Traffic Management System - Detailed Overview

1. Project Overview

This Traffic Management System is a web-based interactive simulation built using HTML, CSS, and JavaScript. It showcases fundamental concepts in data structures and algorithms through three distinct modules: Shortest Route Finding, Collision Prevention in a traffic network, and Traffic Light Management at an intersection. Each module provides a visual representation and interactive controls to explore these concepts.

This System serves as a practical demonstration of several key concepts learned in a Data Structures and Algorithms course. By visualizing pathfinding, spatial data structures, and simulation techniques, it provides a hands-on understanding of how these principles can be applied to real-world problems like traffic management and simulation. The choice of algorithms like A\\* and the use of data structures like the Quad-tree highlight the importance of efficient data organization and algorithmic design in solving complex computational tasks.

2. Project Structure and File Descriptions

The project directory contains the following files:

* **`index.html`:** The main entry point of the application. It provides the user interface for navigating between the different simulation modules (Shortest Route and Collision Prevention). The Traffic Light Management module is accessed via a link to `TL.html`.
* **`style.css`:** Contains the primary Cascading Style Sheets rules for the overall layout, navigation, and the Shortest Route and Collision Prevention sections. It ensures a consistent visual appearance across these modules.
* **`script.js`:** Houses the JavaScript code for the interactive elements and algorithms within the Shortest Route Finder and the Collision Prevention System. This includes grid manipulation, pathfinding logic (A\\* and potentially Dijkstra), vehicle movement, and basic collision detection using a quad-tree.
* **`TL.html`:** A separate HTML file dedicated to the Traffic Light Management simulation. It sets up the canvas and control buttons for this specific module.
* **‘TL.css`:** Contains the CSS rules specifically for styling the Traffic Light Management interface, including the canvas and buttons.
* **‘TL.js`:** Implements the JavaScript logic for the Traffic Light Management simulation. This includes drawing the intersection, managing vehicle movement based on traffic light states, cycling the traffic lights using a timer, and detecting basic collisions at the intersection.

3. Module Details and Usage

3.1. Shortest Route Finder

**Description:** This module allows users to interact with a grid to find the shortest path between a designated start and end point, while avoiding user-placed obstacles. It visually demonstrates the A\* search algorithm.

**Implementation Details:**

* The grid is dynamically created using JavaScript based on the user's selection from the "Grid Size" dropdown.
* Users can click on grid cells to define the start point (green), end point (red), and obstacles (gray).
* The "Find Shortest Path" button triggers the A\* algorithm implemented in `script.js`.
* The A\* algorithm uses a heuristic function (Manhattan distance is likely used) to estimate the cost to the goal from each cell.
* It maintains open and closed sets of nodes, evaluating neighbors and their costs (g-score: cost from start, h-score: heuristic to goal, f-score: g + h).
* The shortest path is reconstructed by backtracking from the end node to the start node, following the parent pointers set during the search.
* The calculated shortest path is displayed on the grid in blue.
* The "Reset Grid" button clears the grid, allowing the user to define a new scenario.

**Usage:** Follow the instructions within the "Shortest Route Finder" section on the `index.html` page.

3.2. Collision Prevention System

**Description:** This module simulates vehicle movement on a basic road network, including roundabouts, and implements a rudimentary collision prevention system. It demonstrates the use of a quad-tree for efficient spatial querying to detect potential collisions.

**Implementation Details:**

* The "Create Network" button in `script.js` generates a grid-based road network on the canvas. Some intersections are configured as roundabouts with specific connection logic to simulate circular flow.
* The `RoadNetwork` class (likely in `script.js`) manages nodes (intersections) and edges (roads) and provides a `draw` method.
* The "Algorithm" dropdown allows selection between Dijkstra's and A\\* algorithms for pathfinding (implemented within the `Vehicle` class in `script.js`).
* The "Add Vehicle" button creates a new `Vehicle` object with a random start and end node and calculates a path using the selected algorithm.
* The `Vehicle` class in `script.js` handles movement along the calculated path, updating position and angle.
* A `QuadTree` data structure (implemented in `script.js`) is used to efficiently query nearby vehicles for collision detection. Each vehicle's bounding box is inserted into the quad-tree.
* The `detectCollisions` function in `script.js` queries the quad-tree for neighbors of each vehicle and checks for overlaps in their bounding boxes (Axis-Aligned Bounding Box or AABB intersection).
* Collision prevention is currently implemented by setting the speed of the colliding vehicles to zero temporarily.
* The simulation loop (`updateCollisionSimulation` in `script.js`) updates vehicle positions, detects collisions, redraws the scene on the canvas, and decreases traffic values on road segments over time.

