import javax.swing.\*; snakepanel.java

import java.awt.\*;

import java.awt.event.\*;

import java.util.Arrays;

import java.util.Random;

public class SnakePanel extends JPanel implements ActionListener{

    static final int S\_Width=600;

    static final int S\_Height=600;

    static final int Game\_unit\_size=50;

    Timer timer;

    Random random;

    int foodEaten;

    int foodX;

    int foodY;

    int bodLength = 2;

    boolean game\_flag = false;

    char dir = 'R';

    static final int DELAY = 160;

    static final int G\_Size=(S\_Width\*S\_Height)/(Game\_unit\_size\*Game\_unit\_size);

    final int[] x\_snake=new int[G\_Size];

    final int[] y\_snake=new int[G\_Size];

    SnakePanel(){

        this.setPreferredSize(new Dimension(S\_Width,S\_Height));

        this.setBackground(Color.black);

        this.setFocusable(true);

        this.addKeyListener(new MyKey());

        this.setVisible(true);

        random = new Random();

        Game\_start();

    }

    public void Game\_start() {

        newFoodPosition();

        game\_flag=true;

        timer=new Timer(DELAY,this);

        timer.start();

    }

    public void paintComponent(Graphics graphic) {

        super.paintComponent(graphic);

        draw(graphic);

    }

    public void draw(Graphics graphic) {

        if(game\_flag){

            graphic.setColor(Color.yellow);

            graphic.fillRect(foodX, foodY,Game\_unit\_size,Game\_unit\_size);

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

                if(i==0){

                    graphic.setColor(Color.green);

//                    graphic.fillRect(x\_snake[i],y\_snake[i],Game\_unit\_size,Game\_unit\_size);

//                    graphic.drawRoundRect(x\_snake[i],y\_snake[i],Game\_unit\_size,Game\_unit\_size,50,30);

                    graphic.fillOval(x\_snake[i],y\_snake[i],Game\_unit\_size,Game\_unit\_size);

                }

                else{

                    graphic.setColor(new Color(50,180,0));

//                    graphic.fillRect(x\_snake[i],y\_snake[i],Game\_unit\_size,Game\_unit\_size);

//                    graphic.drawRoundRect(x\_snake[i],y\_snake[i],Game\_unit\_size,Game\_unit\_size,50,30);

                    graphic.fillOval(x\_snake[i],y\_snake[i],Game\_unit\_size,Game\_unit\_size);

                }

            }

            graphic.setColor(Color.blue);

            graphic.setFont(new Font("Ink Free",Font.BOLD,40));

            FontMetrics font\_me=getFontMetrics(graphic.getFont());

            graphic.drawString("Score:"+foodEaten,(S\_Width-font\_me.stringWidth("Score:"+foodEaten))/2,graphic.getFont().getSize());

        }

        else{

            gameOver(graphic);

        }

    }

    public void move() {

        for(int i=bodLength;i>0;i--){

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

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

        }

        switch (dir) {

            case 'U':

                y\_snake[0]=y\_snake[0]-Game\_unit\_size;

                break;

            case 'L':

                x\_snake[0] = x\_snake[0] - Game\_unit\_size;

                break;

            case 'D':

                y\_snake[0] = y\_snake[0] + Game\_unit\_size;

                break;

            case 'R':

                x\_snake[0] = x\_snake[0] + Game\_unit\_size;

                break;

        }

    }

    public void newFoodPosition() {

        foodX=random.nextInt((int)(S\_Width/Game\_unit\_size))\*Game\_unit\_size;

        foodY=random.nextInt((int)(S\_Height/Game\_unit\_size))\*Game\_unit\_size;

    }

    public void food\_EatenOrNot() {

        if((x\_snake[0]==foodX)&&(y\_snake[0]==foodY)){

            bodLength++;

            foodEaten++;

            newFoodPosition();

        }

    }

    public void checkHit() {

        for (int i=bodLength;i>0;i--)

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

        {game\_flag=false;}}

        if(x\_snake[0]<0)

        {game\_flag=false;}

        if(x\_snake[0]>S\_Width)

        { game\_flag=false;}

        if (y\_snake[0] < 0)

        {    game\_flag = false;}

        if (y\_snake[0] > S\_Height)

        {    game\_flag = false;}

        if(!game\_flag)

        {  timer.stop();}

    }

    public void gameOver(Graphics graphic) {// When ever game is over this function will be called.

        graphic.setColor(Color.red);

        graphic.setFont(new Font("Ink Free", Font.BOLD, 40));

        FontMetrics font\_me = getFontMetrics(graphic.getFont());

        graphic.drawString("Score:" + foodEaten, (S\_Width - font\_me.stringWidth("Score:" + foodEaten)) / 2,

                graphic.getFont().getSize());

        graphic.setColor(Color.red);

        graphic.setFont(new Font("Ink Free", Font.BOLD, 75));

        FontMetrics font\_me2 = getFontMetrics(graphic.getFont());

        graphic.drawString("Game Over", (S\_Width - font\_me2.stringWidth("Game Over")) / 2,

                S\_Height/2);

        graphic.setColor(Color.red);

        graphic.setFont(new Font("Ink Free", Font.BOLD, 40));

        FontMetrics font\_me3 = getFontMetrics(graphic.getFont());

        graphic.drawString("Press R to Replay", (S\_Width - font\_me3.stringWidth("Press R to Replay")) / 2, S\_Height / 2-150);

    }

    public class MyKey extends KeyAdapter{

        @Override

        public void keyPressed(KeyEvent e) {

            switch (e.getKeyCode()) {

                case KeyEvent.VK\_LEFT:

                    if(dir!='R'){

                        dir='L';

                    }

                    break;

                case KeyEvent.VK\_UP:

                    if (dir != 'D') {

                        dir = 'U';

                    }

                    break;

                case KeyEvent.VK\_RIGHT:

                    if (dir != 'L') {

                        dir = 'R';

                    }

                    break;

                case KeyEvent.VK\_DOWN:

                    if (dir != 'U') {

                        dir = 'D';

                    }

                    break;

                case KeyEvent.VK\_R:

                    if(!game\_flag){

                        foodEaten=0;

                        bodLength=2;

                        dir='R';

                        Arrays.fill(x\_snake,0);

                        Arrays.fill(y\_snake,0);

                        Game\_start();

                    }

                    break;

            }

        }

    }

    @Override

    public void actionPerformed(ActionEvent arg0) {

        if (game\_flag) {

            move();

            food\_EatenOrNot();

            checkHit();

        }

        repaint();

    }

}

import javax.swing.JFrame; snakeframe.java

import java.awt.\*;

public class  SnakeFrame extends JFrame {

    SnakeFrame(){

        this.add(new SnakePanel());

        this.setTitle("Snake");

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

        this.setResizable(false);

        this.pack();

        this.setVisible(true);

        this.setLocationRelativeTo(null);

    }

}

public class SnakeGame { snakegame.java

    public static void main(String[] args) throws Exception {

        new SnakeFrame();

    }

}