МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем и цифровых технологий

Работа допущена к защите
_____Руководитель
«05» марта 2025 г.

КУРСОВАЯ РАБОТА

по дисциплине «Программирование на языке Java»

на тему: «Разработка программы «Игра гонки» на языке Java»

Студент _____ Королев А. С.

Шифр 220893

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.03 Прикладная информатика

Направленность (профиль) Интеллектуальная обработка данных

Группа 31ПИ

Руководитель _____ Лукьянов П. В.

Оценка: «отлично» Дата 05.05.2025

Орел 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем и цифровых технологий

УТВЕРЖДАЮ:
_____ и.о. зав. кафедрой
«7» 10 2024 г.

ЗАДАНИЕ
на курсовую работу

по дисциплине «Программирование на языке Java»

Студент Королев А. С. Шифр 220893

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.03 Прикладная информатика

Направленность (профиль) Интеллектуальная обработка данных

Группа 31ПИ

1 Тема курсовой работы

«Разработка программы «Игра гонки» на языке Java»

2 Срок сдачи студентом законченной работы «24» декабря 2024

3 Исходные данные

Разработать программу, реализующую игру «Гонки» на языке Java. При проектировании классов выделить один класс контейнер, в переменные экземпляра которого заносятся объекты других классов. Предусмотреть возможность сериализации объекта класса контейнера, для сохранения и восстановления состояния игры. Разработать абстрактный класс, в котором выделить два метода (draw(), update()), отвечающих за отображение и изменение состояния объектов, а также необходимые переменные экземпляра, отображающие общие характеристики всех объектов. Объекты всех классов, кроме класса контейнера, впоследствии должны наследоваться от абстрактного класса, переопределяя и реализуя его методы (draw(), update()). В одном из методов класса контейнера, реализующего логику игры (игровой цикл), вызывать переопределенные методы draw() и update() для объектов других классов.

4 Содержание курсовой работы

Описание правил игры «Гонки»

Разработка основного алгоритма программы

Диаграмма классов программы

Реализация и тестирование программного средства

5 Отчетный материал курсовой работы

Пояснительная записка курсовой работы; приложение (на Java), записанное на электронном носителе.

Руководитель _____ Лукьянов П. В.

Задание принял к исполнению: «7» октября 2024

Подпись студента _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Жанр гонок — это один из самых популярных и динамичных жанров видеоигр, где основной акцент сделан на соревнованиях с использованием различных транспортных средств. Игроки управляют автомобилями, мотоциклами, грузовиками, самолётами или даже фантастическими транспортными средствами, участвуя в гонках на скорость, выполняя трюки или преодолевая сложные трассы.

Целью курсовой работы по дисциплине «Программирование на языке Java» является приобретение практических навыков программирования на языке Java на примере реализации игры в жанре «Гонки».

Основными задачами выполнения курсовой работы являются:

- Описать и проанализировать поставленную задачу;
- Изучить правила игры «Гонки» и реализовать их в коде игры;
- Построить диаграмму классов предметной области;
- Обосновать выбор концепций и механизмов ООП, необходимых для моделирования предметной области;
- Спроектировать пользовательский интерфейс;
- Реализовать и протестировать готовое программное средство.

ОПИСАНИЕ И АНАЛИЗ ПОСТАВЛЕННОЙ ЗАДАЧИ

Курсовая работа посвящена анализу и реализации игры в жанре «Гонки».

Игра представляет собой гонку, в которой игрок управляет транспортным средством и стремится проехать как можно дальше. Основная задача — избегать столкновений и набирать очки за пройденное расстояние.

В процессе игры машина игрока может передвигаться в разные стороны, тем самым уклоняясь от вражеских машин. Управление персонажем происходит через нажатие клавиш, что позволяет реагировать на препятствия на пути. В игре также реализовано попутное и встречное движение машин по дороге.

Игра продолжается до тех пор, пока персонаж не столкнется с препятствием или врагом. Ключевым элементом игры является нарастающая скорость, что делает ее более сложной и увлекательной по мере прохож

ПОСТРОЕНИЕ ДИАГРАММЫ КЛАССОВ ПРЕДМЕТНОЙ ОБЛАСТИ

1. Класс Main

Описание: Главный класс игры, который управляет запуском окна и переключением между меню и игровым процессом.

Методы:

- Main() – конструктор, настраивающий параметры окна.
- showMenu() – отображает главное меню.
- startGame() – запускает игру, заменяя экран меню на игровое поле.
- main(String[] args) – метод запуска приложения в потоке Swing.

2. Класс Menu

Описание: Отображает стартовое меню игры с кнопкой начала.

Методы:

- Menu(Main mainFrame) – конструктор, инициализирующий главное меню.

3. Класс Road

Описание: Отвечает за игровой процесс, движение дороги, управление игроком и появление препятствий.

