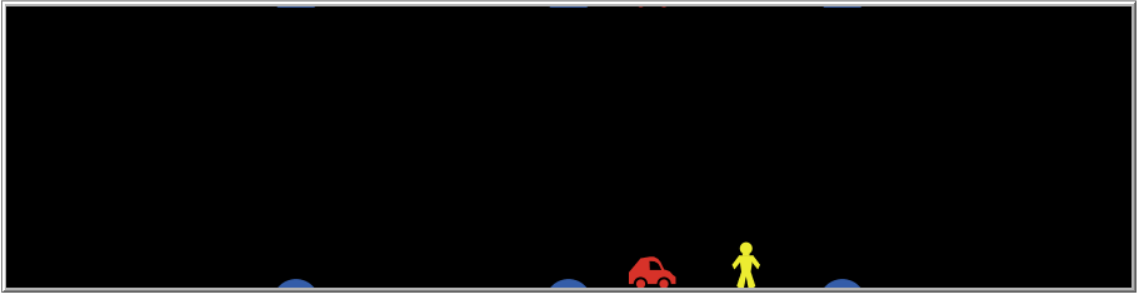
**Ongelukken voorkomen met drempels**



Door: Tom Blokland, Carlo Keizer, Stijn Femijn

Datum: 17/12/2019

**Inhoudsopgave**

[**Introductie onderwerp** 3](#_Toc27685469)

[**De Onderzoeksvraag** 4](#_Toc27685470)

[**Plan van aanpak en toolkeuze** 5](#_Toc27685471)

[**Plan** 5](#_Toc27685472)

[**FSA-Model** 6](#_Toc27685473)

[**Tool keuze** 8](#_Toc27685474)

[**Uitleg/design van het experiment** 9](#_Toc27685475)

[**Resultaten van het experiment** 10](#_Toc27685476)

[**Conclusie** 12](#_Toc27685477)

# **Introductie onderwerp**

Voor het vak Simulation Tooling moeten wij een verkeerssimulatie bouwen. Met als doel het uitvoeren van een experiment op de simulatie. Hierbij moeten wij ook een hypothese opstellen waar we aan de hand van het experiment van de simulatie gaan kijken of de hypothese klopt. Het onderwerp dat wij gekozen hebben voor deze opdracht is een verkeerssimulatie met 3 verkeersdrempels en een persoon die de weg oversteekt. Wij willen met behulp van deze simulatie gaan kijken in hoeveel procent van de gevallen de overstekende persoon wordt aangereden, en wat het effect zal zijn op het percentage ongelukken als er verkeersdrempels geplaatst zullen worden. Wat ook een rol kan spelen in het percentage ongelukken is het soort verkeersdrempel. Wij willen dus ook gaan onderzoeken welke verkeersdrempels het beste geplaatst zouden kunnen worden in een gelijksoortige situatie.

# **De Onderzoeksvraag**

We willen aan de hand van een simulatie erachter komen of het nuttig is om drempels op een weg te plaatsen die auto’s tot een bepaalde snelheid doet afremmen. Dit wordt gemeten aan de hand van het aantal botsingen dat gebeurt op de weg. Als het plaatsen van een drempel op de weg zorgt voor minder botsingen is het de kosten waard. Hiervoor moet de simulatie in staat zijn de remafstand van een auto op verschillende snelheden te simuleren. Verder moeten we een botsing tussen de auto en een object kunnen waarnemen. In de scope van dit onderzoek is het alleen van toepassing of er een botsing plaatsvindt of niet, eventuele grote van de impact is niet relevant voor onze resultaten en onze conclusie.

Om wegwerkzaamheden en de kosten daarvan te verkleinen is het belangrijk om eerst aan te kunnen tonen of het plaatsen van een drempel positief invloed heeft op de verkeerssituatie.

**Hypothese**

H0: De gemiddelde snelheid wordt niet vertraagd en wanneer iemand plotseling de weg oversteekt is de kans hoog dat er een ongeluk gebeurd.

