N\_BLOCKS \* BLOCK\_SIZE;

private final int MAX\_GHOSTS = 12;

private final int PACMAN\_SPEED = 6;

private int N\_GHOSTS = 6;

private int lives, score;

private int[] dx, dy;

private int[] ghost\_x, ghost\_y, ghost\_dx, ghost\_dy, ghostSpeed;

private Image heart, ghost;

private Image up, down, left, right;

private int pacman\_x, pacman\_y, pacmand\_x, pacmand\_y;

private int req\_dx, req\_dy;

private final short levelData[] = {

19, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 22,

17, 16, 24, 24, 24, 24, 16, 16, 16, 24, 24, 24, 24, 16, 20,

17, 20, 0, 0, 0, 0, 17, 16, 20, 0, 0, 0, 0, 17, 20,

17, 20, 0, 19, 18, 18, 16, 16, 16, 18, 18, 22, 0, 17, 20,

17, 20, 0, 17, 16, 16, 16, 16, 16, 16, 16, 20, 0, 17, 20,

17, 20, 0, 17, 16, 16, 24, 24, 24, 16, 16, 20, 0, 17, 20,

17, 16, 18, 16, 16, 20, 0, 0, 0, 17, 16, 16, 18, 16, 20,

17, 16, 16, 16, 16, 20, 0, 0, 0, 17, 16, 16, 16, 16, 20,

17, 16, 24, 16, 16, 20, 0, 0, 0, 17, 16, 16, 24, 16, 20,

17, 20, 0, 17, 16, 16, 18, 18, 18, 16, 16, 20, 0, 17, 20,

17, 20, 0, 17, 16, 16, 16, 16, 16, 16, 16, 20, 0, 17, 20,

17, 20, 0, 25, 24, 24, 16, 16, 16, 24, 24, 28, 0, 17, 20,

17, 20, 0, 0, 0, 0, 17, 16, 20, 0, 0, 0, 0, 17, 20,

17, 16, 18, 18, 18, 18, 16, 16, 16, 18, 18, 18, 18, 16, 20,

25, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 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 Model() {

loadImages();

initVariables();

addKeyListener(new TAdapter());

setFocusable(true);

initGame();

}

private void loadImages() {

down = new ImageIcon("C:\\Users\\Vortex\\Pictures\\PacManGame\\down.gif").getImage();

up = new ImageIcon("C:\\Users\\Vortex\\Pictures\\PacManGame\\up.gif").getImage();

left = new ImageIcon("C:\\Users\\Vortex\\Pictures\\PacManGame\\left.gif").getImage();

right = new ImageIcon("C:\\Users\\Vortex\\Pictures\\PacManGame\\right.gif").getImage();

ghost = new ImageIcon("C:\\Users\\Vortex\\Pictures\\PacManGame\\ghost.gif").getImage();

heart = new ImageIcon("C:\\Users\\Vortex\\Pictures\\PacManGame\\left.gif").getImage();

}

private void initVariables() {

screenData = new short[N\_BLOCKS \* N\_BLOCKS];

d = new Dimension(400, 400);

ghost\_x = new int[MAX\_GHOSTS];

ghost\_dx = new int[MAX\_GHOSTS];

ghost\_y = new int[MAX\_GHOSTS];

ghost\_dy = new int[MAX\_GHOSTS];

ghostSpeed = new int[MAX\_GHOSTS];

dx = new int[4];

dy = new int[4];

timer = new Timer(40, this);

timer.start();

}

private void playGame(Graphics2D g2d) {

if (dying) {

death();

} else {

movePacman();

drawPacman(g2d);

moveGhosts(g2d);

checkMaze();

}

}

private void showIntroScreen(Graphics2D g2d) {

String start = "PRESS SPACE TO PLAY";

g2d.setColor(new Color(255, 223, 0)); //Score color

g2d.drawString(start, (SCREEN\_SIZE)/4, 150);

}

private void drawScore(Graphics2D g) {

g.setFont(smallFont); //Font

g.setColor(new Color(135, 206, 235)); //Score color

String s = "YOUR SCORE: " + score; //Will show your score

g.drawString(s, SCREEN\_SIZE / 2 + 25, SCREEN\_SIZE + 16); //Direction

for (int i = 0; i < lives; i++) {

g.drawImage(heart, i \* 28 + 8, SCREEN\_SIZE + 1, this); //Direction

}

}

private void checkMaze() {

int i = 0;

boolean finished = true;

while (i < N\_BLOCKS \* N\_BLOCKS && finished) {

if ((screenData[i]) != 0) {

finished = false;

