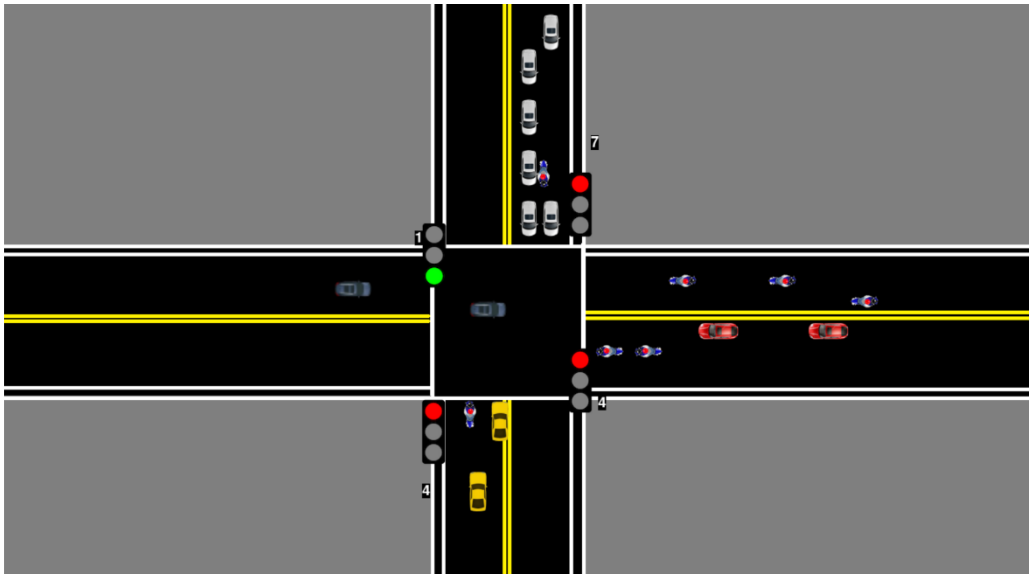
Simulação de Trânsito Inteligente

Luan J. Costa

Introdução

O projeto desenvolvido tem o objetivo de controlar o fluxo de veículos em um cruzamento com 4 semáforos, um para cada via. O controle é feito por meio de verificações de tempo de parada e quantidade de veículos parados nas vias, a via com mais veículos e com maior tempo de parada será a próxima via que irá se locomover, mantendo a dinâmica e fluidez no trânsito.



Estrutura do Código

Bibliotecas Importadas

- **pygame**: Para gerar o teste do código de forma visual.
- **random**: Para gerar veículos de forma aleatória.
- **time**: Para controle de tempos.
- **threading**: Para executar a simulação de forma assíncrona.
- **sys**: Para permitir a saída limpa do programa.

Classes

TrafficSignal

Cada instância dessa classe armazena os tempos de **verde**, **amarelo** e **vermelho** de um semáforo, além de manter o tempo que o semáforo esteve fechado (o tempo que ficou no vermelho).

Atributos:

- **red**: o tempo que o semáforo permanecerá no vermelho.
- **yellow**: o tempo que o semáforo permanecerá no amarelo.
- **green**: o tempo que o semáforo permanecerá no verde.
- **closedTime**: armazena quanto tempo o semáforo esteve no vermelho.

```python
class TrafficSignal:
    def __init__(self, red, yellow, green):
        self.red = red
        self.yellow = yellow
        self.green = green
        self.closedTime = 0
```

Vehicle

Possui a direção (ex: right, down, left, up) que o veículo irá seguir, qual o tipo (car ou bike), a posição inicial e a velocidade de deslocamento.

