|  |
| --- |
| BigBasti |
| Sneaky game |
| 10/08/2014 |

|  |
| --- |
| Source: http://blog.bigbasti.com/java-snake-nachprogrammieren-demo/  Big Basti |

Game

**package** game;

**import** java.awt.\*;

**public** **class** Game {

**public** **static** Logwriter *log* = **new** Logwriter();

**public** **static** Snake *snake*;

**public** **static** WinMain *frmMain*;

**public** **static** Point *aim*;

**public** **static** **double** *playtime* = 0;

**public** **static** **int** *punkte* = 0;

**public** **static** **void** main(String[] args){

*log*.write("Game has started...");

*snake* = **new** Snake();

*frmMain* = **new** WinMain("Sneaky by Sebastian Gross");

*log*.write("main-window is created...");

*newAim*();

*runGame*();

}

**public** **static** **void** newAim(){

Point p;

// create a new point on the field

**while**(**true**){

p = **new** Point((**int**)((Math.*random*()\*20)+0.5),(**int**)((Math.*random*()\*20)+0.5));

//Make sure the aim is in the field

//x & y must be bigger than 0!

**if**(p.x >= 1 && p.y >= 1){

**break**;

}

}

*punkte* += 100; //if you reach the aim you get 100 points

*aim* = p;

}

**public** **static** **void** runGame(){

**int** secs = 0; // Variable to mesure time

**while**(**true**){

// start an unending loop

// which update the screen picture every 100 milliseconds (standard) and

// moves the snake one “box/point/pixel” further

//frmMain.setTitle("S:"+snake.pos.x+snake.pos.y+" - "+aim.x+aim.y);

**if**(*snake*.moveSnake() == **false**){

// stop the program if the game is over

**break**;

}

*frmMain*.pG.paintComponent(*frmMain*.pG.getGraphics());

// update playtime (optional)

// has to be adjusted if the „sleep()“ configurations have been changed

secs++;

**if** (secs == 10){

// one second has gone

*playtime*++;

// reverse the counter

secs = 0;

*punkte*++;

}

**try**{

Thread.*sleep*(100); // standard-configuration: 100

}**catch**(Exception ex){*log*.write(ex.getMessage());}

}

}

}

Logwriter

**package** game;

//Diese Klasse kümmert sich nur um die Ausgabe von Statusinformationen

//auf der Konsole zur Kontrolle und Debugging!

**public** **class** Logwriter {

**public** **void** write(String msg){

System.***out***.println(msg);

}

}

PointA

**package** game;

**import** java.awt.\*;

//Diese Klasse hat ein "A" am Ende, da der name "Point" reserviert ist!

**public** **class** PointA {

**public** Point pos;

**public** **boolean** head; //Gibt an, ob dieser Punkt der Kopf der Schlange ist

**public** **boolean** tail; //Gibt an, ob dieser Punkt das Ende der schlange ist

**private** Logwriter log = **new** Logwriter();

**public** PointA(Point p, **boolean** head, **boolean** tail){

**this**.pos = p;

**this**.head = head;

**this**.tail = tail;

log.write("Neuen Punkt erstellt...");

}

**public** Point getPos(){

**return** pos;

}

**public** **void** setPoint(Point p){

**this**.pos = p;

}

**public** **void** paintPoint(Graphics g){

**if** (head == **true**){

g.setColor(Color.***red***); //Den Kopf der Schlage Rot färben

}**else**{

g.setColor(Color.***black***);

}

g.fillRect((pos.x+1)\*5, (pos.y+1)\*5, 5, 5);

//log.write("Punkt zeichnen...");

}

}

Snake

**package** game;

**import** java.awt.\*;

**import** java.util.\*;

**public** **class** Snake {

**public** Point pos; // current position of the snake’s head

**public** String dir; // direction in which the snake currently moves

(left, right, top, bottom)

**public** **boolean** newPoint = **false**;

**public** LinkedList<PointA> points = **new** LinkedList<PointA>(); //list with all snake-points

**private** Logwriter log = **new** Logwriter();

**public** Snake(){

// When you create a new snake, the snake hasn’t points yet, that means POINTS is empthy

