Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Парадигмы и конструкции языков программирования»

Отчет по Домашнему заданию «Арканоид»

Выполнил:

Макеева Е. А.

ИУ5-31Б

Подпись и дата:

Проверил:

Нардид А. Н.

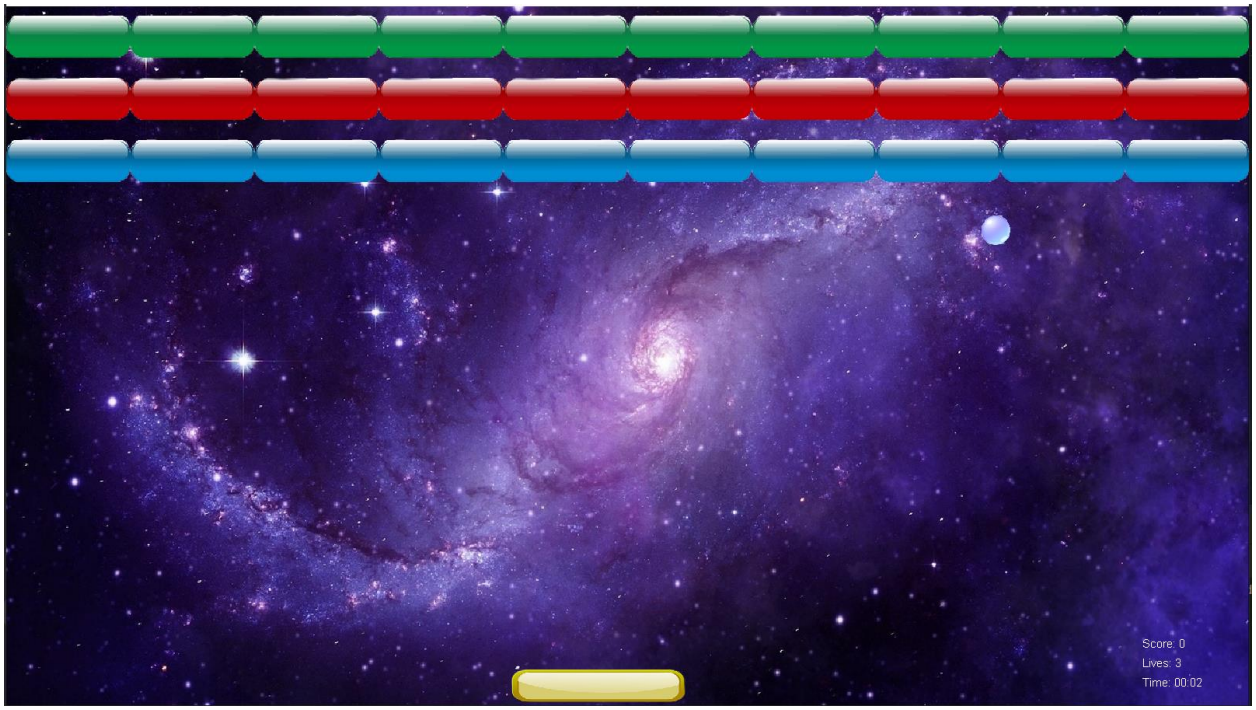
Подпись и дата:

В рамках проекта осуществлено программирование видеоигры «Арканоид» на языке Java, где игрок контролирует небольшую платформу-ракетку, которую можно передвигать горизонтально от одной стенки до другой, подставляя её под шарик, предотвращая его падение вниз. Удар шарика по кирпичу приводит к разрушению кирпича.