HA: De gemiddelde snelheid wordt wel vertraagd en wanneer iemand plotseling de weg oversteekt is de kans 0 dat er een ongeluk gebeurt.

# **Plan van aanpak en toolkeuze**

## **Plan**

* Auto
  + Eigenschappen
    - Remkracht
    - acceleratiekracht
    - Rijsnelheid
    - Locatie
    - Zicht
    - Random spawn persoon
  + Functies
    - Interact-with-person – De auto stopt als hij een persoon ziet
    - Interact-with-speedbump – De auto remt af als hij een drempel ziet
  + Verdeling snelheid
* Omgeving
  + Agents
    - Auto
  + Objecten
    - Verkeersdrempel
    - Overstekende persoon
* Drempel
  + Eigenschappen
    - Invloed op snelheid
    - Positie
* Dataopslag
  + Aantal ongelukken
  + Aantal overlevende
  + Aantal runs
* GUI
  + Aanpassen van variabelen
  + Realtime grafiek van de snelheid van de auto
* Reset
  + Reset de omgeving met nieuwe variabelen

## **Uitleg-plan**

In deze uitleg wordt een verdere toelichting gegeven op het gemaakte plan hierboven. In onze simulatie hebben wij een agent (de auto), verkeersdrempels en een overstekende persoon.

**Auto**

De auto heeft een aantal eigen de eigenschappen die hierboven beschreven zijn. De nieuwe locatie van de auto is een X-coördinaat die elke stap met een differentiaalvergelijking wordt berekend met behulp van de huidige locatie, rijsnelheid, rem- en acceleratiekracht. Het zicht van de auto bepaald vanaf hoeveel meter voor de drempel de auto afgeremd gaat worden. De random spawn person bepaald in welke stap de persoon de straat oversteekt. De snelheid van de auto is ook normaal verdeeld zodat je niet iedere run van de simulatie precies dezelfde snelheid krijgt.

**Drempel**

De verkeersdrempel heeft de eigenschappen zoals hierboven beschreven zijn, de belangrijkste eigenschap is de invloed op de snelheid. De dit is de snelheid tot waar de auto maximaal afgeremd kan worden door de drempel. Dit betekent dus dat de auto de drempel ook sneller mag passeren, als de auto bijvoorbeeld een slecht zicht heeft. De locatie van de drempel is ook zoals bij de auto een X-coördinaat.

## **FSA-Model**

Om tot een tool keuze te komen maken we gebruik van het FSA-model. Voor elke module in de simulatie kijken we hoe goed de verschillende tools passen. De punten gaan van 1 t/m 5 met 1 het slechts en 5 het beste.

**Unity:** Het is zeker goed passend om een agent te maken in Unity. Wel is het in dit geval minder geschikt omdat het gebruiken van Unity deze simulatie onnodig complexer maken.

**Mesa:** In Mesa is het relatief tot andere toolings iets moeilijker om Agents aan te maken, maar de interactie kan heel precies en redelijk simpel worden geimplementeerd. Zo kunnen de variabele massa, snelheid etc makkelijk worden toegewijd aan een auto.

**Netlogo:** Netlogo is een tool waarbij je het makkelijk is om Agents aan te maken en gedrag te geven. Omdat dit in een nieuwe taal is maakt het wel weer iets moeilijker

|  |  |  |  |
| --- | --- | --- | --- |
| **Auto** | Unity | Mesa | Netlogo |
| Feasibility | 2 | 3 | 5 |
| Suitibility | 4 | 2 | 4 |

**Unity:** Unity is enorm geschikt om een omgeving voor een simulatie te maken. Het wordt mooi grafisch weergegeven waardoor het duidelijk is wat er gebeurd en er dus makkelijker fouten in de simulatie ontdekt kunnen worden. Ook het maken van een omgeving in Unity is goed te doen.

