Stausimulation

Nagel-Schreckenberg-Modell

Angewandte Modellierung und Systemsimulation

Tamino Lembcke | 21.07.2025

Gliederung

1. Projektidee
2. Technische Umsetzung
    1. Code
    2. GUI
3. Live Demonstration
4. Quellen


Mittlerer Ring 2R

Vehicle-Klasse

```python
class Vehicle:
    def __init__(self, position, vehicle_type="car", lane=0):
        self.position = position
        self.fractional_position = 0.0  # Für flüssige Bewegung
        self.speed = 0
        self.vehicle_type = vehicle_type  # "car" oder "truck"

        # Realistische Geschwindigkeitsverteilung
        if vehicle_type == "car":
            self.max_speed = random.randint(MIN_SPEED_CAR, MAX_SPEED_CAR) * KMH_TO_CELLS
        else:
            self.max_speed = random.randint(BASE_SPEED_TRUCK - TRUCK_SPEED_VARIATION, BASE_SPEED_TRUCK + TRUCK_SPEED_VARIATION) * KMH_TO_CELLS

        self.length = 1 if vehicle_type == "car" else 2  # Zellen belegt
        self.lane = lane  # 0 = obere Spur, 1 = untere Spur
        self.lane_change_cooldown = 0  # Verhindert häufige Spurwechsel
        self.preferred_lane = 1  # Alle Fahrzeuge bevorzugen untere Spur
```

Nagel-Schreckenberg-Algorithmus

```python
def update_simulation():
    """Ein Schritt des Nagel-Schreckenberg Modells mit Spurwechsel"""
    global step_count
    step_count += 1

    # Schritt 1: Spurwechsel-Prüfung
    for vehicle in vehicles:
        other_lane = 1 - vehicle.lane

        if (can_change_lane(vehicle, other_lane) and get_lane_change_benefit(vehicle)):
            vehicle.lane = other_lane
            vehicle.lane_change_cooldown = 10  # Cooldown setzen

    # Schritt 2: Beschleunigung
    for vehicle in vehicles:
        vehicle.speed = min(vehicle.speed + 1, vehicle.max_speed)

    # Schritt 3: Bremsung (Kollisionsvermeidung)
    for vehicle in vehicles:
        distance = get_distance_to_next_vehicle(vehicle)
        if distance <= vehicle.speed:
            vehicle.speed = max(0, distance - 1)

    # Schritt 4: Zufälliges Bremsen
    for vehicle in vehicles:
        if random.random() < BRAKE_PROB:
            vehicle.speed = max(0, vehicle.speed - 1)

    # Schritt 5: Bewegung
    for vehicle in vehicles:
        vehicle.position = (vehicle.position + vehicle.speed) % NUM_CELLS
```

Nagel-Schreckenberg-Algorithmus

```python
def update_simulation():
    """Ein Schritt des Nagel-Schreckenberg Modells mit Spurwechsel"""
    global step_count
    step_count += 1

    # Schritt 1: Spurwechsel-Prüfung
    for vehicle in vehicles:
        other_lane = 1 - vehicle.lane

        if (can_change_lane(vehicle, other_lane) and get_lane_change_benefit(vehicle)):
            vehicle.lane = other_lane
            vehicle.lane_change_cooldown = 10  # Cooldown setzen

    # Schritt 2: Beschleunigung
    for vehicle in vehicles:
        vehicle.speed = min(vehicle.speed + 1, vehicle.max_speed)

    # Schritt 3: Bremsung (Kollisionsvermeidung)
    for vehicle in vehicles:
        distance = get_distance_to_next_vehicle(vehicle)
        if distance <= vehicle.speed:
            vehicle.speed = max(0, distance - 1)

    # Schritt 4: Zufälliges Bremsen
    for vehicle in vehicles:
        if random.random() < BRAKE_PROB:
            vehicle.speed = max(0, vehicle.speed - 1)

    # Schritt 5: Bewegung
    for vehicle in vehicles:
        vehicle.position = (vehicle.position + vehicle.speed) % NUM_CELLS
```

```python
def get_lane_change_benefit(vehicle):
    """Bestimmt, ob ein Spurwechsel vorteilhaft wäre"""
    current_distance = get_distance_to_next_vehicle(vehicle)
    other_lane = 1 - vehicle.lane
    other_distance = get_distance_to_next_vehicle(vehicle, other_lane)

    # Prüfung ob Fahrzeug erheblich ausgebremst wird
    is_significantly_slowed = vehicle.speed < (vehicle.max_speed * 0.7)

    if vehicle.lane == 0 and other_lane == 1:  # Rückkehr zur bevorzugten Spur
        return other_distance > current_distance + 2
    elif vehicle.lane == 1 and other_lane == 0:  # Überholmanöver
        return (is_significantly_slowed and other_distance > current_distance + 8)

    return False
```

