**ПРИЛОЖЕНИЕ А**

**Исходный код программы**

**Bar.class**

**package** com.pacman.model;

**import** com.pacman.service.Game;

**import** com.pacman.service.GameObject;

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

**public** **class** Bar {

**private** **final** Image healthImage;

**public** **final** **int** offset = Level.BLOCK\_SIZE + 4;

**public** Bar() {

Texture healthTexture = **new** Texture(GameObject.HEALTH);

healthImage = healthTexture.getSprite(**null**, 0);

}

**public** **void** render(Graphics graphics) {

graphics.setFont(**new** Font("Monospaced", Font.BOLD + Font.ITALIC, 24));

graphics.setColor(Color.WHITE);

graphics.drawString("Score: " + PlayBoard.hero.score, 0, Game.HEIGHT - 10);

graphics.drawString("Level: " + (PlayBoard.currentLvl + 1), Game.WIDTH - 150, Game.HEIGHT - 10);

**for** (**int** i = 0; i < PlayBoard.hero.health; i++) {

graphics.drawImage(healthImage, ((Game.WIDTH / 2 - 100) + i \* offset), Game.HEIGHT - Level.BLOCK\_SIZE, 32,32, **null**);

}

}

}

**Booster.class**

**package** com.pacman.model;

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

**public** **class** Booster **extends** Rectangle {

**public** Booster(**int** x, **int** y) {

setBounds(x + 8, y + 8, 8, 8);

}

**public** **void** render(Graphics graphics) {

graphics.setColor(Color.WHITE);

graphics.fillOval(x, y, 16, 16);

}

}

**Hero.class**

**package** com.pacman.model;

**import** com.pacman.service.GameObject;

**import** com.pacman.service.HeroService;

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

**public** **class** Hero **extends** HeroService {

**private** Texture heroTexture;

**public** Hero(**int** x, **int** y) {

startX = x;

startY = y;

setBounds(x, y, Level.BLOCK\_SIZE, Level.BLOCK\_SIZE);

heroTexture = **new** Texture(GameObject.PACMAN);

speed = 4;

}

**public** **void** render(Graphics graphics) {

graphics.drawImage((heroTexture.getSprite(currentDirection, imageIndex % 2)), x, y, 32, 32, **null**);

}

}

**HeroService.class**

**package** com.pacman.service;

**import** com.pacman.model.Level;

**import** com.pacman.model.Movable;

**import** com.pacman.model.PlayBoard;

**import** static com.pacman.service.PlayBoardService.hero;

**public** **class** HeroService **extends** Movable {

**public** Direction currentDirection = Direction.LEFT;

**public** Direction nextDirection = **null**;

**private** **int** time = 0;

**private** **final** **int** targetTime = 15;

**public** **int** imageIndex = 0;

**public** **static** **int** score = 0;

**public** **static** **int** health = 3;

**public** **void** tick() {

moveToDirection(currentDirection);

**if** (nextDirection != **null** && nextDirection != currentDirection) {

**if** (isPossibleMoveToDirection(nextDirection)) {

currentDirection = nextDirection;

nextDirection = **null**;

}

}

Level level = PlayBoard.level;

**for** (**int** i = 0; i < level.tablets.size(); i++) {

**if** (**this**.intersects(level.tablets.get(i))) {

score += 10;

*//PlayBoard.A\_COIN.sound();*

level.tablets.remove(i);

**break**;

}

}

**for** (**int** i = 0; i < level.boosters.size(); i++) {

**if** (**this**.intersects(level.boosters.get(i))) {

level.boosters.remove(i);

score += 50;

**break**;

}

}

**for** (**int** i = 0; i < level.targetGhosts.size(); i++) {

**if** (**this**.intersects(level.targetGhosts.get(i))) {

die();

**break**;

}

}

**for** (**int** i = 0; i < level.randomGhosts.size(); i++) {

**if** (**this**.intersects(level.randomGhosts.get(i))) {

die();