**Mesa:** Mesa is heel beperkt als het gaat om graphics. Ook is het niet praktisch en niet gewenst om beweging te simuleren met Mesa. Dit is mogelijk met een grid, maar dat is minder geschikt dan bij de andere toolings.

**Netlogo:** Netlogo heeft een standaard omgeving, waar in alles wordt weergeven. Waardoor je dus makkelijk kan zien of er iets fout gaat.

|  |  |  |  |
| --- | --- | --- | --- |
| **Omgeving** | Unity | Mesa | Netlogo |
| Feasibility | 4 | 4 | 4 |
| Suitibility | 5 | 3 | 4 |

**Unity:** Zodra de omgeving in Unity is gemaakt is het niet ingewikkeld om daar nog meerdere elementen aan toe te voegen. Maar het toevoegen van een drempel kan in Unity veel problemen met zich meebrengen, vooral op het vlak van physics. Waardoor de kans op bugs in de simulatie flink wordt verhoogt.

**Mesa:** Nieuwe Agents of objecten aanmaken die met de omgeving of andere Agents interacteren is te doen. Maar het is vooral bedoeld om een simulatie te maken van veel dezelfde soort Agents.

**Netlogo:** je kan in netlogo heel makkelijk objecten aanmaken en toevoegen aan de omgeving.

|  |  |  |  |
| --- | --- | --- | --- |
| **Drempel** | Unity | Mesa | Netlogo |
| Feasibility | 2 | 3 | 4 |
| Suitibility | 4 | 3 | 5 |

**Unity:** Het moet vast mogelijk zijn om de data die de simulatie genereerd op te slaan met behulp van Unity. Maar hoe dat werkt en waar we die data met gemak kunnen terug kijken en raatplegen is niet duidelijk.

**Mesa:** Mesa is geschikt om data op te slaan. Er kan makkelijk met variabelen worden gewerkt, de tooling is gericht op de resultaten en de resultaten kunnen makkelijk worden gevisualiseerd. Van de variabelen van het model kan makkelijk een dataframe worden gemaakt dat snel en goed in een grafiek kan worden getoond.

**Netlogo:** In Netlogo is het mogelijk om globale variabelen aan te maken waar ik het aantal ongelukken bij gehouden kan worden.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataopslag** | Unity | Mesa | Netlogo |
| Feasibility | 2 | 3 | 3 |
| Suitibility | 2 | 3 | 4 |

**Unity:** Er is veel informatie beschikbaar in Unity, maar om dat te verwerken in een GUI is lastig en zorgt voor veel extra ontwikkeling tijd voor dit project dat voorkomen kan worden door een ander tool te gebruiken.

**Mesa:** Mesa heeft geen goede ingebouwde GUI. Om dit toe te voegen moet er gewerkt worden met een ander package. Mesa is vooral gericht om snel een resultaat te simuleren.

**Netlogo:** In Netlogo kan je heel gemakkelijk gui maken door middel van knoppen, spider etc. toe te voegen, en het is ook heel simpel om deze samen te laten werken met de code. Omdat dit achter de schermen al gekoppeld is.

|  |  |  |  |
| --- | --- | --- | --- |
| **GUI** | Unity | Mesa | Netlogo |
| Feasibility | 1 | 2 | 4 |
| Suitibility | 4 | 1 | 5 |

**Unity:** Het laten runnen van één simulatie is simpel te doen in Unity. Om meerdere tot wel duizenden simulaties te runnen wordt het wel een stuk lastiger. Ook kan het veel meer compute kracht kosten om alle simulaties te runnen wat Unity weer minder geschikt is.

