package com.zetcode;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.FontMetrics;

import java.awt.Graphics;

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;

public class Board extends JPanel implements ActionListener {

private final int B\_WIDTH = 300;

private final int B\_HEIGHT = 300;

private final int DOT\_SIZE = 10;

private final int ALL\_DOTS = 900;

private final int RAND\_POS = 29;

private final int DELAY = 140;

private final int x[] = new int[ALL\_DOTS];

private final int y[] = new int[ALL\_DOTS];

private int dots;

private int apple\_x;

private int apple\_y;

private boolean leftDirection = false;

private boolean rightDirection = true;

private boolean upDirection = false;

private boolean downDirection = false;

private boolean inGame = true;

private Timer timer;

private Image ball;

private Image apple;

private Image head;

public Board() {

initBoard();

}

private void initBoard() {

addKeyListener(new TAdapter());

setBackground(Color.black);

setFocusable(true);

setPreferredSize(new Dimension(B\_WIDTH, B\_HEIGHT));

loadImages();

initGame();

}

private void loadImages() {

ImageIcon iid = new ImageIcon("src/resources/dot.png");

ball = iid.getImage();

ImageIcon iia = new ImageIcon("src/resources/apple.png");

apple = iia.getImage();

ImageIcon iih = new ImageIcon("src/resources/head.png");

head = iih.getImage();

}

private void initGame() {

dots = 3;

for (int z = 0; z < dots; z++) {

x[z] = 50 - z \* 10;

y[z] = 50;

}

locateApple();

timer = new Timer(DELAY, this);

timer.start();

}

@Override

public void paintComponent(Graphics g) {

super.paintComponent(g);

doDrawing(g);

}

private void doDrawing(Graphics g) {

if (inGame) {

g.drawImage(apple, apple\_x, apple\_y, this);

for (int z = 0; z < dots; z++) {

if (z == 0) {

g.drawImage(head, x[z], y[z], this);

} else {

g.drawImage(ball, x[z], y[z], this);

}

}

Toolkit.getDefaultToolkit().sync();

} else {

gameOver(g);

}

}

private void gameOver(Graphics g) {

String msg = "Game Over";

Font small = new Font("Helvetica", Font.BOLD, 14);

FontMetrics metr = getFontMetrics(small);

g.setColor(Color.white);

g.setFont(small);

g.drawString(msg, (B\_WIDTH - metr.stringWidth(msg)) / 2, B\_HEIGHT / 2);

}

private void checkApple() {

if ((x[0] == apple\_x) && (y[0] == apple\_y)) {

dots++;

locateApple();

}

}

private void move() {

for (int z = dots; z > 0; z--) {

x[z] = x[(z - 1)];

y[z] = y[(z - 1)];

}

if (leftDirection) {

x[0] -= DOT\_SIZE;

}

if (rightDirection) {

x[0] += DOT\_SIZE;

}

if (upDirection) {

y[0] -= DOT\_SIZE;

}

if (downDirection) {

y[0] += DOT\_SIZE;

}

}

private void checkCollision() {

for (int z = dots; z > 0; z--) {

if ((z > 4) && (x[0] == x[z]) && (y[0] == y[z])) {

inGame = false;

}

}

if (y[0] >= B\_HEIGHT) {

inGame = false;

}

if (y[0] < 0) {

inGame = false;

}

if (x[0] >= B\_WIDTH) {

inGame = false;

}

if (x[0] < 0) {

inGame = false;

}

if (!inGame) {

timer.stop();

}

}

private void locateApple() {

int r = (int) (Math.random() \* RAND\_POS);

apple\_x = ((r \* DOT\_SIZE));

r = (int) (Math.random() \* RAND\_POS);

apple\_y = ((r \* DOT\_SIZE));

}

@Override

public void actionPerformed(ActionEvent e) {

if (inGame) {

checkApple();

checkCollision();

move();

}

repaint();

}

private class TAdapter extends KeyAdapter {

@Override

public void keyPressed(KeyEvent e) {

int key = e.getKeyCode();

if ((key == KeyEvent.VK\_LEFT) && (!rightDirection)) {

leftDirection = true;

upDirection = false;

downDirection = false;

}

if ((key == KeyEvent.VK\_RIGHT) && (!leftDirection)) {

rightDirection = true;

upDirection = false;

downDirection = false;

}

if ((key == KeyEvent.VK\_UP) && (!downDirection)) {

upDirection = true;

rightDirection = false;

leftDirection = false;

}

if ((key == KeyEvent.VK\_DOWN) && (!upDirection)) {

downDirection = true;

rightDirection = false;

leftDirection = false;

}

}

}

}