Методы:

- Road() – конструктор, инициализирующий игровое поле, обработчик событий клавиатуры и таймер.
- run() – метод потока, генерирующий врагов с случайным интервалом.
- paint(Graphics g) – рисует дорогу, игрока и врагов.
- actionPerformed(ActionEvent event) – метод для обновления состояния игры.
- testCollisionWithEnemies() – проверяет столкновения игрока с препятствиями.
- returnToMenu() – возвращает игрока в меню после проигрыша.

4. Класс Player

Описание: Отвечает за игрока, его управление и движение.

Методы:

- move() – логика движения игрока.
- keyPressed(KeyEvent e) – обработка нажатия клавиш управления.
- keyReleased(KeyEvent e) – обработка отпускания клавиш.
- getSpeed(), getX(), getY(), getLayer1(), getLayer2() – геттеры для получения характеристик игрока.

5. Класс Enemy

Описание: Обычный вражеский автомобиль, движущийся навстречу игроку.

Методы:

- Enemy(int x, int y, int speed, Road road) – конструктор, принимающий координаты, скорость и объект дороги.
- move() – обновляет положение врага.
- getRect() – возвращает границы объекта для проверки коллизии.

6. Класс EnemyReversed

Описание: Противник, движущийся в одном направлении с игроком.

Методы:

- EnemyReversed(int x, int y, int speed, Road road) – конструктор, аналогичный Enemy.
- move() – обновляет положение врага.
- getRect() – возвращает границы объекта.

Полученная диаграмма классов представлена ниже на рисунке 1.

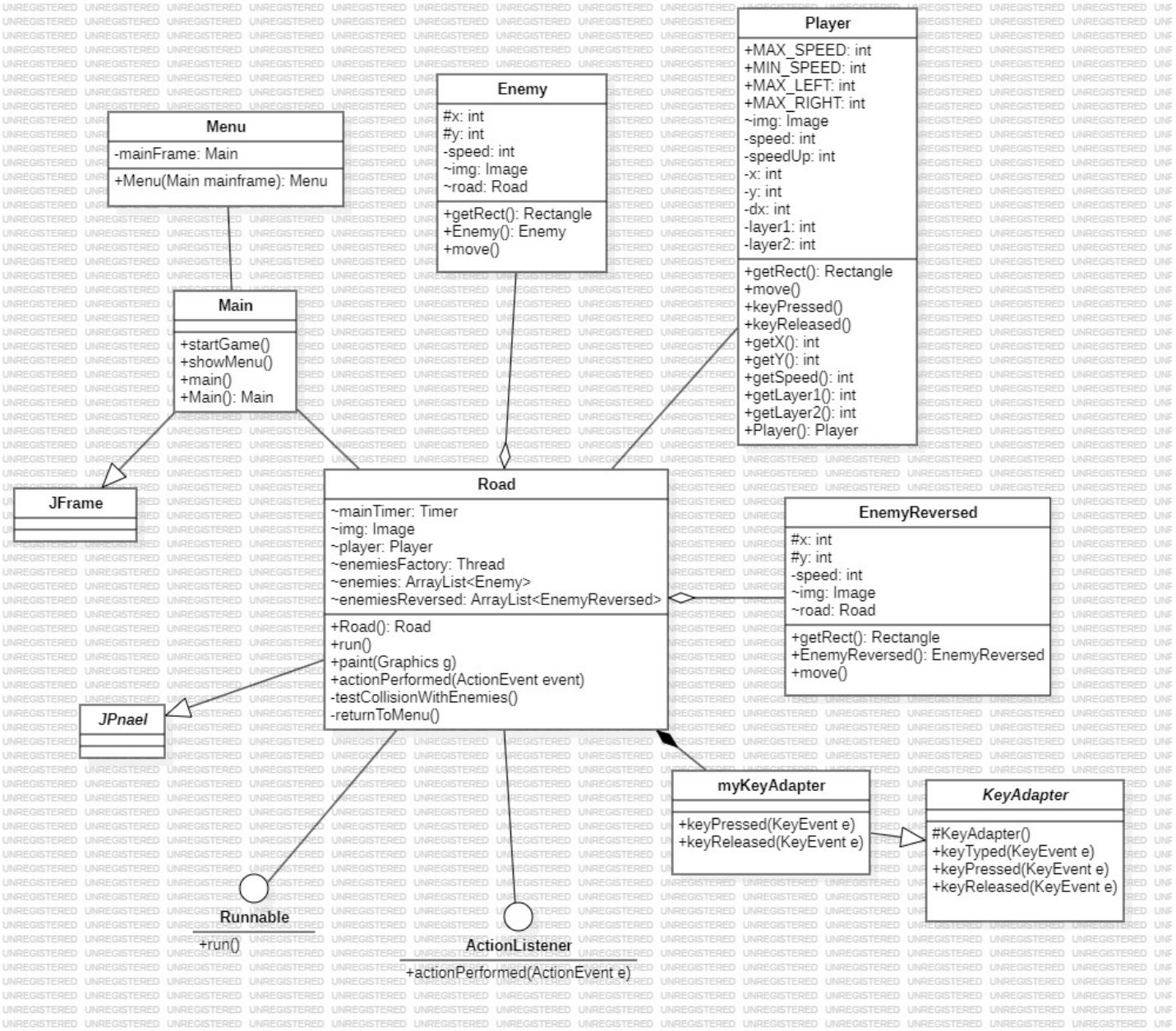

Рисунок 1 - Диаграмма классов

РАЗРАБОТКА ОСНОВНОГО АЛГОРИТМА ПРОГРАММЫ

Игра построена на Swing и работает в событийном цикле с таймером (Timer), который отвечает за обновление экрана и обработку логики. Разберем алгоритм работы игры пошагово.