**Mesa:** Het resetten is niet makkelijk met Mesa. Ook is dit niet makkelijk met een GUI.

**Netlogo:** Je kan in netlogo een knop aan maken, waarin je definieert, wat er moet gebeuren na een reset. Dit is ook heel gemakkelijk te maken.

|  |  |  |  |
| --- | --- | --- | --- |
| **Reset** | Unity | Mesa | Netlogo |
| Feasibility | 1 | 3 | 3 |
| Suitibility | 2 | 2 | 4 |

**Unity:** Zoals gezegd is het met Unity goed te doen om een omgeving op te zetten. Wel is het lastig om erachter te komen hoe alles werkt en ontstaan er snel bugs die niet op het eerste gezicht een duidelijke oorzaak hebben wat Unity weer minder geschikt maakt.

**Mesa:** Er kan makkelijk een for loop worden gemaakt in python en het model van Mesa kan makkelijk worden aangeroepen. De code is straight forward, maar kan niet goed getest worden in visualisaties.

**Netlogo:** Het is mogelijk om in netlogo een loop te maken, waardoor je steeds dezelfde simulatie uitvoerd, die gezien wordt als een vaste setup.

|  |  |  |  |
| --- | --- | --- | --- |
| **Vaste setup** | Unity | Mesa | Netlogo |
| Feasibility | 2 | 3 | 4 |
| Suitibility | 3 | 2 | 4 |