Внешний вид:

Заставка игры (просмотр заставки сопровождается звуковым фрагментом, для пропуска заставки игроку требуется нажать на любую клавишу на клавиатуре или кнопку мыши):

Основное игровое поле:



Экран «You win»

Экран «Game over»


```
import javax.imageio.ImageIO;
import java.awt.*;
import java.awt.image.BufferedImage;
```

```java
import java.io.File;
import java.io.IOException;

public class Ball {
    private int x, y, diameter = 30;
    private int dx = 12, dy = -12;
    private BufferedImage image;
    private MainGame mainGame;

    public Ball(int x, int y, MainGame mainGame) {
        this.x = x;
        this.y = y;
        this.mainGame = mainGame;
        loadImage();
    }

    private void loadImage() {
        try {
            image = ImageIO.read(new File("data/ball.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void move() {
        x += dx;
        y += dy;

        if (x < 0 || x > 1280 - diameter) {
            dx = -dx;
        }
        if (y < 0) {
            dy = -dy;
        }
        if (y > 720 - diameter) {
            y = 720 - diameter;
            dy = -dy;
            mainGame.lives--;
            if (mainGame.lives <= 0) {
                mainGame.endGame();
            }
        }
    }

    public void reverseY() {
        dy = -dy;
    }

    public Rectangle getBounds() {
        return new Rectangle(x, y, diameter, diameter);
    }

    public void draw(Graphics g) {
        g.drawImage(image, x, y, diameter, diameter, null);
    }
}
```

```java
import javax.imageio.ImageIO;
import java.awt.*;
```

```java
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

public class Block {
    private int x, y, width = 128, height = 64;
    private BufferedImage image;

    public Block(int x, int y, String imagePath) {
        this.x = x;
        this.y = y;
        loadImage(imagePath);
    }

    private void loadImage(String imagePath) {
        try {
            image = ImageIO.read(new File(imagePath));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public Rectangle getBounds() {
        return new Rectangle(x, y, width, height);
    }

    public void draw(Graphics g) {
        g.drawImage(image, x, y, width, height, null);
    }
}
```

```java
import javax.imageio.ImageIO;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

public class Player {
    private int x, y, width = 200, height = 40;
    private BufferedImage image;

    public Player(int x, int y) {
        this.x = x;
        this.y = y;
        loadImage();
    }

    private void loadImage() {
        try {
            image = ImageIO.read(new File("data/player.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void moveLeft() {
        if (x > 0) x -= 15;
    }
```

```
    public void moveRight() {
        if (x < 1280 - width) x += 15;
    }

    public void update(boolean leftPressed, boolean rightPressed) {
        if (leftPressed) {
            moveLeft();
        }
        if (rightPressed) {
            moveRight();
        }
    }

    public Rectangle getBounds() {
        return new Rectangle(x, y, width, height);
    }

    public void draw(Graphics g) {
        g.drawImage(image, x, y, width, height, null);
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;

public class StartScreen extends JPanel {
    private BufferedImage backgroundImage;

    public StartScreen() {
        try {
            backgroundImage = ImageIO.read(new File("data/fon_start.jpg"));
        } catch (IOException e) {
            e.printStackTrace();
        }

        setPreferredSize(new Dimension(1280, 720));
        setFocusable(true);
        addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                openMainGame();
            }
        });
    }

    private void openMainGame() {
        JFrame topFrame = (JFrame) SwingUtilities.getWindowAncestor(this);
        topFrame.dispose();

        JFrame gameFrame = new JFrame("Main Game");
        MainGame gamePanel = new MainGame();
        gameFrame.add(gamePanel);
        gameFrame.pack();
        gameFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        gameFrame.setVisible(true);
```

```
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (backgroundImage != null) {
            g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(),
null);
        }
    }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import java.util.ArrayList;
import java.awt.event.ActionEvent;

public class MainGame extends JPanel implements ActionListener, KeyListener {
    private Timer timer;
    private Ball ball;
    private Player player;
    private ArrayList<Block> blocks;
    private int score = 0;
    public int lives = 3;
    private BufferedImage backgroundImage;
    private long startTime;

    private boolean leftPressed = false;
    private boolean rightPressed = false;

    public MainGame() {
        setPreferredSize(new Dimension(1280, 720));
        setBackground(Color.BLACK);
        setFocusable(true);
        addKeyListener(this);

        timer = new Timer(50, this);
        timer.start();

        ball = new Ball(620, 600, this);
        player = new Player(512, 680);
        blocks = new ArrayList<>();
