package tankwar;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.io.File;

import java.io.FileInputStream;

import java.util.ArrayList;

import java.util.Random;

import javax.swing.BorderFactory;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JPanel;

import tankwar.Tank.Direction;

import barrier.EnemyBorn;

import barrier.Gold;

import barrier.Home;

import barrier.Iron;

import barrier.SelfBorn;

import barrier.Wall;

public class TankWar implements KeyListener {

static boolean TIMEOUT = false;

private JFrame f;

private JPanel gamePanel;

private PanelShow messgePanel;

private myPanel p;

private Tank myTank;

public static final int AREA\_WIDTH = 830;

public static final int AREA\_HEIGHT = 800;

private ArrayList<Missle> missles = new ArrayList<Missle>();

private ArrayList<Tank> allTanks = new ArrayList<Tank>();

private ArrayList<Boom> booms = new ArrayList<Boom>();

private ArrayList<Wall> walls = new ArrayList<Wall>();

private ArrayList<Iron> irons = new ArrayList<Iron>();

private ArrayList<Gold> golds = new ArrayList<Gold>();

private ArrayList<Item> items = new ArrayList<Item>();

private ArrayList<EnemyBorn> enemyBorns = new ArrayList<EnemyBorn>();

private SelfBorn selfBorn;

private Home home;

private Tank enemyTank;

private Random r;

private ImageIcon backGround;

private final String map;

private int tankMax;

private boolean over = false;

private static int selfMax = 3;

private boolean win;

private boolean flash = false;

private TankWar tw = this;

static int SCORE = 0;

private final JFrame mainF;

private int style;

public TankWar(String map, int tankMax, JFrame mainF, int style)

throws Exception {

this.map = map;

this.tankMax = tankMax;

this.mainF = mainF;

this.style = style;

init();

}

private void init() {

f = new JFrame("坦克大战 V3.0");

gamePanel = new JPanel(null);

p = new myPanel();

p.setBackground(Color.WHITE);

r = new Random();

messgePanel = new PanelShow();

initMap(new File("map/" + map));

try {

myTank = new Tank(selfBorn.getX(), selfBorn.getY(), true, allTanks,

walls, irons, golds, missles, home, booms, style);

} catch (Exception e1) {

}

myTank.setDir(Direction.U);

allTanks.add(myTank);

addTank();

try {

backGround = new ImageIcon(

TankWar.class.getResource("/pic/whiteback.jpg"));

} catch (Exception e) {

}

p.setBorder(BorderFactory.createEtchedBorder(Color.BLACK, Color.WHITE));

p.setSize(AREA\_WIDTH, AREA\_HEIGHT);

messgePanel.setBounds(AREA\_WIDTH, 0, 200, AREA\_HEIGHT);

gamePanel.add(messgePanel);

gamePanel.add(p);

f.add(gamePanel);

f.setBounds(0, 0, AREA\_WIDTH + 200, AREA\_HEIGHT);

f.setDefaultCloseOperation(3);

f.setResizable(true);

f.setFocusable(true);

f.addKeyListener(this);

f.setVisible(true);

new Thread(new Runnable() {

public void run() {

while (!over) {

if (!myTank.isLive()) {

selfMax--;

if (selfMax < 0) {

f.removeKeyListener(tw);

over = true;

win = false;

break;

} else {

myTank.setLevel(1);

myTank.setX(selfBorn.getX());

myTank.setY(selfBorn.getY());

myTank.setDir(Direction.U);

myTank.setHp(50);

myTank.setLive(true);

}

}

if (tankMax <= 0 && allTanks.size() == 1) {

f.removeKeyListener(tw);

over = true;

win = true;

}

if (!home.isLive()) {

f.removeKeyListener(tw);

over = true;

win = false;

}

p.repaint();

myTank.move();

for (int i = 1; i < allTanks.size(); i++) {

allTanks.get(i).move();

allTanks.get(i).setNoFire(myTank.getNoFire() + 1);

// if(allTanks.get(i).getX()%5==0&&allTanks.get(i).getY()%5==0)

aI(allTanks.get(i));