**break**;

}

}

time++;

**if** (time == targetTime) {

time = 0;

imageIndex++;

}

}

**public** **void** die() {

health--;

PlayBoard.level.respawnActors();

**if** (health == 0) {

PlayBoard.A\_PAC\_MAN\_DEATH.sound();

Game.gameState = GameState.MENU;

health = 3;

}

}

**private** **boolean** isPossibleMoveToDirection(Direction direction) {

**int** x = hero.x;

**int** y = hero.y;

**switch** (direction) {

**case** RIGHT:

x += speed;

**break**;

**case** LEFT:

x -= speed;

**break**;

**case** UP:

y -= speed;

**break**;

**case** DOWN:

y += speed;

**break**;

}

**return** isPossibleToMove(x, y);

}

}

**Level.class**

**package** com.pacman.model;

**import** com.pacman.service.LevelService;

**import** javax.imageio.ImageIO;

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

**import** java.awt.image.BufferedImage;

**import** java.io.File;

**import** java.io.IOException;

**public** **class** Level **extends** LevelService {

**public** **static** **final** **int** BLOCK\_SIZE = 32;

**public** **int** height;

**public** **int** width;

**private** Bar bar;

**public** Level(String path) {

**try** {

BufferedImage map = ImageIO.read(getClass().getResource(path));

**this**.height = map.getHeight();

**this**.width = map.getWidth();

**int**[] pixels = **new** **int**[width \* height];

map.getRGB(0, 0, width, height, pixels, 0, width);

**for** (**int** x = 0; x < width; x++) {

**for** (**int** y = 0; y < height; y++) {

**int** colorValue = pixels[x + (y \* width)];

addObject(colorValue, x \* Level.BLOCK\_SIZE, y \* Level.BLOCK\_SIZE);

}

}

} **catch** (IOException e) {

e.printStackTrace();

}

bar = **new** Bar();

}

**public** **void** render(Graphics graphics) {

**for** (Wall b : walls) {

b.render(graphics);

}

**for** (Booster b : boosters) {

b.render(graphics);

}

**for** (Tablet t : tablets) {

t.render(graphics);

}

**for** (TargetGhost targetGhost : targetGhosts) {

targetGhost.render(graphics);

}

**for** (RandomGhost randomGhost: randomGhosts) {

randomGhost.render(graphics);

}

bar.render(graphics);

}

}

**LevelService.class**

**package** com.pacman.service;

**import** com.pacman.model.\*;

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

**import** java.util.ArrayList;

**import** static com.pacman.service.PlayBoardService.hero;

**public** **class** LevelService **extends** Canvas {

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

**public** ArrayList<Booster> boosters = **new** ArrayList<>();

**public** ArrayList<Tablet> tablets = **new** ArrayList<>();

**public** ArrayList<TargetGhost> targetGhosts = **new** ArrayList<>();

**public** ArrayList<RandomGhost> randomGhosts = **new** ArrayList<>();

**public** **void** respawnActors() {

**for**(TargetGhost targetGhost: targetGhosts)

targetGhost.respawn();

**for**(RandomGhost randomGhost: randomGhosts)

randomGhost.respawn();

hero.respawn();

}

**public** **void** tick() {

**for** (TargetGhost targetGhost : targetGhosts) {

targetGhost.tick();

}

**for** (RandomGhost randomGhost: randomGhosts) {

randomGhost.tick();

}

}

**public** **void** addObject(**int** colorValue, **int** x, **int** y) {

**switch** (colorValue) {

**case** 0xFF000000:

walls.add(**new** Wall(x, y));

**break**;

**case** 0xFFFFFF00:

hero = **new** Hero(x, y);

hero.startY = y;

hero.startX = x;

**break**;

**case** 0xFFFFFFFF:

boosters.add(**new** Booster(x, y));

**break**;

**case** 0xFFD50000:

targetGhosts.add(**new** TargetGhost(x, y));

tablets.add(**new** Tablet(x, y));