// That the user see at least something a snake gets two points are created

// one of these two points is marke das the head of the snake

// the snake starts at the top left corner at position 2,1

=> 2,1 is the position of the head and 1,1 the body

// 0,0 - 0,[field lenghts] is reservated for the boarder

// the start direction is right

pos = **new** Point(3,1); // start-position of the head

dir = "right"; // start-direction

// Create the start-snake-body, consist of 3 points: head, bodypoint, tale points.add(**new** PointA(**new** Point(3,1),**true**,**false**)); // create snake head

points.add(**new** PointA(**new** Point(2,1),**false**,**false**)); // create snake bodypoint

points.add(**new** PointA(**new** Point(1,1),**false**,**true**)); // ceate snake talepoint

log.write("Snake has ben created...");

}

**public** **void** addNewPoint(PointA p){

points.add(p);

}

**public** **void** paintSneak(Graphics g){

//log.write("Draw the points new ... ?");

**for**(PointA p : points){ // Draw all snake points new

p.paintPoint(g);

}

}

**public** **void** setDir(String dir){

// Check if the user tries to change direction in the opposite direction (is not possible)

// This if-chain can be „commented-out“ to extend the game with another mode: become instant

**if**(**this**.dir.equals("right") && dir.equals("left")){

log.write("Change of direction not possible...");

**return**;

}**else** **if**(**this**.dir.equals("left") && dir.equals("right")){

log.write("Change of direction not possible...");

**return**;

}**else** **if**(**this**.dir.equals("top") && dir.equals("bottom")){

log.write("Change of direction not possible...");

**return**;

}**else** **if**(**this**.dir.equals("bottom") && dir.equals("top")){

log.write("Change of direction not possible...");

**return**;

}

log.write("Change moving direction to "+dir+" ...");

**this**.dir = dir; // changing the moving direction of the snake

}

**public** **boolean** moveSnake(){

// this method returnes true if the snake could move on

// or false if the snake hit the border or hit itself

/\*

\* Two possibitlites to programm that snake movement

\* 1:

\* Wenn die Schlange weiterfahren soll nimmt man einfach jeden Punkt und

\* ersetzt dessen koordinaten mit denen des Vordermanns

\* O--X--X--X--X--X--X <= Das soll die schlange sein

\* ^<|^<|^<|^<|^<|^<|^<| => Alles rückt einen Schritt weiter

\*

\* 2:

\* Take the last point and put it at the beginning of the snake:

\* O--X--X--X--X--X--X <= snake

\* ^------------------<| => only the last point moves!

\*

\* Second idea is better than the first

\*/

// Check if the snake can move into the claimed direction

Point nPos = **null**;

**if**(**this**.dir.equals("left")){

nPos = **new** Point(**this**.pos.x-1, **this**.pos.y);

}**else** **if**(**this**.dir.equals("top")){

nPos = **new** Point(**this**.pos.x, **this**.pos.y-1);

}**else** **if**(**this**.dir.equals("bottom")){

nPos = **new** Point(**this**.pos.x, **this**.pos.y+1);

}**else** **if**(**this**.dir.equals("right")){

nPos = **new** Point(**this**.pos.x+1, **this**.pos.y);

}

**if**(nPos.x <= 0 || nPos.x >= 40 || nPos.y <= 0 || nPos.y >= 40){

// if field boarder is reached

**return** **false**;

}**else**{

**for**(PointA p : points){

**if**(p.pos.equals(nPos)){ // snake touches itself

log.write("Toched boarder, Game Over...");

**return** **false**;

}

}

// if there is space left

// mark the current head not to be a head anymore „non-head-body-point“

**for**(PointA p : points){

**if**(p.head == **true**){

p.head = **false**;

}

}

// put the last point in front of the head

**for**(PointA p : points){

**if**(p.tail == **true**){

p.tail = **false**;

p.head = **true**;

p.setPoint(nPos);

PointA nP = p;

points.remove(p);

points.add(0, nP);

}

}

**this**.pos = nPos;

points.get(points.size()-1).tail = **true**;

}

// check if the „feed“ has been found and make the snake one point longer

**if**(nPos.equals(Game.*aim*)){

log.write("Futter gefunden, wachsen...");