1. Запуск игры и отображение главного меню

1. Программа запускается с main() в классе Main.
2. В main() создается объект класса Main, который наследуется от JFrame (главное окно игры).
3. В конструкторе Main() вызывается showMenu(), который устанавливает в качестве содержимого (setContentPane()) панель Menu.
4. В Menu создается кнопка "Играть", которая загружает изображение кнопки (play_button.png).
5. При нажатии на кнопку "Играть" вызывается метод startGame() в Main, который:
    - Создает объект Road (игровое поле).
    - Устанавливает Road как основную панель (setContentPane()).
    - Обновляет окно (revalidate(), repaint()).
    - Передает фокус на Road, чтобы обрабатывать нажатия клавиш.

2. Запуск игрового процесса

При загрузке Road происходит:

1. Запускается главный таймер (mainTimer**)**, который вызывает actionPerformed() каждые 20 мс.
2. Запускается поток enemiesFactory, который каждую случайную величину времени (до 2 секунд) добавляет новых врагов (Enemy и EnemyReversed) в списки enemies и enemiesReversed.
3. Добавляется обработчик клавиатуры MyKeyAdapter, который отслеживает нажатия и отпускания клавиш для управления машиной.

3. Основной игровой цикл

Каждые 20 мс выполняются следующие шаги в методе actionPerformed():

1. Обновляется движение машины игрока (player.move()).
    - Скорость регулируется клавишами W / S (UP / DOWN).
    - Движение влево / вправо A / D (LEFT / RIGHT).
    - Координаты игрока обновляются, но сама машина остается на

месте — вместо нее двигаются слои дороги (layer1 и layer2).

2. Обновляется движение врагов (Enemy, EnemyReversed).
    - Они двигаются по вертикали со скоростью, зависящей от скорости игрока.
    - Если враг выходит за пределы экрана (y > 650 или y < -660), он удаляется из списка.
3. Проверяется коллизия игрока с врагами (testCollisionWithEnemies()).
    - Используется метод intersects() для проверки пересечения прямоугольников (Rectangle).
    - Если коллизия обнаружена:
        - Выводится сообщение "Вы проиграли!"
        - Игра возвращается в меню (returnToMenu()).
4. Вызывается repaint(), чтобы обновить экран.
    - Метод paint(Graphics g):
        - Рисует фон дороги (road_top.png).
        - Отображает машину игрока.
        - Отображает врагов.
        - Отображает скорость в км/ч.

4. Завершение игры и возврат в меню

Если игрок сталкивается с врагом:

1. Вызывается JOptionPane.showMessageDialog(null, "Вы проиграли!").
2. Главный таймер mainTimer останавливается.
3. Вызывается returnToMenu(), который устанавливает Menu как главное содержимое окна.

