Moving Vehicles Application

Magyar Roland

June 2024

1 Introduction

This document serves as a comprehensive guide to understanding, installing, and utilizing the Moving Vehicles Application developed in Java.

2 Overview

The application is a graphical user interface (GUI) application designed to simulate the movement of vehicles on a virtual canvas. It provides users with an interactive platform to add vehicles, control their movement, and visualize their positions in real-time.

3 Features

- Add Vehicles: Users can add new vehicles with specified attributes such as name, position, direction, and speed.
- Auto Add Vehicles: Users can automatically add vehicles with random attributes.
- Move Vehicles: Users can control the movement of vehicles in four directions (up, down, left, right).
- Vehicle List: A separate window displays a list of all vehicles, allowing users to select and modify vehicle parameters.
- Random Colors: Vehicles are displayed in random colors to easily distinguish between them.
- Bounded Movement: Vehicles cannot move outside the bounds of the window.

4 Requirements

4.1 Functional Requirements

- Add Vehicles: Users should be able to add new vehicles by specifying their name, initial coordinates (x, y), direction, and speed. Multiple vehicles can be added at once.
- Display Vehicles: Vehicles should be displayed on the canvas with distinct colors and updated in real-time.
- Move Vehicles: Users should be able to move selected vehicles in four directions: up, down, left, and right.
- Vehicle List: A separate window should display a list of all vehicles with their names and parameters.
- Modify Vehicle Parameters: Users should be able to select a vehicle from the list and update its parameters.
- Boundary Check: Vehicles should not move outside the canvas boundaries.

4.2 Non-Functional Requirements

- Performance: Application should respond to user inputs within 100ms and handle at least 50 vehicles simultaneously.
- Usability: Interface should be intuitive and easy to navigate.
- Reliability: Application should not crash during normal operation and handle errors gracefully.
- Maintainability: Code should be well-documented and modular.
- Portability: Application should run on any system with Java 8 or higher.
- Scalability: Application should be able to support future enhancements and a larger number of vehicles.

5 Installation Instructions

1. Download the Source Code:
    - Download from here.
2. Set Up Development Environment:
    - Install the JDK if not already installed.
    - Set up an Integrated Development Environment (IDE) such as IntelliJ IDEA, Eclipse, or NetBeans.

3. Import the Project:
    - Open your IDE and import the project by selecting the appropriate option (e.g., "Import Project" in IntelliJ IDEA).
4. Build and Run:
    - Build the project using the IDE's build tools. Run the Main.java class to start the application.

6 User Guide

6.1 Adding a Vehicle

1. Open the Application: Launch the application by running the Main.java class.
2. Add Vehicle:
    - Click the "Add Vehicle" button.
    - Fill in the vehicle parameters in the dialog box that appears (name, x, y, direction, speed, and number of vehicles).
    - Click "OK" to add the vehicle(s) to the canvas.

6.2 Moving a Vehicle

1. Select Vehicle: Click on a vehicle in the list displayed in the vehicle list window.
2. Move Vehicle:
    - Use the "Up", "Down", "Left", and "Right" buttons to move the selected vehicle in the respective direction.
    - The vehicle's position will update in real-time on the canvas.

7 Architecture

7.1 Component Overview

The main components include the MovingVehicleGUI, MovingVehiclePanel, Vehicle, and VehicleListWindow.

7.2 Block Diagram

The block diagram (fig 1) illustrates the overall flow and interactions between different operations in the Moving Vehicles Application:

Vehicle Informations: This is the starting point where the user inputs the necessary information for the vehicle(s) such as name, initial coordinates (x, y), direction, and speed.

Create Car: Based on the provided information, the application creates a new vehicle object. This object is then added to the list of vehicles in the application.

Car is Moving: Once created, the vehicle starts moving on the canvas according to its initial parameters (direction and speed).

Modify Parameters: Users can select a vehicle and modify its parameters, such as changing its direction or speed.

Car is Moving with Different Direction/Speed: After modifying the parameters, the vehicle continues to move but now follows the new direction and speed settings provided by the user.

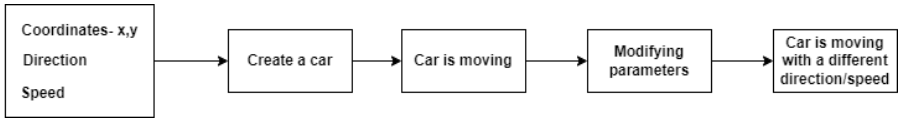

Figure 1: Architecture of the project

7.3 Sequence Diagrams

7.3.1 Adding a Vehicle

The following sequence diagram (fig 2) illustrates the process of adding a vehicle to the application.