**break**;

**case** 0xFFAAAAAA:

randomGhosts.add(**new** RandomGhost(x, y));

tablets.add(**new** Tablet(x, y));

**default**:

tablets.add(**new** Tablet(x, y));

**break**;

}

}

}

**Menu.class**

**package** com.pacman.model;

**import** com.pacman.service.Game;

**import** javax.imageio.ImageIO;

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

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

**import** java.io.File;

**import** java.io.IOException;

**import** static com.pacman.service.PlayBoardService.hero;

**public** **class** Menu {

**public** **static** Rectangle playButton = **new** Rectangle(Game.WIDTH / 2 - 50, 200, 100, 50);

**public** **static** Rectangle helpButton = **new** Rectangle(Game.WIDTH / 2 - 50, 300, 100, 50);

**public** **static** Rectangle quitButton = **new** Rectangle(Game.WIDTH / 2 - 50, 400, 100, 50);

**public** Image menuImage;

**public** Menu() {

**try** {

menuImage = ImageIO.read(getClass().getResource("/res/menu/picture\_pacman.jpg"));

} **catch** (IOException e) {

JOptionPane.showMessageDialog(**null**, "Some files not found. Try to reinstall your application.","Error", JOptionPane.ERROR\_MESSAGE);

System.exit(0);

e.printStackTrace();

}

}

**public** **void** render(Graphics graphics) {

graphics.drawImage(menuImage, 0,0, Game.WIDTH + 400, Game.HEIGHT, **null**);

Graphics2D graphics2D = (Graphics2D) graphics;

graphics.setFont(**new** Font("arial", Font.BOLD, 50));

graphics.setColor(Color.YELLOW);

graphics.drawString("PAC-MAN GAME", Game.WIDTH / 5, 100);

graphics.setColor(Color.WHITE);

graphics.setFont(**new** Font("arial", Font.BOLD, 20));

graphics.drawString("Score: " + Integer.toString(hero.score), Game.WIDTH / 2 - 50, 150);

graphics.setFont(**new** Font("arial", Font.ITALIC + Font.BOLD, 30));

graphics.drawString("Play", playButton.x + 19, playButton.y + 35);

graphics2D.draw(playButton);

graphics.drawString("Help", helpButton.x + 19, helpButton.y + 35);

graphics2D.draw(helpButton);

graphics.drawString("Exit", quitButton.x + 19, quitButton.y + 35);

graphics2D.draw(quitButton);

graphics.setFont(**new** Font("arial", Font.ITALIC, 14));

graphics.drawString("Developed by Vladislav Maiski, BSUIR ", 0, Game.HEIGHT - graphics.getFont().getSize());

}

}

**PlayBoard.class**

**package** com.pacman.model;

**import** com.pacman.service.Game;

**import** com.pacman.service.\*;

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

**public** **class** PlayBoard **extends** PlayBoardService {

**public** PlayBoard() {

hero = **new** Hero(Game.WIDTH / 2, Game.HEIGHT / 2);

level = **new** Level("/res/map/map" + currentLvl + ".png");

}

**public** **void** render(Graphics graphics) {

hero.render(graphics);

level.render(graphics);

}

}

**PlayBoardService.class**

**package** com.pacman.service;

**import** com.pacman.model.Hero;

**import** com.pacman.model.Level;

**import** java.awt.event.KeyEvent;

**import** java.awt.event.KeyListener;

**public** **class** PlayBoardService {

**public** **static** **int** currentLvl = 0;

**private** **final** **int** MAX\_LVL = 1;

**public** **static** String levelPath = "/res/map/map" + currentLvl + ".png";

**public** **static** **final** Audio A\_COIN = **new** Audio("res/sound/game/coin.wav", 0.5);;

**public** **static** Audio A\_PAC\_MAN\_DEATH = **new** Audio("res/sound/game/pacman\_death.wav", 0.5);

**public** **static** Level level;

**public** **static** Hero hero;

**public** **void** tick() {

hero.tick();

level.tick();