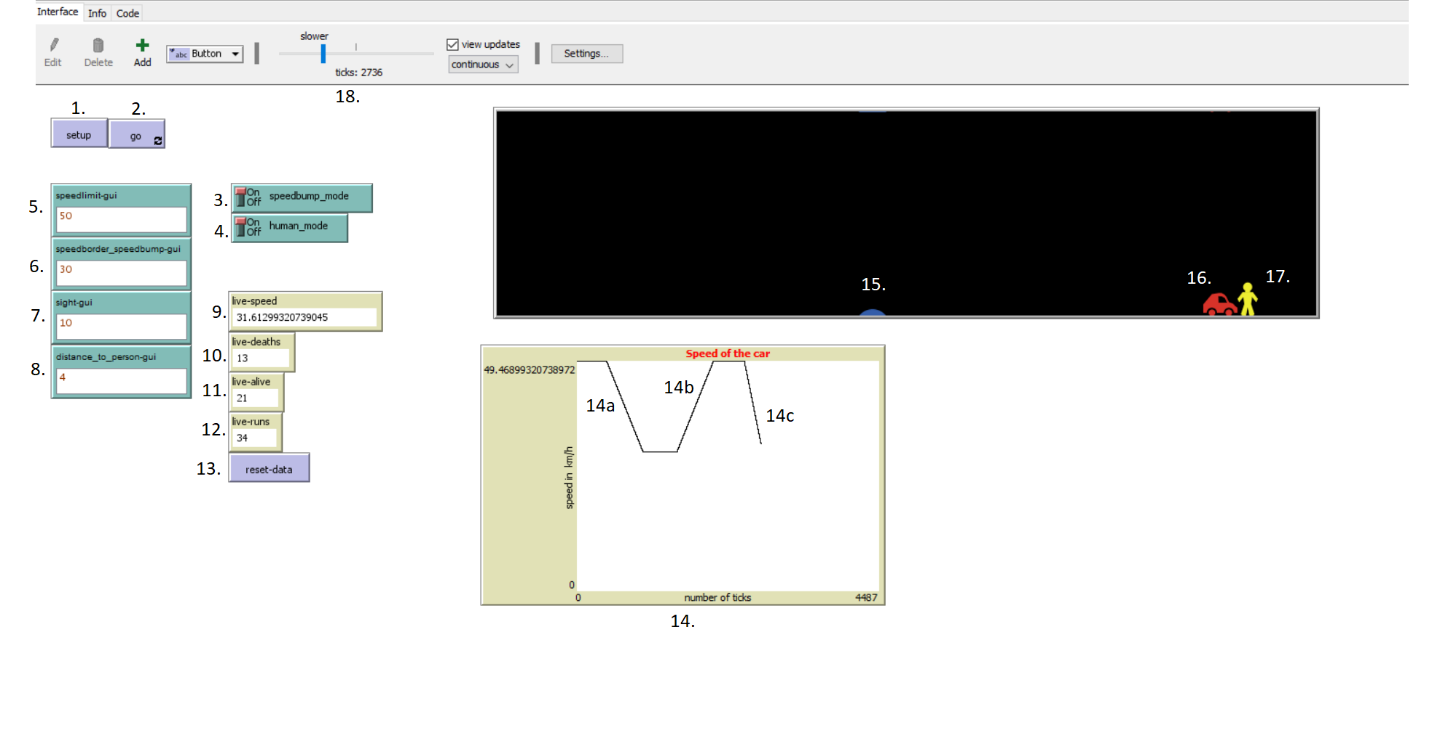
## **Tool keuze**

Na het in kaart brengen van de tools met behulp van een FSA-model. Hebben we aan de hand van onze ervaringen met de tools en het model besloten NetLogo te gebruiken. NetLogo is gericht op ABM en kan een simulatie goed en makkelijk simuleren. We hebben allemaal 1 tool gebruikt voor tijdens de vorige opdracht en die voor deze toolkeuze onderling besproken. Tijdens ons overleg hebben we besproken wat de voor- en nadelen van de verschillende tools zijn, waarbij Unity te veel gericht is op fysische eigenschappen en Mesa weer heel slecht is in het visualiseren. Uiteindelijk zijn we uitgekomen op NetLogo. Netlogo was goed gelukt, en we wilden ook leren omgaan met visuele simulaties: dit maakt het testen makkelijker en het maakt onze simulatie concreter.

# **Uitleg/design van het experiment**

Zoals al eerder verteld gaan wij een experiment doen waarin wij willen meten wat het effect is van een verkeersdrempel in een straat op het aantal ongelukken. Om dit experiment te kunnen uitvoeren hebben wij een simulatie gebouwd waarin de simulatie uitgevoerd wordt. Deze simulatie ziet er als volgt uit.

*Figuur 1*



Aan de rechterkant zie je een zwartvlak die de weg voorstelt. Aan de linkerkant kan je de parameters van de simulatie wijzigen. Hieronder staat een overzicht van alle parameters/visualisaties:

**setup simulatie**

1. Dit herhaalt de simulatie
2. Dit start de simulatie
3. Of er wel of geen verkeersdrempel is
4. Of er wel of geen persoon de weg oversteekt

**input**

1. De gemiddelde snelheid van de auto. De snelheid zal rondom het gemiddelde zitten met standaarddeviatie 8 (Wij hebben onze snelheid op 50 gehouden)
2. Snelheid tot waar de verkeersdrempel de auto afremt (We kijken naar de scenario’s 10, 20, 30 en 40)
3. De afstand vanaf wanneer de auto gaat remmen (default = 8)
4. De afstand tussen auto en persoon wanneer persoon zich op de weg bevind (we kijken naar de scenario’s van 2 t/m 8)

**Resultaten**

1. De snelheid van de auto dat live wordt gemeten
2. Aantal ongelukken/doden
3. Aantal geen ongelukken/levenden
4. Aantal keren gerund/gereset
5. Reset alle resultaten en print de resultaten in de Command Center
6. Grafiek van de snelheid over de tijd
   1. Auto remt af tot de *speedborder\_speedbump* (6.) wanneer hij de drempel tegen komt
   2. Auto accelereert tot zijn beginsnelheid wanneer hij voorbij de drempel is
   3. Auto remt 2 keer zo hard wanneer hij een persoon tegen komt totdat hij stil staat

**Visualisatie**

1. Drempel
2. Auto
3. Persoon
4. Snelheid simulatie. 30% voor een zichtbare simulatie.

**Werking van de simulatie - samengevat**

De simulatie die wij gebouwd hebben werkt als volgt, zodra er op go geklikt begint er een auto te rijden, als de auto de drempel in het zicht krijgt begint de auto te af te remmen tot de snelheidsgrens van de drempel. Als de auto de drempel voorbij is begint de auto weer te accelereren tot de maximumsnelheid. Als de auto een persoon in zijn zicht krijgt dan remt de auto dubbel zo hard tot dat de auto stilstaat.

