package stealth;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.\*;

public class Stealth extends JPanel {

public String[] data = {"\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|",

"\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_",

"\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|",

"\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_",

"\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|",

"\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_",

"\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|",

"\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_",

"\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|",

"\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_",

"\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|",

"\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_",

"\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|",

"\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_",

"\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|",

"\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_",

"\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|",

"\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_\_|\_\_"};

static JFrame f = new JFrame();

public static int score = 0;

static int xoff = 0;

static int yoff = 1000 - (19 \* 13);

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

static ArrayList<Enemy> enemies = new ArrayList<>();

public static HashMap<String, ArrayList<Enemy>> tempenemies = new HashMap();

static ArrayList<Bullet> bullets = new ArrayList<>();

public static HashMap<String, ArrayList<Bullet>> tempbullets = new HashMap();

static ArrayList<Health> healths = new ArrayList<>();

public static HashMap<String, ArrayList<Health>> temphealths = new HashMap();

static int cooldown = 0;

int cameraX = xoff - 2000 / 2;

int cameraY = yoff - 1000 / 2;

static Player first = new Player();

boolean keyA = false, keyW = false, keyS = false, keyD = false, keySpace = false;

Graphics2D g2;

// Bullet one = new Bullet(first.getXLOC()+120, first.getYLOC()+159, 10);

final int PAD = 20;

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2 = (Graphics2D) g;

update();

render(g2);

}

public void update() {

if (first.health <= 0) {

try {

Thread.sleep(1500);

f.dispose();

Thread.sleep(400);

JOptionPane.showMessageDialog(null, "Your score is " + score);

System.exit(0);

} catch (InterruptedException ex) {

Logger.getLogger(Stealth.class.getName()).log(Level.SEVERE, null, ex);

}

}

if (keyW) {

moveUp(-15);

}

if (keyA) {

moveHorizontal(-15);

}

if (keyS) {

moveUp(15);

}

if (keyD) {

moveHorizontal(15);

}

if (keySpace) {

shootBullet(50);

}

first.setXLOC(xoff);

first.setYLOC(yoff);

first.update();

for (Health e : healths) {

if (e.collidesWith(first)) {

ArrayList<Health> temp = Stealth.temphealths.get("destroy");

temp.add(e);

Stealth.temphealths.put("destroy", temp);

}

}

for (Bullet e : bullets) {

boolean test = e.update();

}

for (Bullet bullet : bullets) {

for (Enemy enemy : enemies) {

if (bullet.collidesWith(enemy)) {

}

}

if (bullet.collidesWith(first)) {

}

}

for (Bullet e : tempbullets.get("destroy")) {

bullets.remove(e);

}

for (Bullet e : tempbullets.get("make")) {

bullets.add(e);

}

tempbullets.put("destroy", new ArrayList<>());

tempbullets.put("make", new ArrayList<>());

for (Enemy e : enemies) {

e.update();

}

for (Enemy e : tempenemies.get("destroy")) {

enemies.remove(e);

}

for (Enemy e : tempenemies.get("make")) {

enemies.add(e);

}

tempenemies.put("destroy", new ArrayList<>());

tempenemies.put("make", new ArrayList<>());

for (Health e : temphealths.get("destroy")) {

healths.remove(e);

}

for (Health e : temphealths.get("make")) {

healths.add(e);

}

temphealths.put("destroy", new ArrayList<>());

temphealths.put("make", new ArrayList<>());

}

public void render(Graphics2D g) {

boolean inx = false, iny = false;

for (Wall w : walls) {

w.render(g);

}

int xtemp = 1000, ytemp = 500;

if (xoff < 1000) {

xtemp = xoff;

inx = true;

}

if (yoff < 500) {

ytemp = yoff;

iny = true;

}

int x = 0, y = 0;

x = x - xoff + 1000;

y = y - yoff + 500;

if (inx == true) {

x = 0;

}

if (iny == true) {

y = 0;

}

g.setColor(Color.LIGHT\_GRAY);

for (int z = 0; z < data.length; z++) {

for (int w = 0; w < data.length; w++) {

for (int i = 0; i < data.length; i++) {

g.drawString(data[i], x + z \* 192, i \* 13 + y + w \* 104);

}

}

}

for (Health e : healths) {

e.render(g, xoff, yoff, inx, iny);

}

g.setColor(Color.BLACK);

first.render(g, xtemp, ytemp);

g.setColor(Color.RED);

try {

for (Bullet e : bullets) {

e.render(g, xoff, yoff, inx, iny);

}

} catch (InterruptedException ex) {

Logger.getLogger(Stealth.class.getName()).log(Level.SEVERE, null, ex);