**if** (level.tablets.size() == 0 && level.boosters.size() == 0) {

nextLvl();

}

}

**public** **void** setLevel(**int** number) {

level = **new** Level("/res/map/map" + number + ".png");

}

**public** **void** nextLvl() {

**if** (currentLvl + 1 <= MAX\_LVL) {

currentLvl++;

}

level = **new** Level("/res/map/map" + currentLvl + ".png");

}

}

**RandomGhost.class**

**package** com.pacman.model;

**import** com.pacman.service.Direction;

**import** com.pacman.service.GameObject;

**import** com.pacman.service.RandomGhostService;

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

**public** **class** RandomGhost **extends** RandomGhostService {

**private** Texture randomGhostTexture;

**public** RandomGhost(**int** x, **int** y) {

startX = x;

startY = y;

setBounds(x, y, Level.BLOCK\_SIZE, Level.BLOCK\_SIZE);

randomGhostTexture = **new** Texture(GameObject.RANDOM\_GHOST);

direction = Direction.getRandomDirection();

speed = 2;

}

**public** **void** render(Graphics graphics) {

graphics.drawImage(randomGhostTexture.getSprite(direction, imageIndex % 2), x, y, 32, 32, **null**);

}

}

**RandomGhostService.class**

**package** com.pacman.service;

**public** **class** RandomGhostService **extends** Enemy {

**public** **void** tick() {

randomMove();

}

@Override

**public** **void** randomMove() {

**if** (!moveToDirection(direction)) {

direction = Direction.getRandomDirection();

}

imageTime++;

**if** (imageTime == targetImageTime) {

imageTime = 0;

imageIndex++;

}

}

}

**Movable.class**

**package** com.pacman.model;

**import** com.pacman.service.Direction;

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

**public** **abstract** **class** Movable **extends** Rectangle {

**public** **int** speed;

**public** **int** startX;

**public** **int** startY;

**public** **boolean** isPossibleToMove(**int** xNext, **int** yNext) {

Rectangle bounds = **new** Rectangle(xNext, yNext, width, height);

Level level = PlayBoard.level;

**boolean** isPossible = **true**;

**int** i = 0;

**while** (isPossible && i < level.walls.size()) {

**if** (bounds.intersects(level.walls.get(i)))

isPossible = **false**;

i++;

}

**return** isPossible;

}

**public** **boolean** moveToDirection(Direction direction) {

**boolean** move = **false**;

**switch** (direction) {

**case** RIGHT:

**if** (isPossibleToMove(x + speed, y)) {

x += speed;

move = **true**;

}

**break**;

**case** LEFT:

**if** (isPossibleToMove(x - speed, y)) {

x -= speed;

move = **true**;

}

**break**;

**case** UP:

**if** (isPossibleToMove(x, y - speed)) {

y -= speed;

move = **true**;

}

**break**;

**case** DOWN:

**if** (isPossibleToMove(x, y + speed)) {

y += speed;

move = **true**;

}

**break**;

}

**return** move;

}

**public** **void** respawn() {

x = startX;

y = startY;

}

}

**SpriteSheet.class**

**package** com.pacman.model;

**import** javax.imageio.ImageIO;

**import** java.awt.image.BufferedImage;

**import** java.io.File;

**import** java.io.IOException;

**public** **class** SpriteSheet {

**public** BufferedImage sheet;

**public** SpriteSheet(String path) {

**try** {

sheet = ImageIO.read(getClass().getResource(path));

} **catch** (IOException e) {

System.out.println("IMAGE EXCEPTION");

}

}

**public** BufferedImage getSprite(**int** x, **int** y) {

**return** sheet.getSubimage(x \* 16, y \* 16,16, 16);

}

}

**Tablet.class**

**package** com.pacman.model;

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

**public** **class** Tablet **extends** Rectangle {

**public** Tablet(**int** x, **int** y) {

setBounds(x + 12, y + 12, 8, 8);