Atributos:

- **lane**: a faixa onde o veículo está (0, 1, ou 2).
- **vehicleClass**: a classe do veículo, por exemplo, carro ou moto.
- **speed**: a velocidade do veículo, definida com base na sua classe.
- **direction_number**: número que representa a direção em que o veículo se move (0 para direita, 1 para baixo, 2 para esquerda, 3 para cima).
- **direction**: a direção em que o veículo está indo (uma string, como 'right', 'down', 'left', 'up').
- **x** e **y**: as coordenadas iniciais do veículo no ecrã.
- **crossed**: um indicador se o veículo cruzou a linha de parada ou não.
- **stop**: a coordenada de parada do veículo (onde ele para antes de um semáforo).
- **index**: a posição do veículo em sua faixa (por exemplo, o 1º veículo na faixa ou o 2º).
- **image**: a imagem do veículo, carregada de acordo com sua classe e direção.

```python
class Vehicle(pygame.sprite.Sprite):
    def __init__(self, lane, vehicleClass, direction_number, direction):
        pygame.sprite.Sprite.__init__(self)
        self.lane = lane
        self.vehicleClass = vehicleClass
        self.speed = speeds[vehicleClass]
        self.direction_number = direction_number
        self.direction = direction
        self.x = x[direction][lane]
        self.y = y[direction][lane]
        self.crossed = 0
        vehicles[direction][lane].append(self)
        self.index = len(vehicles[direction][lane]) - 1
        path = "images/" + direction + "/" + vehicleClass + ".png"#busca a imagem do veículo
        self.image = pygame.image.load(path)
```

Métodos:

- **__init__:** inicializa o veículo com os parâmetros fornecidos (faixa, classe, direção, etc.) e calcula a posição inicial e a posição de parada. Ele também carrega a imagem do veículo e a adiciona ao grupo simulation.
- **render**: desenha a imagem do veículo na tela.
- **move**: controla o movimento do veículo, considerando se ele já cruzou a linha de parada ou se está esperando no semáforo. O movimento depende da direção do veículo e do estado do semáforo atual (verde, amarelo, ou vermelho).

Main

A classe Main gerencia a execução da simulação, incluindo a inicialização dos semáforos, geração de veículos e renderização da interface gráfica.

Loop principal

- Captura eventos do Pygame (como fechar a janela).
- Atualiza a contagem de veículos parados em cada direção.
- Renderiza os veículos na tela e atualiza a posição de cada um chamando o método move da classe Vehicle.

- Atualiza os sinais de tráfego, exibindo o estado atual (verde, amarelo, vermelho).
- Mostra na tela o número de veículos parados em cada semáforo, ao lado dos semáforos.

Funções Principais

Initialize

Tem como objetivo inicializar os semáforos com tempos padrão para cada fase (verde, amarelo e vermelho) e iniciar o ciclo de controle dos semáforos.
O que ela faz:

- Cria quatro objetos da classe TrafficSignal, um para cada direção do cruzamento (direita, baixo, esquerda e cima).
- Para o primeiro semáforo (ts1), ele define o tempo de verde, amarelo e vermelho com os valores fornecidos (defaultGreen, defaultYellow, e defaultRed).
- Para os semáforos subsequentes (ts2, ts3, ts4), o tempo de início do vermelho é calculado com base no tempo total do semáforo anterior (somatório de verde, amarelo e vermelho), o que garante que os semáforos se alternem corretamente.
- Após inicializar todos os semáforos, a função chama repeat para iniciar o ciclo contínuo de controle do tráfego.

```python
# Inicializa os semáforos com valores padrão
def initialize():
    ts1 = TrafficSignal(0, defaultYellow, defaultGreen[0])
    signals.append(ts1)
    ts2 = TrafficSignal(ts1.red+ts1.yellow+ts1.green, defaultYellow, defaultGreen[1])
    signals.append(ts2)
    ts3 = TrafficSignal(defaultRed, defaultYellow, defaultGreen[2])
    signals.append(ts3)
    ts4 = TrafficSignal(defaultRed, defaultYellow, defaultGreen[3])
    signals.append(ts4)
    repeat()
```

Repeat

É a principal responsável por controlar o ciclo dos semáforos. Ela gerencia o tempo em que cada semáforo permanece verde, depois muda para amarelo, e finalmente vermelho. Além disso, ela calcula qual semáforo deve ficar verde em seguida com base no número de veículos parados e o tempo em que os semáforos ficaram fechados.