**Usage:** Follow the instructions within the "Collision Prevention System" section on the `index.html` page.

3.3. Traffic Light Management

**Description:** This module simulates vehicles moving through a four-way intersection controlled by a timed traffic light system. It demonstrates basic traffic flow management and collision detection at an intersection.

**Implementation Details:**

* The `TL.html` file sets up a canvas where the intersection and vehicles are drawn.
* The `TL.js` file contains the simulation logic.
* The `drawMap()` function draws the road layout of the four-way intersection.
* The `Vehicle` class (defined in `TL.js`) represents individual cars with properties like position, direction, lane, and speed.
* Vehicles are added to the simulation with random lanes and directions using `addVehicles()`. The number of vehicles per lane is limited by `MAX\_VEHICLES\_PER\_LANE`.
* The `drawTrafficLights()` function visualizes the traffic lights (red for stop, green for go) for each direction (horizontal left, horizontal right, vertical up, vertical down).
* The `trafficLights` object in `TL.js` stores the current state of each traffic light.
* The `switchTrafficLights()` function cycles the traffic lights based on a `timer`. It alternates between allowing horizontal and vertical traffic flow for a set duration (5 seconds in this implementation).
* Vehicles approaching the intersection check the state of the corresponding traffic light in their direction. They stop if the light is red and proceed if it's green.
* The `checkCollisions()` function in `TL.js` detects collisions between vehicles by checking for overlaps in their bounding boxes. A basic collision response is implemented to move colliding vehicles slightly apart.
* The `updateSimulation()` function in `TL.js` clears the canvas, redraws the map, traffic lights, and vehicles, moves the vehicles, and checks for collisions in each animation frame.
* Buttons for "Start," "Pause," and "Reset" control the simulation flow.

**Usage:** Open `TL.html` in your browser and use the control buttons provided.

4. Technologies Used

* HTML5 for the structure of the web pages.
* CSS3 for styling the visual elements and layout (`style.css` and `TL.css`).
* JavaScript (ES6+) for implementing the interactive behavior, algorithms (A\*, potentially Dijkstra), simulation logic, and canvas manipulation (`script.js` and `TL.js`).

5. Data Structures and Algorithms Demonstrated

This project showcases the following concepts:

* Graph Traversal Algorithms:

The Shortest Route Finder likely implements the **A\* search algorithm**(and potentially **Dijkstra's algorithm** as an option in the Collision Prevention system) for finding the shortest path in a grid or road network.

* Spatial Partitioning:

The Collision Prevention System utilizes a **Quad-tree** to efficiently organize and query vehicles based on their spatial location, optimizing the collision detection process.

* Simulation Techniques:

Both the Collision Prevention and Traffic Light Management modules employ simulation loops to model the movement and interaction of vehicles over time.

* Basic Collision Detection:

The project implements simple Axis-Aligned Bounding Box (AABB) intersection to detect collisions between vehicles in both the Collision Prevention and Traffic Light Management modules.

* Heuristic Functions:

The A\* algorithm in the Shortest Route Finder uses a heuristic function (likely Manhattan distance) to guide the search towards the goal.

* Graph Representation:

The road network in the Collision Prevention system can be seen as a graph where intersections are nodes and roads are edges.

6. Potential Future Enhancements

* Advanced Road Network Generation and Editing: Allow users to design and modify the road network in the Collision Prevention module with more complex layouts and features.
* Improved Collision Avoidance: Implement more sophisticated collision avoidance behaviours, such as deceleration, steering, or lane changing, instead of just stopping.
* Detailed Traffic Analysis: Add features to collect and display traffic statistics like average speed, congestion levels, and waiting times.
* Adaptive Traffic Light Control: Implement traffic light algorithms that dynamically adjust the green light duration based on real-time traffic flow.
* Enhanced Visualization: Improve the visual representation of vehicles, traffic flow, and the road environment.

**User Interface Improvements:** Refine the user interface for better interactivity and user experience across all modules.

**Integration of Modules:** Explore the possibility of integrating the different modules, for example, having vehicles from the Collision Prevention system interact with traffic lights.