}

i++;

}

if (finished) {

score += 50;

if (N\_GHOSTS < MAX\_GHOSTS) {

N\_GHOSTS++;

}

if (currentSpeed < maxSpeed) {

currentSpeed++;

}

initLevel();

}

}

private void death() {

lives--;

if (lives == 0) {

inGame = false;

}

continueLevel();

}

private void moveGhosts(Graphics2D g2d) {

int pos;

int count;

for (int i = 0; i < N\_GHOSTS; i++) {

if (ghost\_x[i] % BLOCK\_SIZE == 0 && ghost\_y[i] % BLOCK\_SIZE == 0) {

pos = ghost\_x[i] / BLOCK\_SIZE + N\_BLOCKS \* (int) (ghost\_y[i] / BLOCK\_SIZE);

count = 0;

if ((screenData[pos] & 1) == 0 && ghost\_dx[i] != 1) {

dx[count] = -1;

dy[count] = 0;

count++;

}

if ((screenData[pos] & 2) == 0 && ghost\_dy[i] != 1) {

dx[count] = 0;

dy[count] = -1;

count++;

}

if ((screenData[pos] & 4) == 0 && ghost\_dx[i] != -1) {

dx[count] = 1;

dy[count] = 0;

count++;

}

if ((screenData[pos] & 8) == 0 && ghost\_dy[i] != -1) {

dx[count] = 0;

dy[count] = 1;

count++;

}

if (count == 0) {

if ((screenData[pos] & 15) == 15) {

ghost\_dx[i] = 0;

ghost\_dy[i] = 0;

} else {

ghost\_dx[i] = -ghost\_dx[i];

ghost\_dy[i] = -ghost\_dy[i];

}

} else {

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

if (count > 3) {

count = 3;

}

ghost\_dx[i] = dx[count];

ghost\_dy[i] = dy[count];

}

}

ghost\_x[i] = ghost\_x[i] + (ghost\_dx[i] \* ghostSpeed[i]);

ghost\_y[i] = ghost\_y[i] + (ghost\_dy[i] \* ghostSpeed[i]);

drawGhost(g2d, ghost\_x[i] + 1, ghost\_y[i] + 1);

if (pacman\_x > (ghost\_x[i] - 12) && pacman\_x < (ghost\_x[i] + 12)

&& pacman\_y > (ghost\_y[i] - 12) && pacman\_y < (ghost\_y[i] + 12)

&& inGame) {

dying = true;

}

}

}

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

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

}

private void movePacman() {

int pos;

short ch;

if (pacman\_x % BLOCK\_SIZE == 0 && pacman\_y % BLOCK\_SIZE == 0) {

pos = pacman\_x / BLOCK\_SIZE + N\_BLOCKS \* (int) (pacman\_y / BLOCK\_SIZE);

ch = screenData[pos];

if ((ch & 16) != 0) {

screenData[pos] = (short) (ch & 15);

score++;

}

if (req\_dx != 0 || req\_dy != 0) {

if (!((req\_dx == -1 && req\_dy == 0 && (ch & 1) != 0)

|| (req\_dx == 1 && req\_dy == 0 && (ch & 4) != 0)

|| (req\_dx == 0 && req\_dy == -1 && (ch & 2) != 0)

|| (req\_dx == 0 && req\_dy == 1 && (ch & 8) != 0))) {

pacmand\_x = req\_dx;

pacmand\_y = req\_dy;

}

}

// Check for standstill

if ((pacmand\_x == -1 && pacmand\_y == 0 && (ch & 1) != 0)

|| (pacmand\_x == 1 && pacmand\_y == 0 && (ch & 4) != 0)

|| (pacmand\_x == 0 && pacmand\_y == -1 && (ch & 2) != 0)

|| (pacmand\_x == 0 && pacmand\_y == 1 && (ch & 8) != 0)) {

pacmand\_x = 0;

pacmand\_y = 0;

}

}

pacman\_x = pacman\_x + PACMAN\_SPEED \* pacmand\_x;

pacman\_y = pacman\_y + PACMAN\_SPEED \* pacmand\_y;

}

private void drawPacman(Graphics2D g2d) {

if (req\_dx == -1) {

g2d.drawImage(left, pacman\_x + 1, pacman\_y + 1, this);