        initializeBlocks();
        loadBackgroundImage();

        startTime = System.currentTimeMillis();
    }

    private void loadBackgroundImage() {
        try {
            backgroundImage = ImageIO.read(new File("data/fon.jpg"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
    private void initializeBlocks() {
        for (int i = 0; i < 1280; i += 128) {
            blocks.add(new Block(i, 0, "data/block_green.png"));
            blocks.add(new Block(i, 64, "data/block_red.png"));
            blocks.add(new Block(i, 128, "data/block_blue.png"));
        }
    }

    public void actionPerformed(ActionEvent e) {
        ball.move();
        player.update(leftPressed, rightPressed);
        checkCollisions();
        repaint();
    }

    private void checkCollisions() {
        for (Block block : blocks) {
            if (ball.getBounds().intersects(block.getBounds())) {
                blocks.remove(block);
                ball.reverseY();
                score += 10;
                break;
            }
        }

        if (ball.getBounds().intersects(player.getBounds())) {
            ball.reverseY();
        }

        if (blocks.isEmpty()) {
            winGame();
        }
    }

    public void endGame() {
        JFrame topFrame = (JFrame) SwingUtilities.getWindowAncestor(this);
        topFrame.dispose();

        JFrame gameOverFrame = new JFrame("Game Over");
        JPanel panel = new JPanel() {
            private BufferedImage backgroundImage;

            {
                try {
                    backgroundImage = ImageIO.read(new
File("data/fon_end.jpg"));
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                if (backgroundImage != null) {
                    g.drawImage(backgroundImage, 0, 0, getWidth(),
getHeight(), null);
                }
                g.setColor(Color.WHITE);
                g.drawString("Score: " + score, 640, 660);
                long elapsedTime = System.currentTimeMillis() - startTime;
                long seconds = (elapsedTime / 1000) % 60;
                long minutes = (elapsedTime / 1000) / 60;
                g.drawString(String.format("Time: %02d:%02d", minutes,
```

```
seconds), 640, 700);
            }
        };

        panel.setPreferredSize(new Dimension(1280, 720));
        gameOverFrame.add(panel);
        gameOverFrame.pack();
        gameOverFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        gameOverFrame.setVisible(true);
    }

    private void winGame() {
        JFrame topFrame = (JFrame) SwingUtilities.getWindowAncestor(this);
        topFrame.dispose();

        JFrame winFrame = new JFrame("You Win!");
        JPanel panel = new JPanel() {
            private BufferedImage backgroundImage;

            {
                try {
                    backgroundImage = ImageIO.read(new
File("data/fon_win.jpg"));
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                if (backgroundImage != null) {
                    g.drawImage(backgroundImage, 0, 0, getWidth(),
getHeight(), null);
                }
                g.setColor(Color.WHITE);
                g.drawString("Score: " + score, 640, 660);
                g.drawString("Lives: " + lives, 640, 680);

                long elapsedTime = System.currentTimeMillis() - startTime;
                long seconds = (elapsedTime / 1000) % 60;
                long minutes = (elapsedTime / 1000) / 60;
                g.drawString(String.format("Time: %02d:%02d", minutes,
seconds), 640, 700);
            }
        };

        panel.setPreferredSize(new Dimension(1280, 720));
        winFrame.add(panel);
        winFrame.pack();
        winFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        winFrame.setVisible(true);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (backgroundImage != null) {
            g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(),
null);
        }
        ball.draw(g);
        player.draw(g);
        for (Block block : blocks) {
```

```
            block.draw(g);
        }
        g.setColor(Color.WHITE);
        g.drawString("Score: " + score, 1170, 660);
        g.drawString("Lives: " + lives, 1170, 680);

        long elapsedTime = System.currentTimeMillis() - startTime;
        long seconds = (elapsedTime / 1000) % 60;
        long minutes = (elapsedTime / 1000) / 60;
        g.drawString(String.format("Time: %02d:%02d", minutes, seconds),
1170, 700);
    }

    @Override
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            leftPressed = true;
        }
        if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
            rightPressed = true;
        }
    }

    @Override
    public void keyReleased(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            leftPressed = false;
        }
        if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
            rightPressed = false;
        }
    }

    @Override
    public void keyTyped(KeyEvent e) {
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Start Screen");
        StartScreen startScreen = new StartScreen();
        frame.add(startScreen);
        frame.pack();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```