Nagel-Schreckenberg-Algorithmus

```python
def update_simulation():
    """Ein Schritt des Nagel-Schreckenberg Modells mit Spurwechsel"""
    global step_count
    step_count += 1

    # Schritt 1: Spurwechsel-Prüfung
    for vehicle in vehicles:
        other_lane = 1 - vehicle.lane

        if (can_change_lane(vehicle, other_lane) and get_lane_change_benefit(vehicle)):
            vehicle.lane = other_lane
            vehicle.lane_change_cooldown = 10  # Cooldown setzen

    # Schritt 2: Beschleunigung
    for vehicle in vehicles:
        vehicle.speed = min(vehicle.speed + 1, vehicle.max_speed)

    # Schritt 3: Bremsung (Kollisionsvermeidung)
    for vehicle in vehicles:
        distance = get_distance_to_next_vehicle(vehicle)
        if distance <= vehicle.speed:
            vehicle.speed = max(0, distance - 1)

    # Schritt 4: Zufälliges Bremsen
    for vehicle in vehicles:
        if random.random() < BRAKE_PROB:
            vehicle.speed = max(0, vehicle.speed - 1)

    # Schritt 5: Bewegung
    for vehicle in vehicles:
        vehicle.position = (vehicle.position + vehicle.speed) % NUM_CELLS
```