Voor de volgende settings hebben we de simulatie laten runnen:

* Soort drempel: 10, 20, 30, 40 geen drempel
  + Dit geeft de steilheid van een drempel aan. Bij drempel 10 wordt de auto afgeremd tot snelheid 10, bij drempel 20 tot 20, etc.
* Afstand tussen persoon en auto:2,3,4,5,6,7,8

Voor elke setting wordt de simulatie 500 keer gesimuleerd. Er wordt per setting gekeken naar hoeveel procent van de keren een ongeluk voor komt.

<*Hoe hebben we het getest>*

**Hoe geeft deze simulatie antwoord op het experiment**

Met behulp van deze simulatie kunnen wij het experiment uitvoeren om erachter te komen wat het effect is van een verkeersdrempel. Want tijdens het uitvoeren van de simulatie wordt er bijgehouden hoeveel ongelukken er voorkomen, en hoe vaak de auto op tijd heeft kunnen stoppen. Deze data kunnen wij verwerken in een grafiek om vervolgens te kijken hoe groot de kans is op een ongeluk met verschillende soorten drempels. Dit kunnen we doen door per drempel en per afstand van de persoon tot de auto, te kijken wat de kans is op een ongeluk.

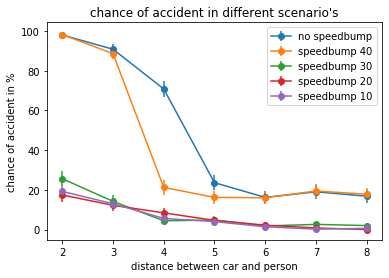
# **Resultaten van het experiment**

We hebben de simulatie laten runnen tot en met 500 trials per scenario. Vervolgens kijken we naar hoeveel procent van de trials de persoon wordt aangereden. Dit geeft uiteindelijk aan wat de gemiddelde kans op een ongeluk is.

De volgende resultaten kwamen hieruit.

*(We hebben python – matplotlib gebruikt om de grafieken te potten)*

*Figuur 2*



*Tabel 1: chance of accident in different scenario’s*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **afstand persoon en auto** | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| **soort drempel** |  |  |  |  |  |  |  |
| no speedbump | 98.0 | 90.8 | 70.8 | 23.6 | 16.2 | 19.0 | 16.8 |
| speedbump 40 | 98.2 | 88.6 | 21.2 | 16.2 | 16.0 | 19.4 | 17.8 |
| speedbump 30 | 25.6 | 14.2 | 4.4 | 4.8 | 1.8 | 2.6 | 2.0 |
| speedbump 20 | 17.4 | 12.2 | 8.4 | 4.6 | 2.2 | 0.8 | 0.0 |
| speedbump 10 | 19.2 | 13.0 | 5.6 | 4.0 | 1.4 | 0.2 | 0.6 |

Hier zien we dat het aantal ongelukken afneemt naarmate de afstand tot persoon toeneemt, en dat het aantal ongelukken afneemt bij een steilere verkeersdrempel (speedbump 10 is steiler dan 20, 30 of 40). Ook is er een groot verschil wanneer er geen drempel is. In de meest gevaarlijke momenten is er een 98% kans dat er een ongeluk plaats vindt.