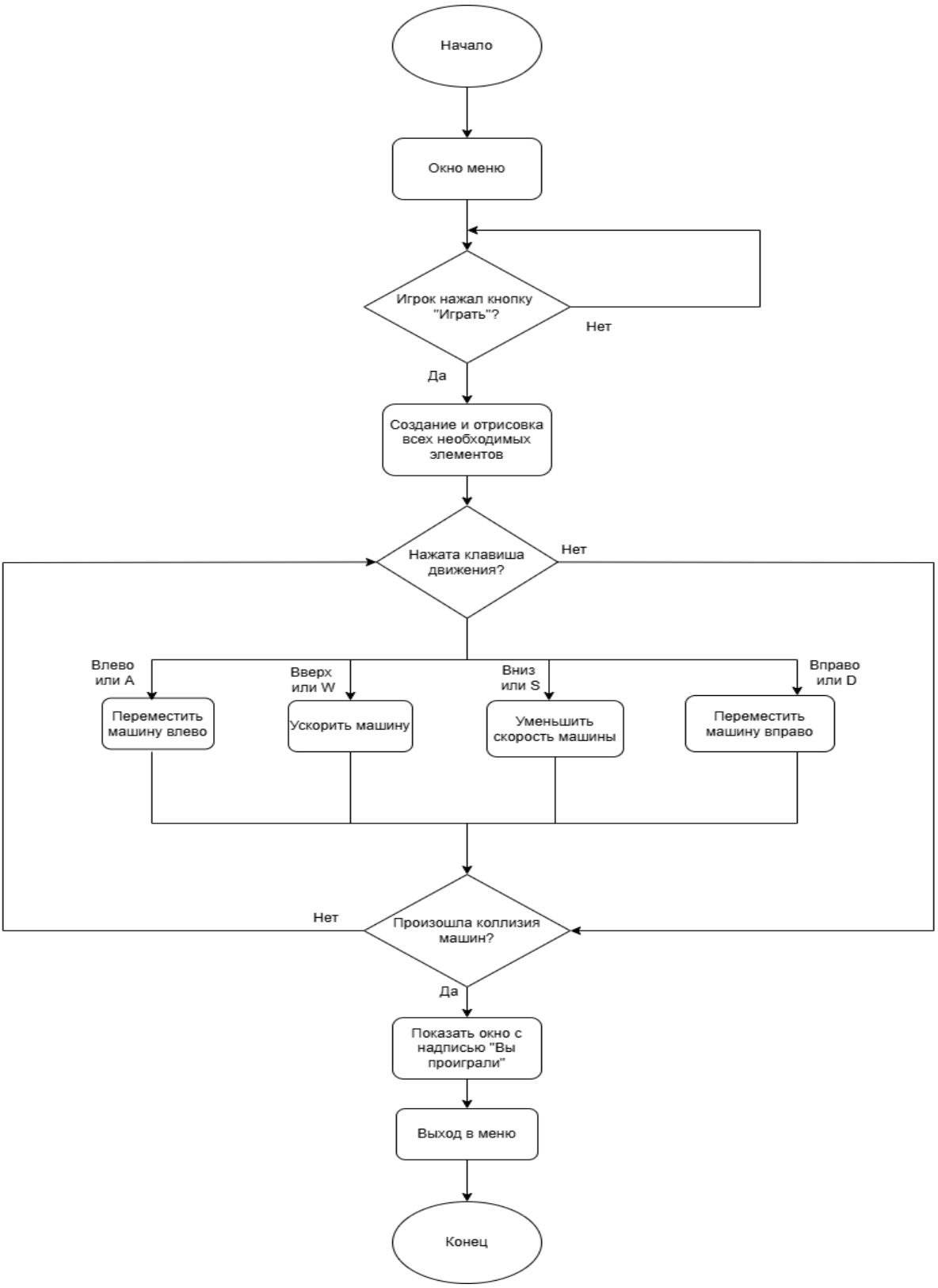

Рисунок 2 – Блок-схема основного алгоритма

ПРОЕКТИРОВАНИЕ И ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Интерфейс включает в себя меню с одной кнопкой: играть, при нажатии на которую, игра запускается.

