Relatório Técnico — Simulação Inteligente de Fluxo de Tráfego (Baseado em SmartTrafficSimulation)

Feito por:
Afonso Fabiana(30014511)
Gonçalo Marques(30012478)
Gonçalo Pardelhas(30014665)

Relatório Técnico — Simulação Inteligente de Fluxo de Tráfego

1. Introdução

O aumento contínuo do tráfego urbano tem vindo a colocar desafios significativos à mobilidade, segurança e eficiência dos sistemas rodoviários. Problemas como congestionamento, tempos de espera elevados e dificuldades na circulação de veículos de emergência evidenciam a necessidade de soluções de gestão de tráfego mais inteligentes e adaptativas.

Neste contexto, o presente projeto consiste no desenvolvimento de uma **simulação inteligente de tráfego urbano**, implementada em Java, cujo principal objetivo é modelar de forma realista a interação entre **veículos**, **peões** e **semáforos**, bem como avaliar o impacto de **estratégias inteligentes de controlo de tráfego**, em especial na prioridade dada a veículos de emergência.

O sistema foi concebido recorrendo a princípios de **Programação Orientada a Objetos (OOP)**, à arquitetura **Model–View–Controller (MVC)** e à aplicação de **padrões de design** como *State* e *Strategy*. A componente gráfica foi desenvolvida com **JavaFX**, permitindo a observação visual do funcionamento da simulação e a análise empírica dos resultados obtidos.

2. Arquitetura do Sistema

A aplicação adota a arquitetura **Model–View–Controller (MVC)**, que promove

a separação clara entre a lógica do domínio, a apresentação gráfica e o controlo da aplicação. Esta abordagem facilita a manutenção do código, melhora a legibilidade e permite a evolução do sistema sem comprometer as restantes componentes.

2.1 Camada Model

A camada **Model** representa o estado do mundo simulado e contém todas as entidades que nele existem. A classe central desta camada é `World`, que atua como contentor global de todas as entidades da simulação, tais como estradas, veículos, semáforos e peões. Esta classe é responsável pela criação, atualização e remoção dos objetos à medida que entram ou saem do campo de visão.

As principais entidades desta camada incluem:

- **Vehicle** (classe abstrata): define as propriedades físicas comuns a todos os veículos, como posição, velocidade, aceleração e distância de segurança;
- **Road**: modela as vias de circulação e impõe limites espaciais e regras de movimento;
- **TrafficLight**: representa os semáforos responsáveis pela regulação do tráfego nas interseções;
- **EmergencyVehicle**: extensão de `Vehicle` que introduz comportamento especial relacionado com prioridade de passagem;
- **Pedestrian** e **Crosswalk**: representam os peões e as zonas críticas de atravessamento.

2.2 Camada View

A camada **View** é responsável pela visualização da simulação e foi implementada utilizando **JavaFX**. O sistema recorre a um `Canvas` para renderizar graficamente o estado do mundo.

A classe `CanvasView` lê os dados do modelo e converte coordenadas lógicas do mundo simulado em píxeis no ecrã, assegurando uma representação coerente e fluida do movimento de veículos, peões e semáforos.

2.3 Camada Controller

A camada **Controller** atua como intermediária entre a interação do utilizador e a lógica da simulação. A classe principal é `SimulationController`, responsável por gerir o ciclo de execução da simulação através de um `AnimationTimer`.

Em cada iteração (tick), o controlador solicita ao `World` a atualização do estado das entidades e desencadeia o redesenho do cenário na interface gráfica.

---

3. Gestão de Semáforos com o Padrão State

A gestão do comportamento dos semáforos é um dos aspetos centrais do projeto. Para evitar estruturas rígidas baseadas em condicionais extensos, foi adotado o **padrão de design State**.

3.1 Interface LightState

Foi definida a interface `LightState`, que obriga todos os estados do semáforo a implementar o método `update(TrafficLight light)`. Desta forma, o objeto `TrafficLight` delega o seu comportamento ao estado atual, sem necessidade de conhecer explicitamente a cor ativa.

3.2 Estados Concretos

Os estados implementados são:

- **RedState**: bloqueia a passagem dos veículos e, após o término do temporizador interno, transita para o `GreenState`;
- **GreenState**: permite o movimento dos veículos e integra a lógica de prioridade para veículos de emergência;
- **YellowState**: atua como estado intermédio de transição e aviso.