}

if (allTanks.size() <= enemyBorns.size() + 1)

addTank();

myTank.setNoFire(myTank.getNoFire() + 1);

messgePanel.setEnemyCount(tankMax);

messgePanel.setSelfCount(selfMax);

messgePanel.setScore(SCORE);

if (SCORE % 500 == 0) {

SCORE += 100;

Item item = new Item(allTanks, booms, irons, home);

items.add(item);

item.start();

}

try {

Thread.sleep(30);

} catch (InterruptedException e) {

}

}

over();

}

}).start();

}

private class myPanel extends JPanel {

/\*\*

\*

\*/

private static final long serialVersionUID = 4408440723797225328L;

@Override

public void paintComponent(Graphics g) {

super.paintComponent(g);

g.drawImage(backGround.getImage(), 0, 0, null);

for (int j = 0; j < walls.size(); j++) {

walls.get(j).draw(g);

}

for (int j = 0; j < irons.size(); j++) {

irons.get(j).draw(g);

}

for (int j = 0; j < golds.size(); j++) {

golds.get(j).draw(g);

}

for (int j = 0; j < enemyBorns.size(); j++) {

enemyBorns.get(j).draw(g);

}

home.draw(g);

selfBorn.draw(g);

for (int j = 0; j < allTanks.size(); j++) {

allTanks.get(j).drawTank(g);

}

for (int j = 0; j < irons.size(); j++) {

irons.get(j).draw(g);

}

for (int i = 0; i < missles.size(); i++) {

missles.get(i).drawMissle(g);

if (!missles.get(i).isLive())

missles.remove(i);

}

for (int i = 0; i < booms.size(); i++) {

if (booms.get(i).isLive())

booms.get(i).drawBoom(g);

else

booms.remove(i);

}

for (int j = 0; j < items.size(); j++) {

if (!items.get(j).isLive()) {

items.remove(j);

continue;

}

items.get(j).draw(g);

}

if (over)

drawOver(g);

messgePanel.repaint();

}

}

@Override

public void keyTyped(KeyEvent e) {

// TODO Auto-generated method stub

}

@Override

public void keyPressed(KeyEvent e) {

if (over) {

if (e.getKeyCode() == KeyEvent.VK\_F1) {

over = false;

missles.clear();

allTanks.clear();

booms.clear();

walls.clear();

irons.clear();

golds.clear();

enemyBorns.clear();

try {

init();

} catch (Exception e1) {

}

} else {

f.setVisible(false);

mainF.setSize(800, 800);

mainF.setVisible(true);

}

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

myTank.fire();

} else {

myTank.keyPress(e);

}

}

@Override

public void keyReleased(KeyEvent e) {

myTank.keyReleased(e);

}

public void aI(Tank tank) {

if (TIMEOUT) {

tank.setUp(false);

tank.setLeft(false);

tank.setDown(false);

tank.setRight(false);

return;

}

if (r.nextInt(40) == 0)

tank.fire();

if (r.nextInt(10) == 0) {

if (tank.getX() >= myTank.getX()

&& tank.getX() <= myTank.getX() + Tank.SIZE

&& tank.getY() > myTank.getY()) {

tank.setUp(true);

tank.setLeft(false);

tank.setDown(false);

tank.setRight(false);

tank.setDir(Direction.U);

return;

} else if (tank.getX() >= myTank.getX()

&& tank.getX() <= myTank.getX() + Tank.SIZE

&& tank.getY() < myTank.getY()) {

tank.setUp(false);

tank.setLeft(false);

tank.setDown(true);

tank.setRight(false);

tank.setDir(Direction.D);

return;

}

else if (tank.getX() > myTank.getX()

&& tank.getY() >= myTank.getY()

&& tank.getY() <= myTank.getY() + Tank.SIZE) {

tank.setUp(false);

tank.setLeft(true);

tank.setDown(false);

tank.setRight(false);

tank.setDir(Direction.L);

return;

} else if (tank.getX() < myTank.getX()

&& tank.getY() >= myTank.getY()

&& tank.getY() <= myTank.getY() + Tank.SIZE) {

tank.setUp(false);

tank.setLeft(false);

tank.setDown(false);

tank.setRight(true);

tank.setDir(Direction.R);

return;