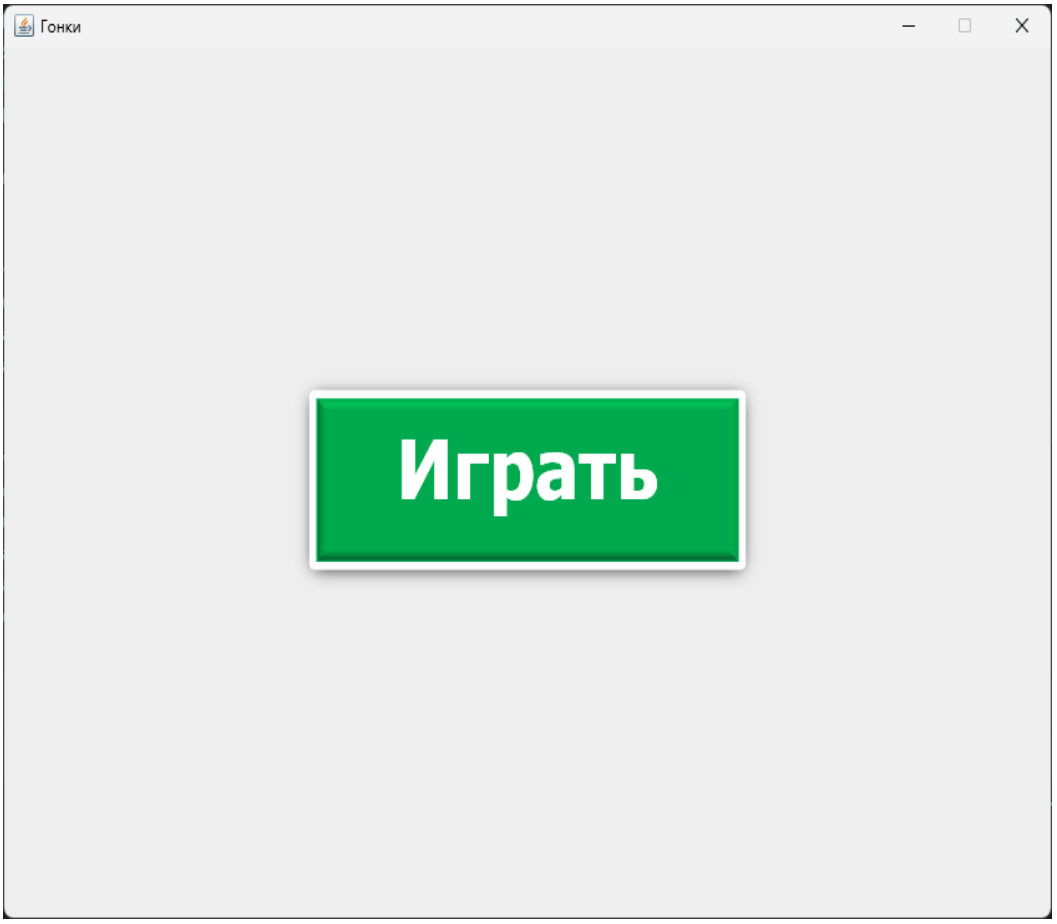

Рисунок 3 – Окно меню

У пользователя есть выбор, начать игру или подождать. При запуске игры появляется машина игрока и машины противников, что наглядно представлено на рисунке 4. Управление производится стрелками на клавиатуре, то есть клавишами стрелок или клавишами WASD.

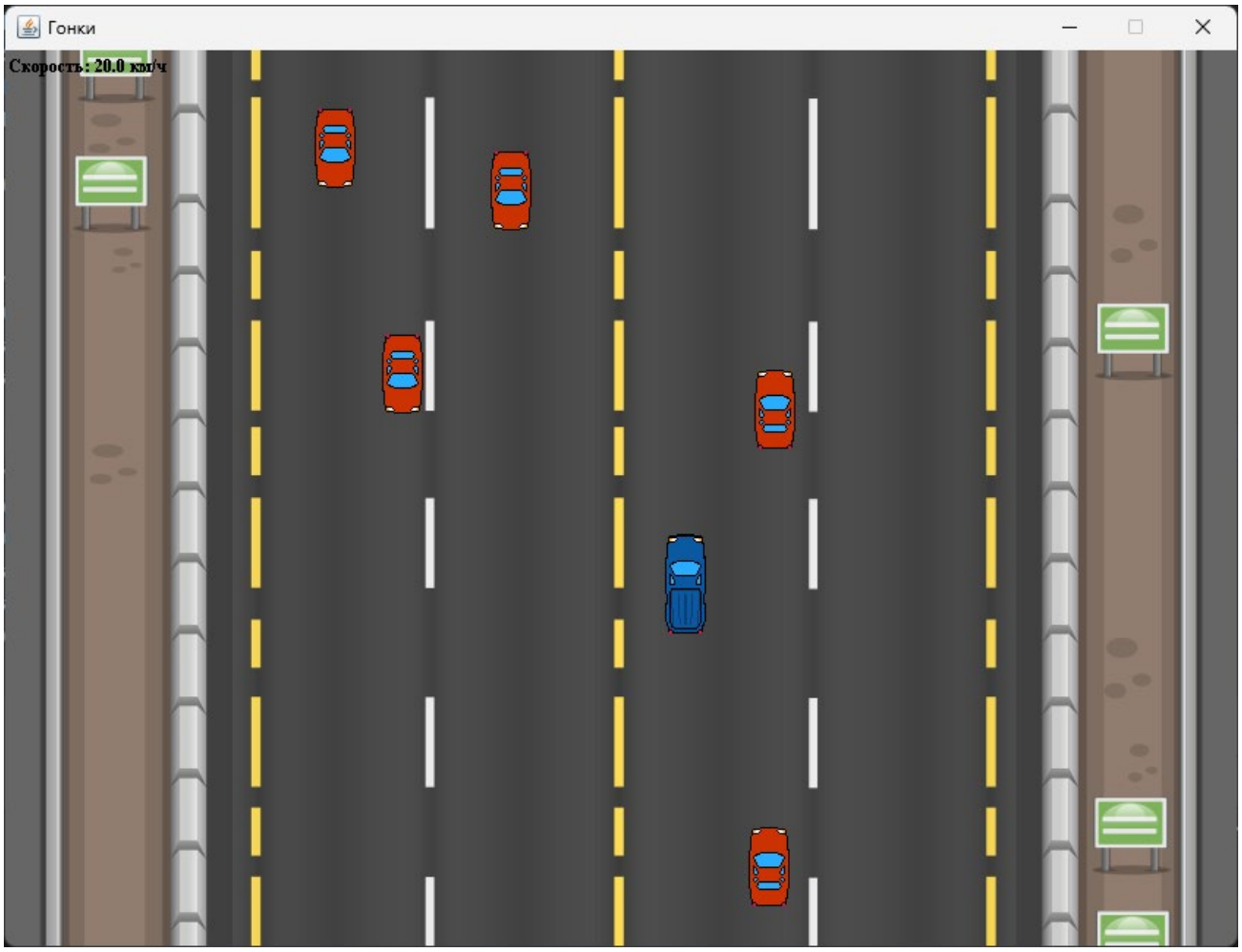

Рисунок 4 – Вид игры

Когда машина игрока врезается в машину противника, игра завершается, вылезает соответствующее окно и затем игрок выходит в меню. Интерфейс экрана поражения представлен на рисунке 5.