} else if (req\_dx == 1) {

g2d.drawImage(right, pacman\_x + 1, pacman\_y + 1, this);

} else if (req\_dy == -1) {

g2d.drawImage(up, pacman\_x + 1, pacman\_y + 1, this);

} else {

g2d.drawImage(down, pacman\_x + 1, pacman\_y + 1, this);

}

}

private void drawMaze(Graphics2D g2d) {

short i = 0;

int x, y;

for (y = 0; y < SCREEN\_SIZE; y += BLOCK\_SIZE) {

for (x = 0; x < SCREEN\_SIZE; x += BLOCK\_SIZE) {

g2d.setColor(new Color(192,192,192));

g2d.setStroke(new BasicStroke(5));

if ((levelData[i] == 0)) {

g2d.fillRect(x, y, BLOCK\_SIZE, BLOCK\_SIZE);

}

if ((screenData[i] & 1) != 0) {

g2d.drawLine(x, y, x, y + BLOCK\_SIZE - 1);

}

if ((screenData[i] & 2) != 0) {

g2d.drawLine(x, y, x + BLOCK\_SIZE - 1, y);

}

if ((screenData[i] & 4) != 0) {

g2d.drawLine(x + BLOCK\_SIZE - 1, y, x + BLOCK\_SIZE - 1,

y + BLOCK\_SIZE - 1);

}

if ((screenData[i] & 8) != 0) {

g2d.drawLine(x, y + BLOCK\_SIZE - 1, x + BLOCK\_SIZE - 1,

y + BLOCK\_SIZE - 1);

}

if ((screenData[i] & 16) != 0) {

g2d.setColor(new Color(255,255,255));

g2d.fillOval(x + 10, y + 10, 6, 6);

}

i++;

}

}

}

private void initGame() {

lives = 5;

score = 0;

initLevel();

N\_GHOSTS = 6;

currentSpeed = 3;

}

private void initLevel() {

int i;

for (i = 0; i < N\_BLOCKS \* N\_BLOCKS; i++) {

screenData[i] = levelData[i];

}

continueLevel();

}

private void continueLevel() {

int dx = 1;

int random;

for (int i = 0; i < N\_GHOSTS; i++) {

ghost\_y[i] = 4 \* BLOCK\_SIZE; //start position

ghost\_x[i] = 4 \* BLOCK\_SIZE;

ghost\_dy[i] = 0;

ghost\_dx[i] = dx;

dx = -dx;

random = (int) (Math.random() \* (currentSpeed + 1));

if (random > currentSpeed) {

random = currentSpeed;

}

ghostSpeed[i] = validSpeeds[random];

}

pacman\_x = 7 \* BLOCK\_SIZE;

pacman\_y = 11 \* BLOCK\_SIZE;

pacmand\_x = 0;

pacmand\_y = 0;

req\_dx = 0;

req\_dy = 0;

dying = false;

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g;

g2d.setColor(Color.black);

g2d.fillRect(0, 0, d.width, d.height);

drawMaze(g2d);

drawScore(g2d);

if (inGame) {

playGame(g2d);

} else {

showIntroScreen(g2d);

}

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) {

req\_dx = -1;

req\_dy = 0;

} else if (key == KeyEvent.VK\_RIGHT) {

req\_dx = 1;

req\_dy = 0;

} else if (key == KeyEvent.VK\_UP) {

req\_dx = 0;

req\_dy = -1;

} else if (key == KeyEvent.VK\_DOWN) {

req\_dx = 0;

req\_dy = 1;

} else if (key == KeyEvent.VK\_ESCAPE && timer.isRunning()) {

inGame = false;

}

} else {

if (key == KeyEvent.VK\_SPACE) {

inGame = true;

initGame();

}

}

}

}

@Override

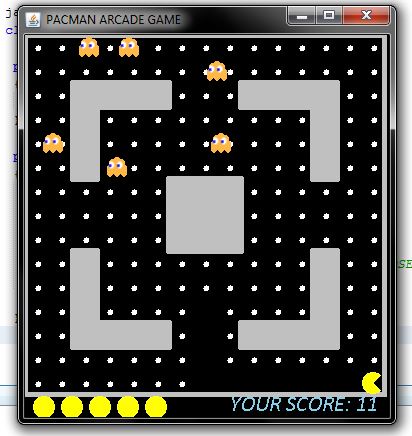
public void actionPerformed(ActionEvent e) {

repaint();

}

}

**Output/Gameplay:**



**Pictures used:**

* 