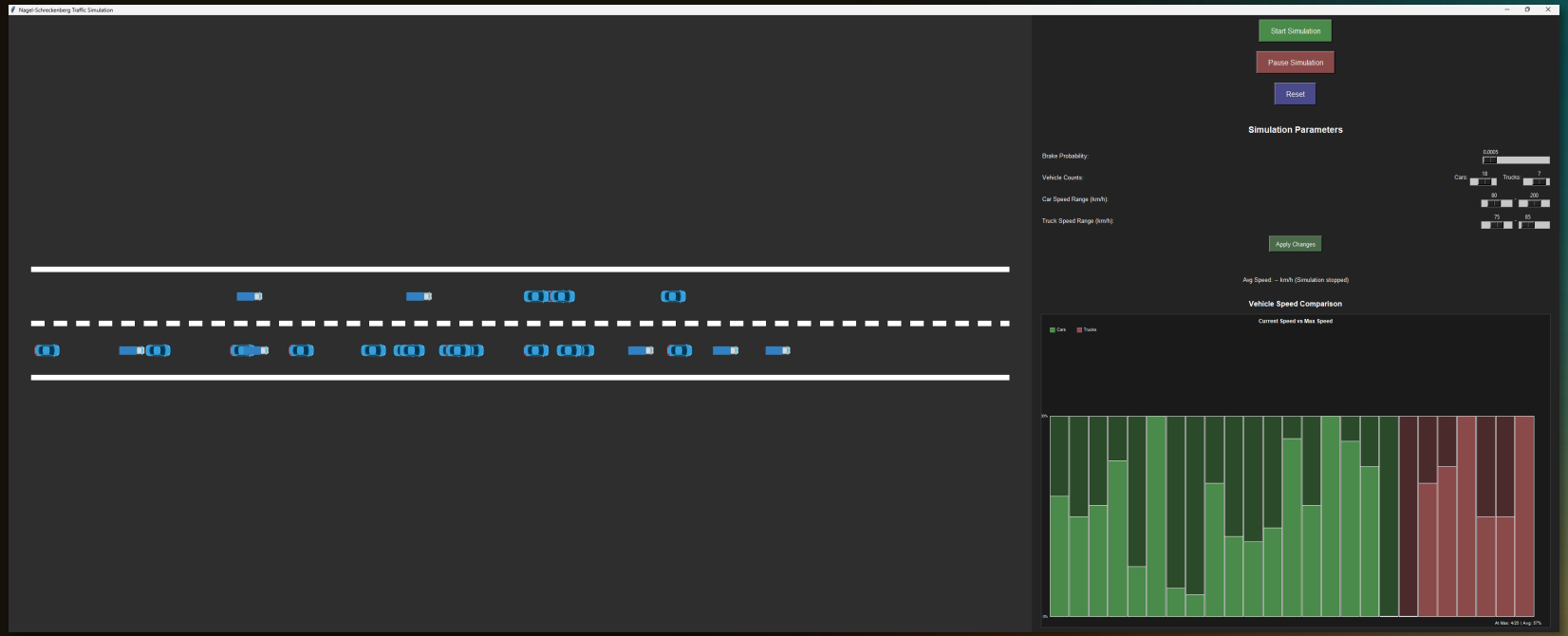
Nagel-Schreckenberg-Algorithmus

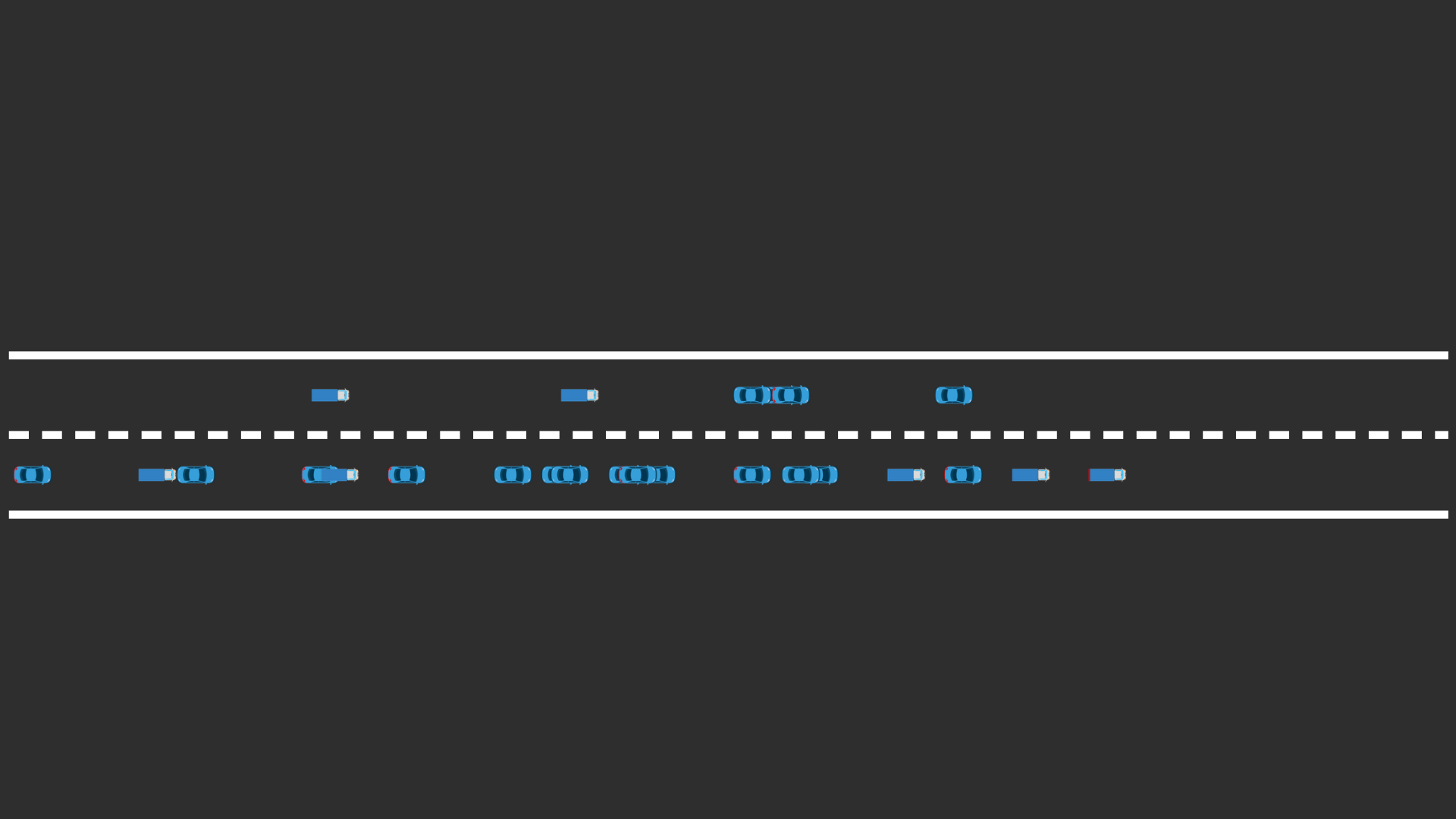
```python
def update_simulation():
    """Ein Schritt des Nagel-Schreckenberg Modells mit Spurwechsel"""
    global step_count
    step_count += 1

    # Schritt 1: Spurwechsel-Prüfung
    for vehicle in vehicles:
        other_lane = 1 - vehicle.lane

        if (can_change_lane(vehicle, other_lane) and get_lane_change_benefit(vehicle)):
            vehicle.lane = other_lane
            vehicle.lane_change_cooldown = 10  # Cooldown setzen

    # Schritt 2: Beschleunigung
    for vehicle in vehicles:
        vehicle.speed = min(vehicle.speed + 1, vehicle.max_speed)

    # Schritt 3: Bremsung (Kollisionsvermeidung)
    for vehicle in vehicles:
        distance = get_distance_to_next_vehicle(vehicle)
        if distance <= vehicle.speed:
            vehicle.speed = max(0, distance - 1)

    # Schritt 4: Zufälliges Bremsen
    for vehicle in vehicles:
        if random.random() < BRAKE_PROB:
            vehicle.speed = max(0, vehicle.speed - 1)

    # Schritt 5: Bewegung
    for vehicle in vehicles:
        vehicle.position = (vehicle.position + vehicle.speed) % NUM_CELLS
```