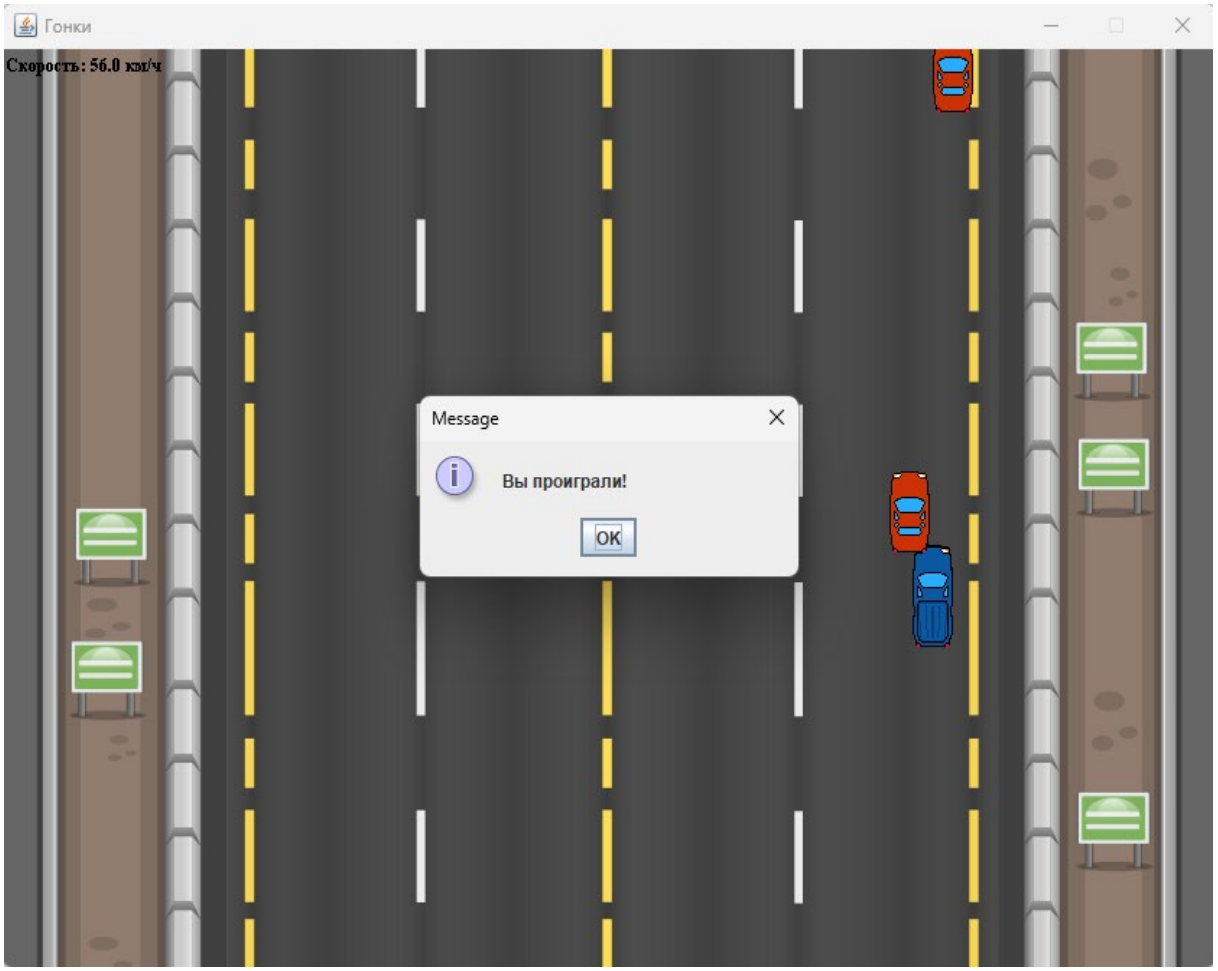

Рисунок 5 – Завершение игры

РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

В качестве инструментария была выбрана среда разработки JetBrains IntelliJ IDEA (https://www.jetbrains.com/idea/).

Для проверки корректности работы программы были разработаны тестовые задачи, в которых проверялись все возможности игры в жанре «Гонки».

В результате тестирования было выявлено, что программа корректно работает и выполняет все заданные операции.

ЗАКЛЮЧЕНИЕ

В заключение хочется сказать, что в процессе выполнения курсовой работы были закреплены практические навыки программирования на языке Java, описана и проанализирована поставленная задача, изучены и реализованы правила игры в жанре «Гонки», построена диаграмма классов, обоснован выбор концепций и механизмов ООП, спроектирован пользовательский интерфейс, реализовано и протестировано готовое программное средство.

СПИСОК ЛИТЕРАТУРЫ

Хорстманн Кей Java. Библиотека профессионала [Книга]. - [б.м.] : Диалектика-Вильямс, 2020. - Т. 1.

