**Snake Game Main**

public class SnakeGame {

public static void main(String[] args) {

new GameFrame();

// **TODO** Auto-generated method stub

}

}

**GAME FRAME**

import java.awt.CardLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

public class GameFrame extends JFrame {

GamePanel gamepanel;

JPanel contentPane;

CardLayout cardLayout = new CardLayout();

GameFrame(){

this.add(new GamePanel());

this.setTitle("Snake");

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setResizable(true);

this.pack();

this.setVisible(true);

this.setLocationRelativeTo(null);

}

public void playAgain() {

cardLayout.next(contentPane);

gamepanel.restart();

gamepanel.requestFocusInWindow();

}

}

**GamePanel**

import java.awt.\*;

import java.awt.event.\*;

import java.util.Random;

import javax.swing.\*;

public class GamePanel extends JPanel implements ActionListener {

static final int SCREEN\_WIDTH = 1250;

static final int SCREEN\_HEIGHT = 600;

static final int UNIT\_SIZE = 25;

static final int GAME\_UNITS = (SCREEN\_WIDTH\*SCREEN\_HEIGHT)/(UNIT\_SIZE\*UNIT\_SIZE);

static final int DELAY = 50;

final int x[] = new int[GAME\_UNITS];

final int y[] = new int[GAME\_UNITS];

int bodyParts = 6;

int applesEaten;

int appleX;

int appleY;

char direction = 'R';

boolean running = false;

Timer timer;

Random random;

static boolean gameOn = false;

JFrame frame;

GamePanel(){

random = new Random();

this.setPreferredSize(new Dimension(SCREEN\_WIDTH,SCREEN\_HEIGHT));

this.setBackground(Color.black);

this.setFocusable(true);

this.addKeyListener(new MyKeyAdapter());

startGame();

}

public void startGame() {

newApple();

running = true;

timer = new Timer(DELAY,this);

timer.start();

}

public void pause() {

GamePanel.gameOn = true;

timer.stop();

}

public void resume() {

GamePanel.gameOn = false;

timer.start();

}

public void restart() {

bodyParts = 6;

direction = 'R';

applesEaten = 0;

for(int i = bodyParts; i>=0; i--) {

x[i] = 0;

y[i] = 0;

}

startGame();

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

draw(g);

}

public void draw(Graphics g) {

if (running) {

/\*

for( int i=0;i<SCREEN\_HEIGHT/UNIT\_SIZE; i++){

g.drawLine(i\*UNIT\_SIZE ,0, i\*UNIT\_SIZE, SCREEN\_HEIGHT);

g.drawLine(0 ,i\*UNIT\_SIZE, SCREEN\_WIDTH, i\* UNIT\_SIZE);

}

\*/

//borders

if(applesEaten % 5 == 0&& applesEaten != 0 ) {

g.setColor(new Color(random.nextInt(255),random.nextInt(255),random.nextInt(255)));

g.fillRect(0, 0, 3, 600);

g.fillRect(0, 0, 1250, 4);

g.fillRect(1248, 0, 4, 600);

g.fillRect(0,599,1250,4);

}else {

g.setColor(Color.yellow);

g.fillRect(0, 0, 3, 600);

g.fillRect(0, 0, 1250, 4);

g.fillRect(1248, 0, 4, 600);

g.fillRect(0,599,1250,4);

}

g.setColor(Color.white);

g.fillOval(appleX, appleY, UNIT\_SIZE, UNIT\_SIZE);

timer.setDelay(75);

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

if(applesEaten % 5 == 0 && applesEaten != 0) {

timer.setDelay(45);

g.setColor(new Color(random.nextInt(255),random.nextInt(255),random.nextInt(255)));

g.fillRect(x[i], y[i],UNIT\_SIZE, UNIT\_SIZE);

}else if(i == 0) {

g.setColor(Color.GREEN);

//g.setColor(new Color(45,180,0));

g.fillRect(x[i], y[i],UNIT\_SIZE, UNIT\_SIZE);

}else {

g.setColor(new Color(45,180,0));

g.fillRect(x[i], y[i],UNIT\_SIZE, UNIT\_SIZE);

}

}

if(applesEaten % 5 == 0 && applesEaten != 0) {

g.setColor(new Color(random.nextInt(255),random.nextInt(255),random.nextInt(255)));

g.setFont(new Font("Microsoft YaHei Light", Font.BOLD,40));