O que ela faz:

- Loop de sinal verde: A função verifica se o semáforo atual ainda tem tempo de verde. Enquanto o semáforo estiver verde, ele chama a função updateValues para atualizar os temporizadores e espera 1 segundo.
- Mudança para amarelo: Quando o tempo do sinal verde acaba, o semáforo atual passa para o amarelo, e a função ajusta a parada de todos os veículos que estavam movendo-se na direção desse semáforo. O temporizador do amarelo é então decrementado da mesma forma que o verde, chamando updateValues a cada segundo.
- Reseta o temporizador: Quando o ciclo do semáforo (verde e amarelo) termina, o temporizador é reiniciado com os valores padrão.
- Seleciona a direção que tem o maior número de veículos parados e que está fechada há mais tempo.
- Após selecionar a direção, ela ajusta o temporizador do semáforo atual para começar a contagem de vermelho, e passa o controle para o próximo semáforo.
- Depois de escolher o próximo semáforo, a função repeat é chamada novamente, reiniciando o ciclo.

updateValues

Essa função é responsável por atualizar os temporizadores de todos os semáforos a cada segundo. Ela é chamada dentro da função repeat para garantir que os temporizadores sejam decrementados conforme o tempo passa.
O que ela faz:

- Para cada semáforo, se ele for o semáforo verde atual:
    - Se o semáforo estiver no ciclo verde (currentYellow == 0), ele decrementa o tempo de verde.
    - Se estiver no ciclo amarelo (currentYellow == 1), ele decrementa o tempo de amarelo.
- Para os outros semáforos (que estão no vermelho):
    - Decrementa o tempo do vermelho.
    - Incrementa o valor de closedTime, que mantém o registro de quanto tempo aquele semáforo ficou fechado. Esse valor é usado na função repeat para decidir qual semáforo abrir a seguir.

Essa função essencialmente controla o "relógio" de cada semáforo, garantindo que os temporizadores avancem corretamente.

generateVehicles

Tem a responsabilidade de adicionar novos veículos à simulação continuamente. Ela cria veículos de maneira aleatória em diferentes direções e faixas, simulando o tráfego no cruzamento.

- O que ela faz:
- Escolhe aleatoriamente o tipo de veículo (carro ou moto).
- Escolhe aleatoriamente uma faixa para o veículo (0, 1 ou 2).
- Define aleatoriamente a direção para o veículo (direita, baixo, esquerda, cima), usando uma distribuição que define a probabilidade de cada direção ser escolhida.
- Para cada veículo gerado, a função instancia um objeto da classe Vehicle e o adiciona ao grupo simulation (que mantém todos os veículos que estão na simulação).
- Após adicionar um novo veículo, a função faz uma pausa de 1 segundo antes de gerar o próximo veículo, criando um fluxo constante de tráfego.

isVehicleStopped

Verifica se um veículo específico está parado antes do semáforo. Isso é usado para atualizar a contagem de veículos parados em cada direção, ajudando a decidir qual semáforo abrir a seguir.

O que ela faz:

- Verifica a posição atual do veículo em relação à linha de parada do semáforo.
- Se o veículo ainda não cruzou a linha de parada (crossed == 0), e sua posição indica que ele está parado antes dessa linha, a função retorna True.
- Se o veículo já cruzou a linha ou não está mais parado, ela retorna ‘False’.

Lógica de Funcionamento

Veículos são gerados de forma aleatória em quatro direções, pela função **generateVehicle**, se movimentando em direção a um cruzamento que é gerenciado por 4 semáforos, caso o semáforo que o veículo estiver se dirigindo estiver vermelho ou amarelo, o veículo automaticamente irá parar, caso ele esteja verde, ele irá continuar seu percurso.
Cada um dos semáforos possuem um contador de veículos parados e um timer que marca quanto tempo este semáforo ficou fechado onde, na função repeat, define qual será o próximo semáforo a ficar verde baseando-se na quantidade de carros que ficaram parados e qual o semáforo ficou mais tempo fechado, o semáforo que tiver a maior quantidade de carros e o maior tempo vermelho será o próximo a ficar verde.