}

**public** **void** render(Graphics graphics) {

graphics.setColor(Color.WHITE);

graphics.fillOval(x, y, 8, 8);

}

}

**TargetGhost.class**

**package** com.pacman.model;

**import** com.pacman.service.Direction;

**import** com.pacman.service.GameObject;

**import** com.pacman.service.TargetGhostService;

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

**public** **class** TargetGhost **extends** TargetGhostService {

**private** Texture targetGhostTexture;

**public** TargetGhost(**int** x, **int** y) {

startX = x;

startY = y;

setBounds(x, y, Level.BLOCK\_SIZE, Level.BLOCK\_SIZE);

direction = Direction.getRandomDirection();

targetGhostTexture = **new** Texture(GameObject.TARGET\_GHOST);

speed = 1;

}

**public** **void** render(Graphics graphics) {

**if** (state == State.SMART) {

graphics.drawImage(targetGhostTexture.getSprite(lastDirection, imageIndex % 2), x, y, 32, 32, **null**);

} **else** {

graphics.drawImage(targetGhostTexture.getSprite(direction, imageIndex % 2), x, y, 32, 32, **null**);

}

}

}

**Texture.class**

**package** com.pacman.model;

**import** com.pacman.service.Direction;

**import** com.pacman.service.GameObject;

**import** java.awt.image.BufferedImage;

**public** **class** Texture {

**private** SpriteSheet spriteSheet = **new** SpriteSheet("/res/spritesheet/spritesheet.png");

**private** BufferedImage[] right = **new** BufferedImage[2];

**private** BufferedImage[] left = **new** BufferedImage[2];

**private** BufferedImage[] up = **new** BufferedImage[2];

**private** BufferedImage[] down = **new** BufferedImage[2];

**public** Texture(GameObject actor) {

loadSprites(actor);

}

**private** **void** loadMovable(**int** line) {

right[0] = spriteSheet.getSprite(0, line);

right[1] = spriteSheet.getSprite(1, line);

left[0] = spriteSheet.getSprite(2, line);

left[1] = spriteSheet.getSprite(3, line);

up[0] = spriteSheet.getSprite(4, line);

up[1] = spriteSheet.getSprite(5, line);

down[0] = spriteSheet.getSprite(6, line);

down[1] = spriteSheet.getSprite(7, line);

}

**private** **void** loadSprites(GameObject actor) {

**switch** (actor) {

**case** TARGET\_GHOST:

loadMovable(3);

**break**;

**case** RANDOM\_GHOST:

loadMovable(1);

**break**;

**case** PACMAN:

loadMovable(0);

**break**;

**case** WALL:

right[0] = spriteSheet.getSprite(0,2);

**break**;

**case** HEALTH:

right[0] = spriteSheet.getSprite(1,2);

**break**;

}

}

**public** BufferedImage getSprite(Direction direction, **int** index) {

BufferedImage resImage = **null**;

**if** (direction == **null**) {

**return** right[0];

}

**switch** (direction) {

**case** UP:

**if** (index == 0) {

resImage = up[0];

} **else** {

resImage = up[1];

}

**break**;

**case** LEFT:

**if** (index == 0) {

resImage = left[0];

} **else** {

resImage = left[1];

}

**break**;

**case** DOWN:

**if** (index == 0) {

resImage = down[0];

} **else** {

resImage = down[1];

}

**break**;

**case** RIGHT:

**if** (index == 0) {

resImage = right[0];

} **else** {

resImage = right[1];

}

**break**;

}

**return** resImage;

}

}

**Wall.class**

**package** com.pacman.model;

**import** com.pacman.service.GameObject;

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

**public** **class** Wall **extends** Rectangle {

**private** Texture wallTexture;

**public** Wall(**int** x, **int** y) {

setBounds(x, y, Level.BLOCK\_SIZE, Level.BLOCK\_SIZE);

wallTexture = **new** Texture(GameObject.WALL);