**Betrouwbaarheid**

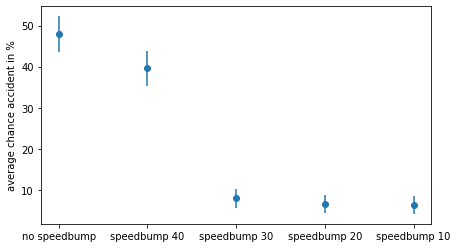
Omdat we de normal distribution voor de snelheid hebben gebruikt zal elke test net iets anders verlopen. Om te kijken hoe accuraat onze resultaten zijn hebben we een error range opgesteld op basis van de centrale limietstelling. De lower en upper bound wordt berekend met de volgende formule:

*waarbij p de kans op een ongeluk is dat uit onze simulatie komt, en z de z-waarde is bijhorend het gewenste betrouwbaarheids interval*

We willen een foutmarge van 5% hebben kiezen voor de bijhorende z-waarde = 1.96. Deze intervallen hebben we ook gesimuleerd en we kunnen met 95% zekerheid zeggen dat de percentages tussen de verticale strepen liggen. Omdat we een grote test size hebben gekozen (n=500) zien we een klein betrouwbaarheidsinterval, dat natuurlijk goed is.

Realistisch gezien heb je geen of weinig invloed op de afstand tussen de auto en de persoon. Daarom kijken we ook naar de gemiddelde percentage ongelukken per soort drempel.

*Figuur 3*



Gemiddelde kans op een ongeluk bij een snelheid van 50:

*Tabel 2*

|  |  |  |
| --- | --- | --- |
| **Soort drempel** | **Kans op ongeluk** | **Error bij 95% zekerheid** |
| Zonder verkeersdrempel | 47.9% | 4.3% |
| Verkeersdrempel 40 | 39.6% | 4.3% |
| Verkeersdrempel 30 | 7.9% | 2.4% |
| Verkeersdrempel 20 | 6.5% | 2.1% |
| Verkeersdrempel 10 | 6.3% | 2.1% |

# **Conclusie**

Uit onze resultaten zijn een aantal dingen opgevallen

* Het plaatsen van een verkeersdrempel verminderd de kans op een ongeluk met minimaal 8.3%
* Bij kleine verkeersdrempels is het effect van het verhogen van de verkeersdrempel erg hoog. Maar bij grote verkeersdrempels is het effect van het nog slomer laten rijden van auto’s een stuk minder. Het verschil tussen verkeersdrempel 40 en 20 is 33.1%, terwijl het verschil tussen verkeersdrempel 10 en 20 maar 0.2% is.
* Het effect van de hoeveelheid tijd die de een bestuurder heeft om te remmen wordt ook steeds groter naarmate de drempel steiler wordt. Zo zien we in figuur 1 dat bij de lijnen “*no speedbump”* en “*speedbump 40”* de kans op een ongeluk veel sneller daalt bij een grotere afstand tussen persoon en auto. Bij “*speedbump 30, speedbump 20*” en *“speedbump 10”* zien we een veel lager effect van de afstand tussen persoon en auto.
* In tabel 1 zien we dat in 1 scenario het aantal ongelukken 0 is. Dit is in de situatie bij drempels die de auto vertraagd naar 20, waarbij de afstand tussen de auto en persoon 8 is.
* De betrouwbaarheid van de resultaten is hoog. We kunnen met 95% zekerheid het percentage ongelukken voorspellen met maximaal 4.3% speling.

Op basis van onze bevindingen adviseren wij in eerste geval 3 drempels te plaatsen. Hiermee vermindert de kans op een ongeluk met minimaal 8.3% en potentieel met 41.1%!.

We adviseren ook om drempels te plaatsen die auto’s vertragen naar een snelheid van 30. We zien nauwelijks effect bij het plaatsen van drempels die de auto vertraagt naar 40. Gemiddeld verlaagd dat de kans op een ongeluk met maar 8.3%, terwijl bij “*speedbump 30”* de kans verlaagd wordt met 33.1%. We zien ook een duidelijke scheiding tussen 2 groepen. Geen drempels plaatsen of een speedbump 40 plaatsen, en een speedbump 30, 20 of 10 plaatsen.