Esta abordagem permite uma elevada extensibilidade. Por exemplo, a introdução de um novo estado (como luz intermitente) pode ser feita através da criação de uma nova classe, sem alterações ao código existente.

---

4. Gestão do Fluxo de Tráfego com o Padrão Strategy

A lógica de controlo do tráfego nos cruzamentos foi implementada através do **padrão Strategy**, permitindo alterar dinamicamente o comportamento do sistema conforme o cenário.

4.1 Estratégia de Ciclo Fixo

A estratégia `FixedCycleStrategy` representa o comportamento base do sistema, onde os semáforos alternam de estado com base em tempos predefinidos e constantes.

4.2 Estratégia de Prioridade de Emergência

A `EmergencyPriorityStrategy` introduz inteligência adicional ao sistema. O algoritmo executado segue os seguintes passos:

1. Verifica a existência de um `EmergencyVehicle` dentro de um raio pré-definido do cruzamento;
2. Caso seja detetada uma ambulância, força a transição imediata do semáforo relevante para `GreenState`;
3. Após a passagem do veículo de emergência, o sistema retoma o ciclo normal.

Esta estratégia reduz significativamente o tempo de resposta dos veículos de emergência.

---

5. Lógica de Simulação de Veículos

A classe `Vehicle` contém a lógica central do movimento. Em vez de um deslocamento simplificado a velocidade constante, cada veículo reage dinamicamente ao ambiente.

5.1 Algoritmo de Deteção (Look-ahead)

Em cada iteração, o veículo executa o método `scanEnvironment()`, que inclui:

- **Deteção de veículos à frente**: se a distância for inferior à distância de segurança, é aplicada uma força de travagem proporcional;
- **Interação com semáforos**: semáforos vermelhos ou amarelos são tratados como obstáculos, iniciando o processo de paragem gradual.

Este modelo resulta num comportamento mais realista e evita colisões.

---

6. Veículos de Emergência

A classe `EmergencyVehicle` herda de `Vehicle`, mas redefine certos comportamentos através de **polimorfismo**.

6.1 Comunicação Objeto-a-Objeto

Ao aproximar-se de um cruzamento, a ambulância atua como emissor de eventos, notificando a estratégia de tráfego. Esta comunicação suspende temporariamente o temporizador do semáforo e força a transição para verde, garantindo prioridade absoluta.

---

7. Monitorização e Recolha de Métricas

Para avaliar o desempenho do sistema, foi implementado um módulo de monitorização na classe `StatisticsManager`.

7.1 Métricas Recolhidas

- **Waiting Time**: tempo acumulado em que os veículos permanecem praticamente parados;
- **Throughput**: número de veículos que atravessam o cruzamento por minuto;
- **Emergency Response Time**: tempo total de percurso de um veículo de emergência, com e sem prioridade ativa.

7.2 Exportação de Resultados

Os dados são exportados para o ficheiro `simulation_metrics.csv` utilizando `PrintWriter`, permitindo análise posterior em ferramentas externas.

---

8. Peões e Passadeiras

Os peões são tratados como entidades independentes dentro do `World`, com regras de movimento específicas.

As passadeiras funcionam como zonas críticas de interseção. Quando um peão entra na área de uma `Crosswalk`, os veículos executam verificações constantes e param sempre que necessário, garantindo segurança.

---

9. Gestão de Projeto e Controlo de Versão

O desenvolvimento do projeto recorreu ao sistema de controlo de versões **Git**, permitindo coordenação eficaz entre os elementos da equipa, divisão clara de tarefas e histórico completo de alterações.

---

10. Conclusão

O projeto alcançou com sucesso os objetivos propostos, resultando numa simulação realista e extensível onde veículos, peões e veículos de emergência interagem de forma coerente. Os resultados obtidos confirmam que a estratégia de prioridade reduz significativamente o tempo de resposta das ambulâncias.

A arquitetura adotada e o uso de padrões de design tornam o sistema preparado para futuras extensões, como a introdução de rotundas, múltiplos cruzamentos interligados ou algoritmos de otimização avançados para gestão de tráfego urbano.