}

**public** **void** render(Graphics graphics) {

graphics.drawImage(wallTexture.getSprite(**null**, 0), x, y, 32, 32, **null**);

}

}

**Audio.class**

**package** com.pacman.service;

**import** java.io.File;

**import** java.io.IOException;

**import** javax.sound.sampled.\*;

**import** javax.sound.sampled.spi.AudioFileReader;

**public** **class** Audio {

**private** String track;

**private** Clip clip = **null**;

**private** FloatControl volumeC;

**private** **double** wt;

**private** **boolean** play\_audio;

**public** Audio(String track, **double** wt) {

**this**.track = track;

**this**.wt = wt;

**this**.play\_audio = **false**;

}

**public** **void** sound() {

File file = **new** File(track);

AudioInputStream audioInputStream = **null**;

**try** {

audioInputStream = AudioSystem.getAudioInputStream(file);

} **catch** (UnsupportedAudioFileException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}

**try** {

clip = AudioSystem.getClip();

clip.open(audioInputStream);

volumeC = (FloatControl) clip.getControl(FloatControl.Type.MASTER\_GAIN);

setVolume();

clip.setFramePosition(0);

clip.start();

} **catch** (LineUnavailableException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**private** **void** setVolume () {

**if** (wt < 0)

wt = 0;

**if** (wt > 1)

wt = 1;

**float** min = volumeC.getMinimum();

**float** max = volumeC.getMaximum();

volumeC.setValue((max - min) \* (**float**)wt + min);

}

}

**Direction.class**

**package** com.pacman.service;

**import** java.util.Random;

**public** **enum** Direction {

RIGHT,

LEFT,

UP,

DOWN;

**public** **static** Direction getRandomDirection() {

Random random = **new** Random();

**return** values()[random.nextInt(values().length)];

}

}

**Enemy.class**

**package** com.pacman.service;

**import** com.pacman.model.Movable;

**import** static com.pacman.service.PlayBoardService.hero;

**public** **abstract** **class** Enemy **extends** Movable {

**public** **enum** State {

RANDOM,

SMART,

FIND\_PATH

}

**public** **int** time = 0;

**public** **final** **int** targetTime = 60 \* 4;

**public** **final** **int** targetImageTime = 15;

**public** **int** imageTime = 0;

**public** **int** imageIndex = 0;

**public** Direction lastDirection;

**public** TargetGhostService.State state = TargetGhostService.State.RANDOM;

**public** Direction direction;

**public** **void** randomMove() {

**switch** (direction) {

**case** RIGHT:

**if** (isPossibleToMove(x + speed, y)) {

x += speed;

} **else** {

direction = Direction.getRandomDirection();

}

**break**;

**case** LEFT:

**if** (isPossibleToMove(x - speed, y)) {

x -= speed;

} **else** {

direction = Direction.getRandomDirection();

}

**break**;

**case** UP:

**if** (isPossibleToMove(x, y - speed)) {

y -= speed;

} **else** {

direction = Direction.getRandomDirection();

}

**break**;

**case** DOWN:

**if** (isPossibleToMove(x, y + speed)) {

y += speed;

} **else** {

direction = Direction.getRandomDirection();

}

**break**;

}

time++;

**if** (time == targetTime) {

state = State.SMART;

time = 0;

}

}

**public** **void** smartMove() {

**boolean** inMove = **false**;

**if** (x < hero.x && isPossibleToMove(x + speed, y)) {

x += speed;

lastDirection = Direction.RIGHT;

inMove = **true**;

}

**if** (x > hero.x && isPossibleToMove(x - speed, y)) {

x -= speed;

lastDirection = Direction.LEFT;

inMove = **true**;

}

**if** (y > hero.y && isPossibleToMove(x, y - speed)) {

y -= speed;

lastDirection = Direction.UP;

inMove = **true**;

}

**if** (y < hero.y && isPossibleToMove(x, y + speed)) {

y += speed;

lastDirection = Direction.DOWN;

inMove = **true**;