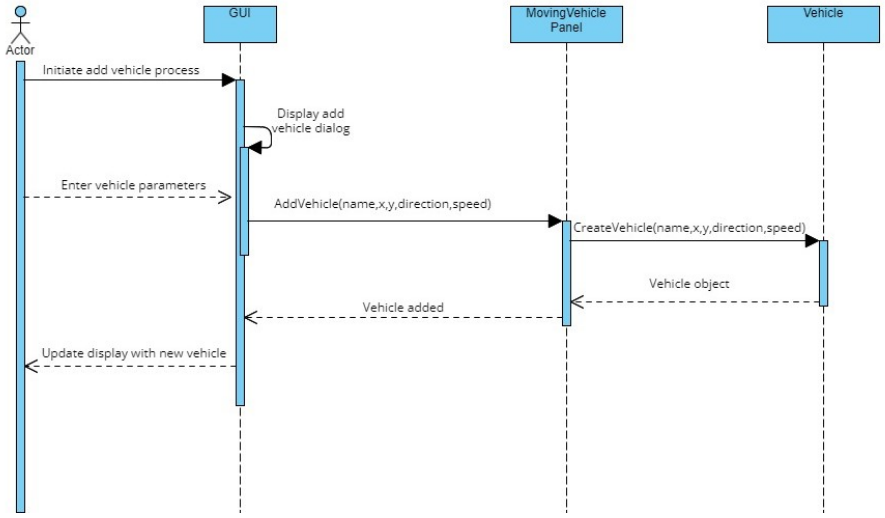

Figure 2: Add vehicle diagram

7.3.2 Moving a Vehicle

The following sequence diagram (fig 3)illustrates the process of moving a vehicle in a specific direction.

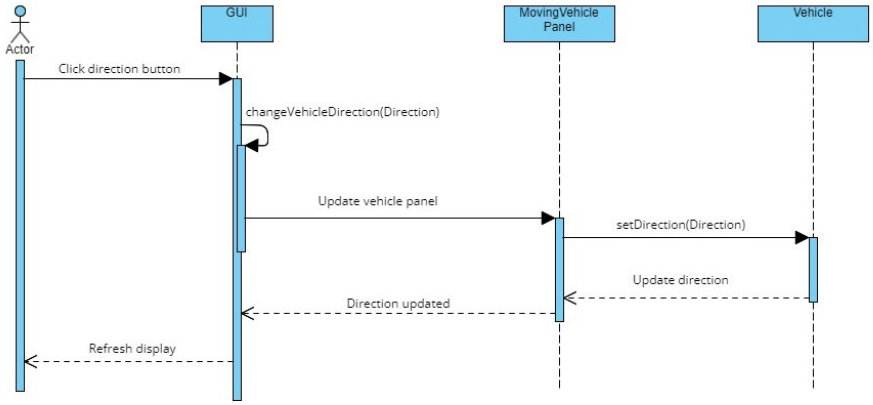

Figure 3: Direction change diagram

8 Class Overview

8.1 Main

- Contains the main method, which is executed when the application is launched.
- Initializes the main graphical user interface (GUI) by creating an instance of MovingVehicleGUI.
- Starts the application and displays the GUI to the user.

8.2 MovingVehicleGUI

- Responsible for managing the main graphical user interface of the application.
- Handles user interactions such as adding vehicles, modifying parameters, and moving vehicles.
- Interacts with other components such as MovingVehiclePanel and VehicleListWindow to update the UI.

8.3 MovingVehiclePanel

- Represents the canvas where vehicles are displayed and their movements are simulated.
- Manages the rendering of vehicles and their movements.
- Provides methods for adding vehicles, updating their positions, and performing boundary checks.

8.4 Vehicle

- Represents Represents a single vehicle in the application.
- Stores information such as name, coordinates (x, y), direction, and speed of the vehicle.
- Provides methods for updating the vehicle's position, changing its direction, and modifying its parameters.

8.5 VehicleListWindow

- Represents Displays a list of all vehicles currently present in the application.
- Allows users to select a vehicle from the list and modify its parameters.
- Keeps track of changes to the vehicle list and updates the UI accordingly.

9 Conclusion

I am pleased to announce that I have successfully developed an application featuring moving vehicles. Through planning and implementation, we have created a dynamic and interactive platform that allows users to simulate vehicle movements, modify parameters, and visualize their positions in real-time.