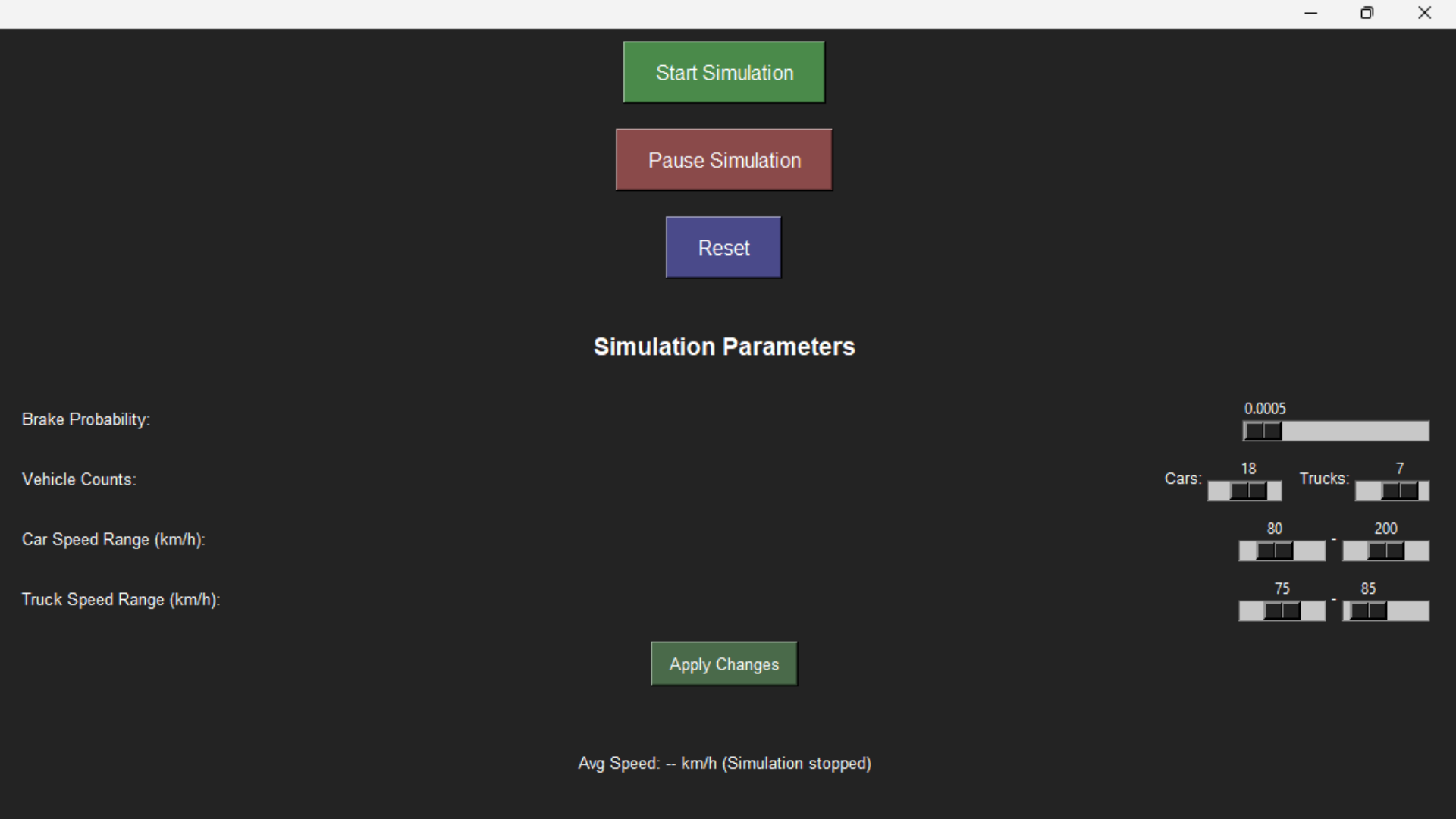
Nagel-Schreckenberg-Algorithmus

```python
def update_simulation():
    """Ein Schritt des Nagel-Schreckenberg Modells mit Spurwechsel"""
    global step_count
    step_count += 1

    # Schritt 1: Spurwechsel-Prüfung
    for vehicle in vehicles:
        other_lane = 1 - vehicle.lane

        if (can_change_lane(vehicle, other_lane) and get_lane_change_benefit(vehicle)):
            vehicle.lane = other_lane
            vehicle.lane_change_cooldown = 10  # Cooldown setzen

    # Schritt 2: Beschleunigung
    for vehicle in vehicles:
        vehicle.speed = min(vehicle.speed + 1, vehicle.max_speed)

    # Schritt 3: Bremsung (Kollisionsvermeidung)
    for vehicle in vehicles:
        distance = get_distance_to_next_vehicle(vehicle)
        if distance <= vehicle.speed:
            vehicle.speed = max(0, distance - 1)

    # Schritt 4: Zufälliges Bremsen
    for vehicle in vehicles:
        if random.random() < BRAKE_PROB:
            vehicle.speed = max(0, vehicle.speed - 1)

    # Schritt 5: Bewegung
    for vehicle in vehicles:
        vehicle.position = (vehicle.position + vehicle.speed) % NUM_CELLS
```