}

**if** (x == hero.x) {

inMove = **true**;

}

**if** (!inMove) {

state = State.FIND\_PATH;

}

time++;

**if** (time == targetTime) {

state = State.RANDOM;

time = 0;

}

}

**public** **void** findPathMove() {

**if** (lastDirection != **null**)

**switch** (lastDirection) {

**case** RIGHT:

**if** (y < hero.y && isPossibleToMove(x, y + speed)) {

y += speed;

state = State.SMART;

} **else** **if** (isPossibleToMove(x, y - speed)) {

y -= speed;

state = State.SMART;

}

**if** (isPossibleToMove(x + speed, y))

x += speed;

**break**;

**case** LEFT:

**if** (y < hero.y && isPossibleToMove(x, y + speed)) {

y += speed;

state = State.SMART;

} **else** **if** (isPossibleToMove(x, y - speed)) {

y -= speed;

state = State.SMART;

}

**if** (isPossibleToMove(x - speed, y))

x -= speed;

**break**;

**case** UP:

**if** (x < hero.x && isPossibleToMove(x + speed, y)) {

x += speed;

state = State.SMART;

} **else** **if** (isPossibleToMove(x - speed, y)) {

x -= speed;

state = State.SMART;

}

**if** (isPossibleToMove(x, y - speed))

y -= speed;

**break**;

**case** DOWN:

**if** (x < hero.x && isPossibleToMove(x + speed, y)) {

x += speed;

state = State.SMART;

} **else** **if** (isPossibleToMove(x - speed, y)) {

x -= speed;

state = State.SMART;

}

**if** (isPossibleToMove(x, y + speed))

y += speed;

**break**;

}

time++;

**if** (time == targetTime) {

state = State.RANDOM;

time = 0;

}

}

}

**MouseInput.class**

**package** com.pacman.service;

**import** com.pacman.model.Hero;

**import** com.pacman.model.Level;

**import** com.pacman.model.Menu;

**import** com.pacman.model.PlayBoard;

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

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

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**public** **class** MouseInput **implements** MouseListener {

**private** Audio a\_click = **new** Audio("res/sound/menu/click.wav", 0.5);

**private** String HELP\_BTN = "Use the W, S, D, and A keys to navigate. Collect all the pills to go to the next level.**\n** Avoid ghosts. Good luck ...**\n**Developed by Vladislav Maiski";

@Override

**public** **void** mouseClicked(MouseEvent e) {

}

@Override

**public** **void** mousePressed(MouseEvent e) {

}

@Override

**public** **void** mouseReleased(MouseEvent e) {

**if** (Game.gameState == GameState.MENU) {

Rectangle mouseRange = **new** Rectangle(e.getX(), e.getY(), 1, 1);

**if** (mouseRange.intersects(Menu.playButton)) {

a\_click.sound();

PlayBoard.currentLvl = 0;

Game.playBoard.setLevel(PlayBoard.currentLvl);

Game.gameState = GameState.GAME;

Hero.score = 0;

}

**if** (mouseRange.intersects(Menu.quitButton)) {

a\_click.sound();

System.exit(0);

}

**if** (mouseRange.intersects(Menu.helpButton)) {

a\_click.sound();

JOptionPane.showMessageDialog(Game.gameFrame, HELP\_BTN, "Information", JOptionPane.INFORMATION\_MESSAGE);