FontMetrics metrics = getFontMetrics(g.getFont());

g.drawString("Score : " +applesEaten,(SCREEN\_WIDTH - metrics.stringWidth("Score: "+applesEaten))/2,g.getFont().getSize());

}

else {

g.setColor(Color.red);

g.setFont(new Font("Microsoft YaHei Light", Font.BOLD,40));

FontMetrics metrics = getFontMetrics(g.getFont());

g.drawString("Score : " +applesEaten,(SCREEN\_WIDTH - metrics.stringWidth("Score: "+applesEaten))/2,g.getFont().getSize());

}

}

else {

gameOver(g);

}

}

public void newApple() {

appleX = random.nextInt((int)(SCREEN\_WIDTH/UNIT\_SIZE))\*UNIT\_SIZE;

appleY = random.nextInt((int)(SCREEN\_HEIGHT/UNIT\_SIZE))\*UNIT\_SIZE;

}

public void move() {

for(int i = bodyParts; i>0;i--) {

x[i] = x[i-1];

y[i] = y[i-1];

}

switch(direction) {

case 'U':

y[0] = y[0] - UNIT\_SIZE;

break;

case 'D':

y[0] = y[0] + UNIT\_SIZE;

break;

case 'L':

x[0] = x[0] - UNIT\_SIZE;

break;

case 'R':

x[0] = x[0] + UNIT\_SIZE;

break;

}

}

public void checkApple() {

if((x[0] == appleX) && (y[0] == appleY)) {

bodyParts++;

applesEaten++;

newApple();

}

}

public void checkCollisions() {

//check if head collides with border

for(int i = bodyParts; i>0; i--) {

if((x[0] == x[i]) && (y[0] == y[i])) {

running = false;

}

}

//check if head touches left border

if(x[0] < 0) {

running = false;

}

//check if head touches right border

if(x[0] > SCREEN\_WIDTH) {

running = false;

}

//check if head touches top border

if(y[0] < 0) {

running = false;

}

//check if head touches bottom border

if(y[0] > SCREEN\_HEIGHT) {

running = false;

}

if(!running) {

timer.stop();

}

}

public void gameOver(Graphics g) {

//Score

g.setColor(Color.RED);

g.setFont(new Font("Microsoft YaHei Light", Font.BOLD,40));

FontMetrics metrics1 = getFontMetrics(g.getFont());

g.drawString("Score : " +applesEaten,(SCREEN\_WIDTH - metrics1.stringWidth("Score: "+applesEaten))/2,g.getFont().getSize());

//GameOver Text

g.setColor(new Color(random.nextInt(255),random.nextInt(255),random.nextInt(255)));

g.setFont(new Font("Microsoft YaHei Light", Font.BOLD,75));

FontMetrics metrics2 = getFontMetrics(g.getFont());

g.drawString("Game Over" ,(SCREEN\_WIDTH - metrics2.stringWidth("Game Over"))/2,SCREEN\_HEIGHT/3);

//Press again to Restart Text

g.setColor(new Color(random.nextInt(255),random.nextInt(255),random.nextInt(255)));

g.setFont(new Font("Microsoft YaHei Light", Font.BOLD,25));

FontMetrics metrics21 = getFontMetrics(g.getFont());

g.drawString("Press Enter to Restart :)" ,(SCREEN\_WIDTH - metrics21.stringWidth("Press Enter to Restart :)"))/2,SCREEN\_HEIGHT/2);

}

@Override

public void actionPerformed(ActionEvent e) {

if(running) {

move();

checkApple();

checkCollisions();

}

repaint();

}

public class MyKeyAdapter extends KeyAdapter{

@Override

public void keyPressed(KeyEvent e) {

switch(e.getKeyCode()) {

case KeyEvent.VK\_LEFT:

if(direction != 'R') {

direction = 'L';

}

break;

case KeyEvent.VK\_RIGHT:

if(direction != 'L') {

direction = 'R';

}

break;

case KeyEvent.VK\_UP:

if(direction != 'D') {

direction = 'U';

}

break;

case KeyEvent.VK\_DOWN:

if(direction != 'U') {

direction = 'D';

}

break;

case KeyEvent.VK\_ENTER:

restart();

repaint();

case KeyEvent.VK\_SPACE:

if(GamePanel.gameOn) {

resume();

}else {

pause();

}break;

}

}

@Override

public String toString() {

return "MyKeyAdapter []";

}

}

}