Романчик Валерий Станиславович Java. Методы программирования [Книга]. - [б.м.] : Четыре четверти, 2016.

Шилдт Герберт Java. Руководство для начинающих. Современные методы создания, компиляции и выполнения программ на Java [Книга]. - [б.м.] : Диалектика-Вильямс, 2018.

Берд Барри Java для чайников [Книга]. - [б.м.] : Диалектика-Вильямс, 2019.

[В Интернете] // Wikipedia. - https://ru.wikipedia.org/wiki/Тетрис.

ПРИЛОЖЕНИЕ 1 (ЛИСТИНГ КОДА)

Класс Main:

```
import javax.swing.*;

public class Main extends JFrame {
    public Main() {
        setTitle("Гонки");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(856, 650);
        setLocationRelativeTo(null);
        setResizable(false);

        showMenu();
    }

    public void showMenu() {
        setContentPane(new Menu(this));
        revalidate();
        repaint();
    }

    public void startGame() {
        Road road = new Road();
        setContentPane(road);
        revalidate();
        repaint();
        road.requestFocusInWindow(); // принудительно передаем фокус на Road
    }


    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new Main().setVisible(true);
        });
    }
}
```

Класс Enemy:

```
import javax.swing.*;
import java.awt.*;
```

```
public class Enemy {
    protected int x;
    protected int y;
    private int speed;

    Image img = new ImageIcon("res/car-truck1.png").getImage();
    Road road;

    public Rectangle getRect(){
        return new Rectangle(x, y, 28, 54);
    }

    public Enemy(int x, int y, int speed, Road road){
        this.x = x;
        this.y = y;
        this.road = road;
        this.speed = speed;
    }

    public void move(){
        y = y+ road.player.getSpeed() - speed;
    }
}
```

Класс EnemyReversed:

```
import javax.swing.*;
import java.awt.*;

public class EnemyReversed {
    protected int x;
    protected int  y;
    private int speed;

    Image img = new ImageIcon("res/enemy_car_reversed.png").getImage();
    Road road;

    public Rectangle getRect(){
        return new Rectangle(x, y, 28, 54);
    }

    public EnemyReversed(int x, int y, int speed, Road road){
        this.x = x;
        this.y = y;
        this.road = road;
```

```
        this.speed = speed;
    }

    public void move(){
        y = y+ road.player.getSpeed() + speed;
    }
}
```

Класс Road:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Random;

public class Road extends JPanel implements ActionListener, Runnable {

    Timer mainTimer = new Timer(20, this); //выполнение функции actionPerformed
каждые 20 ms

    Image img = new ImageIcon("res/road_top.png").getImage(); //загрузка изображений

    Player player = new Player();

    Thread enemiesFactory = new Thread(this);

    //список вражеских машин
    ArrayList<Enemy> enemies = new ArrayList<Enemy>();
    ArrayList<EnemyReversed> enemiesReversed = new ArrayList<EnemyReversed>();

    public Road (){
        mainTimer.start();
        enemiesFactory.start();
        addKeyListener(new MyKeyAdapter());
        setFocusable(true); //делаем дорогу в фокусе, чтобы все нажатия клавиш
обрабатывались
    }

    @Override
    public void run() {
```

```
        while (true){
            Random random = new Random();
            try {
                Thread.sleep(random.nextInt(2000));
                enemies.add(new Enemy(random.nextInt(422, 650), -100, random.nextInt(1,
15),this));
                enemiesReversed.add(new EnemyReversed(random.nextInt(185, 390), -100,
random.nextInt(1, 15),this));
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }

    private class MyKeyAdapter extends KeyAdapter{
        /*реализация слушателя, наследуемся от класса
        при нажатии клавиш будут происходить соответствующие движения машины
         */
        public void keyPressed(KeyEvent e){
            player.keyPressed(e);
        }

        public void keyReleased(KeyEvent e){
            player.keyReleased(e);
        }
    }

    //рисуем дорогу
    @Override
    public void paint(Graphics g){
        g = (Graphics2D) g;
        g.drawImage(img, 0, (-1) * player.getLayer1(), null); //движение, машина стоит на
месте, слои дороги двигаются вниз
        g.drawImage(img, 0, (-1) * player.getLayer2(), null);
        g.drawImage(player.img, player.getX(), player.getY(), null);

        //вывод спидометра машины
        double speed = (200 / Player.MAX_SPEED) * player.getSpeed();
        //делаем шрифт
        g.setColor(Color.BLACK);
        Font font = new Font("Times New Roman", Font.BOLD, 12);
        g.setFont(font);
        g.drawString("Скорость: " + speed + " км/ч", 2, 15);
```