}

}

if (tank.getX() <= 0) {

tank.setUp(false);

tank.setLeft(false);

tank.setDown(false);

tank.setRight(true);

tank.setDir(Direction.R);

}

if (tank.getY() <= 0) {

tank.setUp(false);

tank.setLeft(false);

tank.setDown(true);

tank.setRight(false);

tank.setDir(Direction.D);

}

if (tank.getX() >= AREA\_WIDTH - Tank.SIZE) {

tank.setUp(false);

tank.setLeft(true);

tank.setDown(false);

tank.setRight(false);

tank.setDir(Direction.L);

}

if (tank.getY() >= AREA\_HEIGHT - Tank.SIZE) {

tank.setUp(true);

tank.setLeft(false);

tank.setDown(false);

tank.setRight(false);

tank.setDir(Direction.U);

} else if (r.nextInt(300) == 1) {

tank.setUp(true);

tank.setLeft(false);

tank.setDown(false);

tank.setRight(false);

tank.setDir(Direction.U);

}

else if (r.nextInt(300) == 2) {

tank.setUp(false);

tank.setLeft(true);

tank.setDown(false);

tank.setRight(false);

tank.setDir(Direction.L);

} else if (r.nextInt(300) == 3) {

tank.setUp(false);

tank.setLeft(false);

tank.setDown(true);

tank.setRight(false);

tank.setDir(Direction.D);

} else if (r.nextInt(300) == 4) {

tank.setUp(false);

tank.setLeft(false);

tank.setDown(false);

tank.setRight(true);

tank.setDir(Direction.R);

}

}

public void initMap(File file) {

try {

FileInputStream read = new FileInputStream(file);

for (int i = 0; i < AREA\_HEIGHT / 50; i++) {

for (int j = 0; j < AREA\_WIDTH / 50; j++) {

switch (read.read()) {

case 1:

walls.add(new Wall(j \* 50, i \* 50));

break;

case 2:

irons.add(new Iron(j \* 50, i \* 50));

break;

case 3:

golds.add(new Gold(j \* 50, i \* 50));

break;

case 4:

selfBorn = new SelfBorn(j \* 50, i \* 50);

break;

case 5:

enemyBorns.add(new EnemyBorn(j \* 50, i \* 50));

break;

case 6:

home = new Home(j \* 50, i \* 50);

break;

}

}

}

read.close();

} catch (Exception e) {

}

;

}

public void addTank() {

if (tankMax <= 0)

return;

for (int i = allTanks.size(); i < enemyBorns.size() + 1; i++) {

try {

int temp = r.nextInt(enemyBorns.size());

enemyTank = new Tank(enemyBorns.get(temp).getX(), enemyBorns

.get(temp).getY(), false, allTanks, walls, irons,

golds, missles, home, booms, r.nextInt(3) + 1);

} catch (Exception e) {

e.printStackTrace();

}

enemyTank.setDir(Direction.D);

enemyTank.setDown(true);

allTanks.add(enemyTank);

tankMax--;

if (tankMax <= 0)

return;

}

}

public static void addMyTankLives() {

selfMax++;

}

private void over() {

for (int i = 0; i < AREA\_HEIGHT / 50; i++) {

for (int j = 0; j < AREA\_WIDTH / 50; j++) {

irons.add(new Iron(j \* 50, i \* 50));

p.repaint();

try {

Thread.sleep(5);

} catch (InterruptedException e) {

}

}

}

while (true) {

flash = !flash;

p.repaint();

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

}

f.addKeyListener(this);

}

}

private void drawOver(Graphics g) {

p.repaint();

g.setColor(Color.red);

g.setFont(new Font("Arial", 1, 100));

g.drawString("GAME OVER", 100, 200);

g.setFont(new Font("Arial", 2, 50));

if (win)

g.drawString("Congratulation! You Win!", 100, 400);

else

g.drawString("So Sorry, You Lose!", 100, 400);

if (flash) {

g.setFont(new Font("Arial", 2, 30));

g.setColor(Color.BLACK);

g.drawString("Press F1 to try again...,", 150, 500);

g.drawString("Press the other Key to Return the Title...,", 150,

600);

}

}

}