Verkeersimulatie Kruispunt

Applied Artificial Intelligence

Analytical Computing

Berry Hijwegen

Snelheid berekenen op basis van tijd en afstand

Voor de relatie van tijd, afstand en snelheid geldt de volgende formule

$$\Delta s = v * \Delta t$$

Om te schrijven naar de volgende formules:

$$v = \frac{\Delta s}{\Delta t}$$

$$t = \frac{\Delta s}{v}$$

Waarvoor geldt dat:

Δs = Verandering van afstand

Δt = Verandering van tijd

v = snelheid

Dit wordt in de code toegepast in het volgende snippet:

```
    timestamp           = float(times[i])
    posCar1             = float(positionsCar1[i])
    posCar2             = float(positionsCar2[i])

    prevTimestamp       = float(times[i - 1]) if i != 0 else timestamp
    prevPosCar1         = float(positionsCar1[i- 1]) if i != 0 else posCar1
    prevPosCar2         = float(positionsCar2[i- 1]) if i != 0 else posCar2

    elapsedTime         = round(timestamp - prevTimestamp,1)
    traveledDistance1   = posCar1 - prevPosCar1
    traveledDistance2   = posCar2 - prevPosCar2

    currSpeedCar1KM     = traveledDistance1 / elapsedTime if elapsedTime != 0
else 0
    currSpeedCar2KM     = traveledDistance2 / elapsedTime if elapsedTime != 0
else 0
```

Dit stuk code staat in een loop die alle momentopnames per 0.1 seconde bevat. Hierbij wordt steeds de verlopen tijd sinds de vorige moment opname berekent (Δt), en de afgelegde afstand sinds de vorige momentopname Δs, Vervolgens wordt de afgelegde afstand gedeeld door de verlopen tijd ($\frac{\Delta s}{\Delta t}$). Dit levert de huidige snelheid op.

```
    speedDataCar1.append(currSpeedCar1KM * 36)
    speedDataCar2.append(currSpeedCar2KM * 36)
    timestamps.append(timestamp)

    maxSpeedCar1 = currSpeedCar1KM if currSpeedCar1KM > maxSpeedCar1 or i == 0
else maxSpeedCar1
    minSpeedCar1 = currSpeedCar1KM if currSpeedCar1KM < minSpeedCar1 or i == 0
else minSpeedCar1

    maxSpeedCar2 = currSpeedCar2KM if currSpeedCar2KM > maxSpeedCar2 or i == 0
else maxSpeedCar2
    minSpeedCar2 = currSpeedCar2KM if currSpeedCar2KM < minSpeedCar2 or i == 0
else minSpeedCar2
```

Vervolgens wordt in deze zelfde loop de huidige snelheid toegevoegd aan een lijst met alle snelheden over tijd in km/h per auto, en de timestamps aan een eigen lijst. Daarnaast wordt er gekeken of de huidige snelheid hoger is dan de vorige maximale snelheid, zoals gevraagd in de opdracht.

```
plt.plot(timestamps, speedDataCar1)
plt.plot(timestamps, speedDataCar2)
plt.xlabel('Time in seconds')
plt.ylabel('Speed in km/h')
plt.show()
```

Door middel van matplotlib wordt de data in een grafiek weergeven.

Afstand berekenen op basis van tijd en snelheid

Eerdergenoemde formules zijn dus ook te gebruiken om de afstand te berekenen op basis van een bekende tijd en bekende snelheid. Hiervoor geldt de formule:

$\Delta s = v * \Delta t$

```
currentSpeed[i] = float(row[i+1])
                index = 1 if directions[i + 1] == 'v' else 0
                carPositions[i + 1][index] += currentSpeed[i] * 0.1

                plt.scatter(carPositions[i + 1][0],carPositions[i + 1][1],
color=carColor[i], s=15, zorder=10)
                plt.pause(0.1)
```

In bovenstaande code wordt steeds de huidige snelheid bijgesteld, en daarbij de huidige positie van de auto, in een loop die alle rijen langs gaat uit de CSV. Elke auto heeft een index in 'directions', of 'v', of 'h'. Deze bepaalt of de positie in de x-as of de y-as bijgewerkt moet worden. Om de nieuwe positie van de auto te berekenen wordt bovenstaande formule gebruikt, omdat Δt constant is in de CSV wordt deze standaard op 0.1 gezet. De huidige snelheid van de auto wordt vermenigvuldigt met de huidige snelheid, deze wordt toegevoegd op de juiste index van de positie van de auto, die eerder is bepaald aan de hand van de 'v' of de 'h'. Vervolgens wordt dit punt direct geplot.

```
for carPos1 in carPositions:

      # Get length of car in right direction
      distanceCoordinates1 = [2,1.5] if directions[carPos1]  == 'h'
                                                              else [1.5,2]
      for carPos2 in carPositions:

            if(abs(carPositions[carPos1][0]-carPositions[carPos2][0]) <
                  distanceCoordinates1[0]
                  and abs(carPositions[carPos1][1]-carPositions[carPos2][1])
                  < distanceCoordinates1[0]
                  and carPos1 != carPos2):
                 print(f"{timestamp}s: Car {carPos1} bumped into car
{carPos2}!\n{carPositions}\n")
```

In bovenstaande code wordt gekeken of er een auto tegen een andere auto aangereden is op dat moment. Hier wordt ook weer onderscheidt gemaakt tussen horizontaal en verticaal georiënteerde auto's. Voor elke auto wordt er gekeken of de afstand tussen die auto en een van de andere auto's kleiner is dan de grootte van de auto. Vervolgens wordt weergeven wanneer dit gebeurt is en welke auto's betrokken zijn.

GitHub

https://github.com/berryhijwegen/AC_HU/tree/master/calculus

Bronnen

https://nl.wikipedia.org/wiki/Snelheid#Snelheid_in_formulevorm