```
        Iterator<Enemy> i = enemies.iterator();
        Iterator<EnemyReversed> iReversed = enemiesReversed.iterator();
        while (i.hasNext() && iReversed.hasNext()){
            Enemy e = i.next();
            EnemyReversed enemyReversed = iReversed.next();
            if ((e.y >= 650 || e.y <= -660) && (enemyReversed.y >= 650 || enemyReversed.y <= -
660)){
                i.remove();
                iReversed.remove();
            }
            else {
                e.move();
                enemyReversed.move();
                g.drawImage(e.img, e.x, e.y, null);
                g.drawImage(enemyReversed.img, enemyReversed.x, enemyReversed.y, null);
            }
        }
    }

    public void actionPerformed(ActionEvent event){
        player.move();
        repaint();
        testCollisionWithEnemies();
    }

    private void testCollisionWithEnemies() {
        Iterator<Enemy> i = enemies.iterator();
        Iterator<EnemyReversed> iReversed = enemiesReversed.iterator();

        while (i.hasNext() && iReversed.hasNext()){
            Enemy e = i.next();
            EnemyReversed enemyReversed = iReversed.next();
            //пересечение прямоугольников, то есть проверка коллизии
            if (player.getRect().intersects(e.getRect())){
                JOptionPane.showMessageDialog(null, "Вы проиграли!");
                returnToMenu();
            }
            else if (player.getRect().intersects(enemyReversed.getRect())){
                JOptionPane.showMessageDialog(null, "Вы проиграли!");
                returnToMenu();
            }
        }
    }

    private void returnToMenu() {
```

```
        mainTimer.stop();

        JFrame parentFrame = (JFrame) SwingUtilities.getWindowAncestor(this);
        if (parentFrame instanceof Main) {
            ((Main) parentFrame).showMenu();
        }
    }

}
```

Класс Player:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.KeyEvent;

public class Player {

    public static final int MAX_SPEED = 50;
    public static final int MIN_SPEED = 5;
    public static final int MAX_LEFT = 139;
    public static final int MAX_RIGHT = 677;

    Image img = new ImageIcon("res/car-truck3.png").getImage();

    private int speed = 3;
    private int speedUp = 0;

    //координаты машины в начальный момент
    private int x = 450;
    private int y = 330;
    private int dx = 0;

    private int layer1 = 0; //координата первого слоя
    private int layer2 = 650;

    public Rectangle getRect(){
        return new Rectangle(x, y, 28, 69);
    }

    public void move() {
        speed += speedUp;
        //Ограничения на выезды за пределы дороги и на реверсивную езду
        if (speed <= 0) speed = 0;
        if (speed <= MIN_SPEED) speed = MIN_SPEED;
```

```
    if (speed >= MAX_SPEED) speed = MAX_SPEED;
    if (x >= MAX_RIGHT) x = MAX_RIGHT;
    if (x <= MAX_LEFT) x = MAX_LEFT;
    x -= dx;
    //если второй слой закончился, то возвращаем слои на исходную
    if (layer2 - speed <= 0) {
        layer1 = 0;
        layer2 = 650;
    } else {
        layer1 -= speed; //слой двигается
        layer2 -= speed;
    }
}

public void keyPressed(KeyEvent e) {
    int key = e.getKeyCode();
    if (key == KeyEvent.VK_UP || key == KeyEvent.VK_W) {
        speedUp = 1;
    }
    if (key == KeyEvent.VK_DOWN || key == KeyEvent.VK_S) {
        speedUp = -1;
    }
    if (key == KeyEvent.VK_LEFT || key == KeyEvent.VK_A) {
        dx = 5;
    }
    if (key == KeyEvent.VK_RIGHT || key == KeyEvent.VK_D) {
        dx = -5;
    }
}

public void keyReleased(KeyEvent e) {
    int key = e.getKeyCode();
    if (key == KeyEvent.VK_UP || key == KeyEvent.VK_W) {
        speedUp = 0;
    }
    else if (key == KeyEvent.VK_DOWN || key == KeyEvent.VK_S) {
        speedUp = 0;
    }
    if (key == KeyEvent.VK_LEFT || key == KeyEvent.VK_A) {
        dx = 0;
    }
    if (key == KeyEvent.VK_RIGHT || key == KeyEvent.VK_D) {
        dx = 0;
    }
}
```

```
    public int getSpeed() {
        return speed;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public int getLayer1() {
        return layer1;
    }

    public int getLayer2() {
        return layer2;
    }

}
```

Класс Menu:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Menu extends JPanel {
    private Main mainFrame;

    public Menu(Main mainFrame) {
        this.mainFrame = mainFrame;
        setLayout(new BorderLayout());

        // Загружаем изображение кнопки
        ImageIcon playIcon = new ImageIcon("res/play_button.png");

        JButton playButton = new JButton(playIcon);
        playButton.setBorderPainted(false); // Убираем рамку
        playButton.setContentAreaFilled(false); // Убираем фон
        playButton.setFocusPainted(false); // Убираем рамку фокуса
        playButton.setOpaque(false); // Делаем прозрачным
```

```
        playButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                mainFrame.startGame();
            }
        });

        add(playButton, BorderLayout.CENTER);
    }
}
```