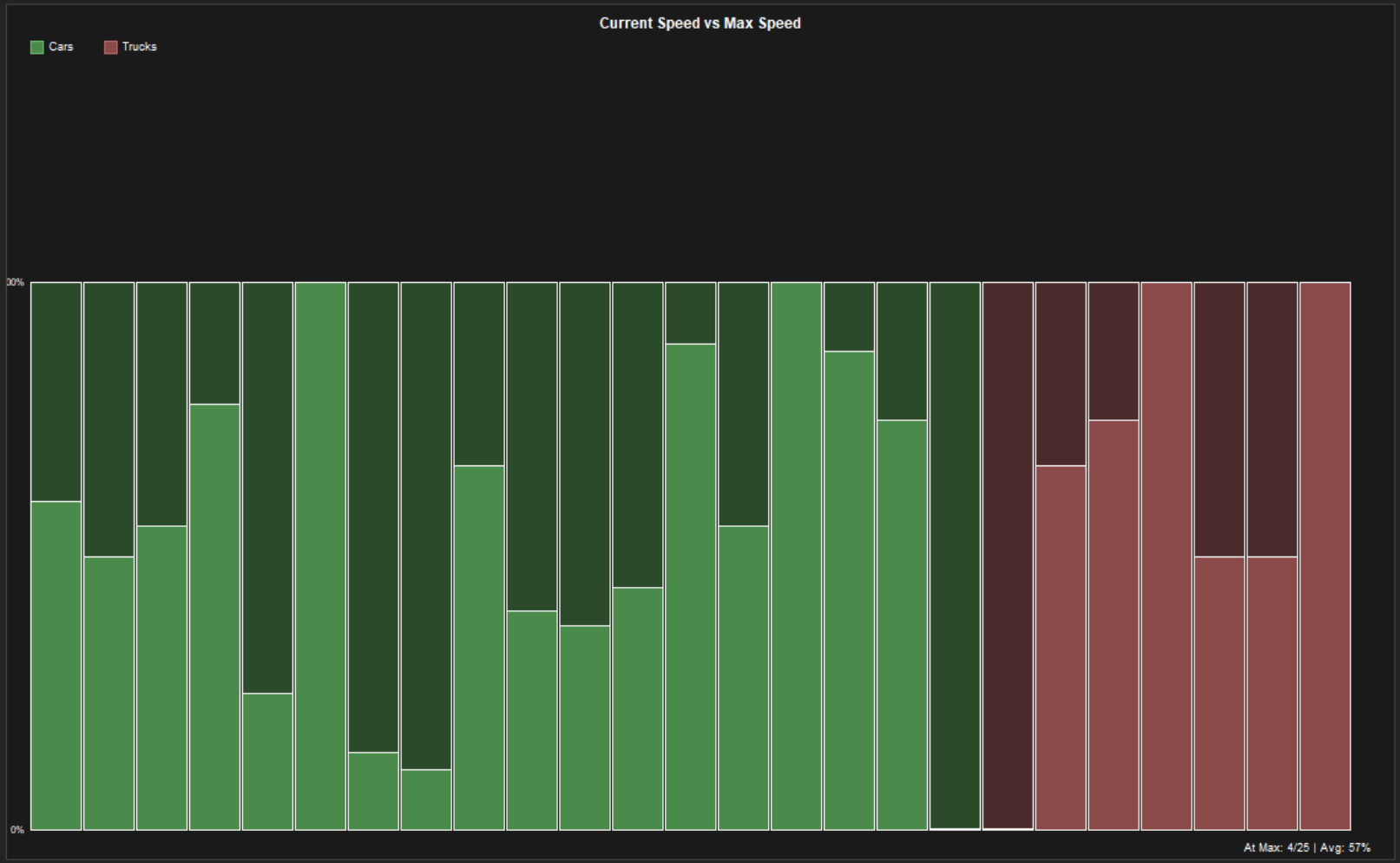
GUI


Nagel-Schreckenberg Traffic Simulation
Start Simulation
Pause Simulation
Reset
Simulation Parameters
Brake Probability:
0.0005
Vehicle Counts:
Cars:
18
Trucks:
7
Car Speed Range (km/h):
80
200
Truck Speed Range (km/h):
75
85
Apply Changes
Avg Speed: – km/h (Simulation stopped)
Vehicle Speed Comparison
Current Speed vs Max Speed
Cars
Trucks
At Max: 4/25 | Avg: 57%




Start Simulation
Pause Simulation
Reset
Simulation Parameters
Brake Probability:
0.0005
Vehicle Counts:
Cars:
18
Trucks:
7
Car Speed Range (km/h):
80
-
200
Truck Speed Range (km/h):
75
-
85
Apply Changes
Avg Speed: -- km/h (Simulation stopped)

Vehicle Speed Comparison



Live Demonstration

Inhaltliche Quellen

- https://de.wikipedia.org/wiki/Nagel-Schreckenberg-Modell (letzter Zugriff 06.07.2025)

- Nutzung von KI:
  - ChatGPT (gpt-4o) zur inhaltlichen Zusammenfassung für die Präsentation
  - GitHub Copilot (Agent Mode mit Claude Sonnet 4.0 Preview) zur Unterstützung bei der Implementierung und inhaltlicher Zusammenfassung für die Dokumentation

Sonstige Quellen

- https://slidesgo.com/ (letzter Zugriff 06.07.2025)
- https://www.qrcode-monkey.com/de/ (letzter Zugriff 06.07.2025)
- https://images.tagesschau.de/image/d880fb19-6f13-49b7-ad99-2032e094f0f9/AAABlfRzw30/AAABkZLiamM/16x9-1920/stau-muenchen-109.jpg (letzter Zugriff 06.07.2025)
- https://www.esslinger-zeitung.de/media.media.35f2bbe3-ad9e-4995-b3e8-0fb87f64d056.original1024.jpg (letzter Zugriff 06.07.2025)
- https://cdn.prod.www.spiegel.de/images/c85f4073-0001-0004-0000-000000666442_w960_r1.778_fpx41.4_fpy49.97.jpg (letzter Zugriff 06.07.2025)

Stausimulation

Nagel-Schneckenberg-Modell

Tamino Lembcke | 21.07.2025

Projekt auf GitHub