}

}

}

@Override

**public** **void** mouseEntered(MouseEvent e) {

}

@Override

**public** **void** mouseExited(MouseEvent e) {

}

}

**Game.class**

**package** com.pacman.service;

**import** com.pacman.model.Menu;

**import** com.pacman.model.PlayBoard;

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

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

**import** java.awt.event.KeyEvent;

**import** java.awt.event.KeyListener;

**import** java.awt.image.BufferStrategy;

**public** **class** Game **extends** Canvas **implements** Runnable, KeyListener {

**private** **boolean** isRunning = **false**;

**public** **static** **final** **int** WIDTH = 640;

**public** **static** **final** **int** HEIGHT = 512;

**private** **static** **final** **double** TICK\_RATE = 60.0;

**private** **static** **final** String TITLE = "PAC-MAN 2";

**public** **static** GameState gameState = GameState.MENU;

**public** **static** JFrame gameFrame = **new** JFrame();

**public** **static** Game game;

**public** **static** Menu menu;

**public** **static** PlayBoard playBoard;

**public** **static** Thread thread;

**public** Game() {

Dimension dimension = **new** Dimension(WIDTH, HEIGHT);

setPreferredSize(dimension);

setMinimumSize(dimension);

setMaximumSize(dimension);

addKeyListener(**this**);

addMouseListener(**new** MouseInput());

menu = **new** Menu();

playBoard = **new** PlayBoard();

}

**private** **synchronized** **void** start() {

**if** (isRunning)

**return**;

isRunning = **true**;

thread = **new** Thread(**this**);

thread.start();

}

**private** **synchronized** **void** stop() {

**if** (!isRunning)

**return**;

isRunning = **false**;

**try** {

thread.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

**public** **void** render() {

BufferStrategy bufferStrategy = getBufferStrategy();

**if** (bufferStrategy == **null**) {

createBufferStrategy(3);

**return**;

}

Graphics graphics = bufferStrategy.getDrawGraphics();

graphics.setColor(Color.BLACK);

graphics.fillRect(0, 0, WIDTH, HEIGHT);

**if** (gameState == GameState.GAME) {

playBoard.render(graphics);

} **else** **if** (gameState == GameState.MENU) {

menu.render(graphics);

}

graphics.dispose();

bufferStrategy.show();

}

**public** **void** tick() {

**if** (gameState == GameState.GAME) {

playBoard.tick();

}

}

@Override

**public** **void** run() {

requestFocus();

**double** timer = System.currentTimeMillis();

**long** prefTime = System.nanoTime();

**int** fps = 0;

**double** delta = 0;

**double** nanoSecs = 1e9 / TICK\_RATE;

**while** (isRunning) {

**long** curTime = System.nanoTime();

delta += (curTime - prefTime) / nanoSecs;

prefTime = curTime;

**while** (delta >= 1) {

render();

tick();

delta--;

fps++;

}

**if** (System.currentTimeMillis() - timer >= 1000) {

System.out.println(fps);

timer += 1000;

fps = 0;

}

}

start();

}

**public** **void** keyPressed(KeyEvent keyEvent) {

**if** (Game.gameState == GameState.GAME) {

**if** (keyEvent.getKeyCode() == KeyEvent.VK\_W || keyEvent.getKeyCode() == KeyEvent.VK\_UP) {

PlayBoard.hero.nextDirection = Direction.UP;

}

**if** (keyEvent.getKeyCode() == KeyEvent.VK\_S || keyEvent.getKeyCode() == KeyEvent.VK\_DOWN) {

PlayBoard.hero.nextDirection = Direction.DOWN;

}

**if** (keyEvent.getKeyCode() == KeyEvent.VK\_D || keyEvent.getKeyCode() == KeyEvent.VK\_RIGHT) {

PlayBoard.hero.nextDirection = Direction.RIGHT;

}

**if** (keyEvent.getKeyCode() == KeyEvent.VK\_A || keyEvent.getKeyCode() == KeyEvent.VK\_LEFT) {

PlayBoard.hero.nextDirection = Direction.LEFT;

}

}

}

@Override

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

}

@Override

**public** **void** keyTyped(KeyEvent keyEvent) {

}

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

game = **new** Game();

gameFrame.add(game);

gameFrame.setTitle(TITLE);

gameFrame.setResizable(**false**);

gameFrame.pack();

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

gameFrame.setLocationRelativeTo(**null**);

gameFrame.setVisible(**true**);

game.start();

}

}