// Have a look at the current direction of the snake

**if**(**this**.dir.equals("left")){

**for**(PointA p : points){

// move the head forward in moving direction

**if**(p.head == **true**){

p.setPoint(**new** Point(**this**.pos.x-1,**this**.pos.y));

}

}

// put the new point right behind the head

**this**.pos = **new** Point(**this**.pos.x-1,**this**.pos.y);

points.add(1,**new** PointA(**new** Point(0,0),**false**,**false**));

}**else** **if**(**this**.dir.equals("top")){

**for**(PointA p : points){

**if**(p.head == **true**){

p.setPoint(**new** Point(**this**.pos.x,**this**.pos.y-1));

}

}

**this**.pos = **new** Point(**this**.pos.x,**this**.pos.y-1);

points.add(1,**new** PointA(**new** Point(0,0),**false**,**false**));

}**else** **if**(**this**.dir.equals("bottom")){

**for**(PointA p : points){

**if**(p.head == **true**){

p.setPoint(**new** Point(**this**.pos.x,**this**.pos.y+1));

}

}

**this**.pos = **new** Point(**this**.pos.x,**this**.pos.y+1);

points.add(1,**new** PointA(**new** Point(0,0),**false**,**false**));

}**else** **if**(**this**.dir.equals("right")){

**for**(PointA p : points){

**if**(p.head == **true**){

p.setPoint(**new** Point(**this**.pos.x+1,**this**.pos.y));

}

}

**this**.pos = **new** Point(**this**.pos.x+1,**this**.pos.y);

points.add(1,**new** PointA(**new** Point(0,0),**false**,**false**));

}

// make a new aim point

Game.*newAim*();

}

**return** **true**;

}

}

WinMain

**package** game;

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.swing.\*;

**public** **class** WinMain **extends** JFrame{

**public** gPanel pG; // panel for the game process

**public** JPanel pE; // panel for the end of the game

**public** WinMain(String title){

// Do preperations for the main-window

Game.*log*.write("creating the main-window...");

**this**.setTitle(title);

**this**.setSize(230,270); // it supposed tob e 200,200.. but this measures don’t

fit into pixel with windows.. a part get cut away!

**this**.setLayout(**null**);

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

// initialize the panels

Game.*log*.write("creat game-panel...");

**this**.pG = **new** gPanel();

**this**.pG.setBounds(0, 0, 250, 250);

**this**.pG.setBackground(Color.***black***);

// catch keyboard-events

**this**.addKeyListener(**new** KeyListener(){

//@Override

**public** **void** keyPressed(KeyEvent e) {}

//@Override

**public** **void** keyReleased(KeyEvent e) {

// react on keystrokes

**if**(e.getKeyCode() == KeyEvent.***VK\_RIGHT***){

Game.*snake*.setDir("right");

}**else** **if**(e.getKeyCode() == KeyEvent.***VK\_LEFT***){

Game.*snake*.setDir("left");

}**else** **if**(e.getKeyCode() == KeyEvent.***VK\_UP***){

Game.*snake*.setDir("top");

}**else** **if**(e.getKeyCode() == KeyEvent.***VK\_DOWN***){

Game.*snake*.setDir("bottom");

}

}

//@Override

**public** **void** keyTyped(KeyEvent e) {}

});

**this**.add(pG);

**this**.setVisible(**true**);

}

**public** **class** gPanel **extends** JPanel{

@Override **public** **void** paintComponent(Graphics gr){

//Game.log.write(" draw game-panel...");

Graphics2D g2d = (Graphics2D) gr;

g2d.setColor(Color.***white***);

g2d.fillRect(0, 0, 250, 250);

g2d.setColor(Color.***black***);

g2d.drawRect(10, 10, 195, 195);

g2d.setColor(Color.***green***);

g2d.fillRect((Game.*aim*.x+1)\*5, (Game.*aim*.y+1)\*5, 5, 5);

g2d.setColor(Color.***black***);

g2d.drawString("Punkte: "+Game.*punkte*+" - Zeit: "+Game.*playtime*, 10, 220);

Game.*snake*.paintSneak(g2d);

}

}

}