}

g.setColor(Color.BLACK);

for (Enemy e : enemies) {

e.render(g, xoff, yoff, inx, iny);

}

try {

Thread.sleep(1);

} catch (InterruptedException ex) {

Logger.getLogger(Stealth.class.getName()).log(Level.SEVERE, null, ex);

}

}

public static void main(String[] args) {

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

f.add(new Stealth());

f.setState(Frame.NORMAL);

f.setSize(2000, 1000);

f.setLocation(0, 0);

f.setVisible(true);

while (true) {

f.repaint();

if (cooldown < 0) {

enemies.add(new Enemy(Math.round((float) Math.random() \* 2000 + 1000), Math.round((float) Math.random() \* 1700 + 1)));

cooldown = 100000000;

if (Math.random() < 0.2 && healths.size() <= 3) {

healths.add(new Health());

}

} else {

cooldown--;

}

}

// while(true) {

// Thread.sleep(100);

// updateLogic();

// f.repaint();

// }

}

public Stealth() {

walls.add(new Wall(7, 10));

tempbullets.put("make", new ArrayList<Bullet>());

tempbullets.put("destroy", new ArrayList<Bullet>());

tempenemies.put("make", new ArrayList<Enemy>());

tempenemies.put("destroy", new ArrayList<Enemy>());

temphealths.put("make", new ArrayList<>());

temphealths.put("destroy", new ArrayList<>());

// enemies.add(new Enemy());

// enemies.add(new Enemy(1900,500));

// enemies.add(new Enemy(1900,100));

this.g2 = (Graphics2D) g2;

// bindKeyWith("y.up", KeyStroke.getKeyStroke(KeyEvent.VK\_W, 0), new VerticalAction(-15));

// bindKeyWith("y.down", KeyStroke.getKeyStroke(KeyEvent.VK\_S, 0), new VerticalAction(15));

// bindKeyWith("x.left", KeyStroke.getKeyStroke(KeyEvent.VK\_A, 0), new HorizontalAction(-15));

// bindKeyWith("x.right", KeyStroke.getKeyStroke(KeyEvent.VK\_D, 0), new HorizontalAction(15));

// bindKeyWith("bullet", KeyStroke.getKeyStroke(KeyEvent.VK\_SPACE, 0), new BulletAction(100));

f.addKeyListener(new KeyListener() {

@Override

public void keyTyped(KeyEvent e) {

}

@Override

public void keyPressed(KeyEvent e) {

switch (e.getKeyCode()) {

case KeyEvent.VK\_UP:

keyW = true;

break;

case KeyEvent.VK\_LEFT:

keyA = true;

break;

case KeyEvent.VK\_DOWN:

keyS = true;

break;

case KeyEvent.VK\_RIGHT:

keyD = true;

break;

case KeyEvent.VK\_SPACE:

keySpace = true;

break;

default:

break;

}

}

@Override

public void keyReleased(KeyEvent e) {

switch (e.getKeyCode()) {

case KeyEvent.VK\_UP:

keyW = false;

break;

case KeyEvent.VK\_LEFT:

keyA = false;

break;

case KeyEvent.VK\_DOWN:

keyS = false;

break;

case KeyEvent.VK\_RIGHT:

keyD = false;

break;

case KeyEvent.VK\_SPACE:

keySpace = false;

break;

default:

break;

}

}

});

}

protected void bindKeyWith(String name, KeyStroke keyStroke, Action action) {

InputMap im = getInputMap(WHEN\_IN\_FOCUSED\_WINDOW);

ActionMap am = getActionMap();

im.put(keyStroke, name);

am.put(name, action);

}

public abstract class MoveAction extends AbstractAction {

private int delta;

public MoveAction(int delta) {

this.delta = delta;

}

public int getDelta() {

return delta;

}

protected abstract void applyDelta();

@Override

public void actionPerformed(ActionEvent e) {

applyDelta();

}

}

public static void moveUp(int delta) {

yoff += delta;

if (yoff < 0) {

yoff = 0;

} else if (yoff + 100 > 1700) {

yoff = 1700 - 100;

}

first.setYLOC(yoff);

}

public static void moveHorizontal(int delta) {

xoff += delta;

if (xoff < 0) {

xoff = 0;

} else if (xoff + 100 > 3000) {

xoff = 3000 - 100;

}

first.setXLOC(xoff);

}

public static void shootBullet(int delta) {

if (first.ammo != 0) {

first.ammo = first.ammo - 1;

ArrayList<Bullet> temp = Stealth.tempbullets.get("make");

temp.add(new Bullet(first.getXLOC() - 120, first.getYLOC() + 120, 50 ));

Stealth.tempbullets.put("make", temp);

} else {

first.reloading = true;

}

}

}