Onze hypthese is: *De gemiddelde snelheid wordt wel vertraagd en wanneer iemand plotseling de weg oversteekt is de kans 0 dat er een ongeluk gebeurt.*

We konden 1 scenario simuleren waarin het aantal ongelukken 0 was, namelijk bij een drempel die auto’s vertraagd naar 20 en een afstand van 8 tussen persoon en auto. Echter is het niet realistisch dat er ook nooit een ongeluk zou plaats vinden als er drempels worden geplaatst die de snelheid vertraagd naar 20. Dit heeft te maken met 2 dingen:

* In scenario “speedbump 10” en afstand tussen auto en persoon 8, is het aantal ongelukken groter dan 0. Als bij een steile helling het aantal ongelukken niet voorkomen kan worden zal dat ook niet zo zijn bij een minder steilere helling.
* We hebben geen invloed op de afstand tussen persoon en auto. Het kan dus zomaar zijn dat er mensen vlak voor de auto rennen.

Onze hypothese klopt, aan de hand van onze simulatie, dus niet.

Ondanks dat onze hypothese niet klopt, en dat we het ongelukken niet 100% kunnen voorkomen, adviseren we het volgende op basis van ons experiment:

*tabel 3*

|  |  |  |  |
| --- | --- | --- | --- |
| **mogelijkheden** | **Kans op ongeluk** | **Vermindering kans op ongeluk** | **omstandigheden** |
| Speedbump 30 | 5.5% - 10.4% | 40.0% | Rustige wegen, geen oversteekplaatsen en buiten bebouwde kom. Er kan goed worden geanticipeerd en de afstand tussen persoon en auto is groot. |
| Speedbump 20 of 10 | 4.4% - 8.6% /  4.2% - 8.4% | 41.4% / 41.6% | Drukke wegen, binnen bebouwde kom en veel zijstraatjes. Er kan niet goed worden geanticipeerd en de afstand tussen persoon en auto is klein. |

**Discussie**

In het kopje discussie gaan wij verder in op punten wat er allemaal goed ging bij het maken van de simulatie, wat er beter kan, en wat we eventueel nog kunnen aanpassen.

**Is het resultaat wat we verwacht hadden**

Na het uitvoeren van het experiment hebben wij de resultaten bekeken. De resultaten zagen er ongeveer uit hoe wij verwacht hadden dat de resultaten er uit zouden zien. Wij hadden namelijk verwacht dat als de persoon een grotere afstand had met de auto, dat er een kleinere kans is op een ongeluk. En dat deze kans nog kleiner zou worden als er een drempel is, per afstand met de auto. Wat wij niet verwacht hadden is dat het verschil van de kans op een ongeluk met een verkeersdrempel die afremt tot 20 km/h en 10 km/h niet heel groot is.

**Tonen resultaten in NetLogo**

Het tonen van de resultaten in NetLogo was iets wat minder goed ging. Dit kwam omdat wij niet zo heel veel ervaring hebben met NetLogo. Daarom hebben we de data verder verwerkt in Python met matplotlib. Hier hadden wij al eerder meegewerkt en deze resultaten staan ook bij de resultaten.

**Is de simulatie realistisch**

De simulatie die wij gebouwd hebben is deels realistisch maar deels ook niet. Het gene wat wel realistisch is was de snelheid en tijd. NetLogo werkt met coördinaten de omgeving is een soort raster, wij zijn er van uit gegaan dat 1 coördinaat gelijk staat aan 1 meter. Hiermee visualiseren wij de snelheid door uit te gaan van de snelheid in m/s.

Het gene wat niet realistisch was aan de simulatie was de rem- en acceleratiekracht. Om dit zo goed mogelijk te simuleren hebben wij een getal gekozen wat de auto per seconde sneller of zachter gaat rijden. Deze getallen zijn gekozen omdat het er wel realistisch uitzag. Maar je zou het niet kunnen vergelijken met de rem- acceleratiekracht van een echte auto. Wij verwachten dat hierdoor de resultaten ook anders zullen zijn, Dit is dan dus ook wat wij nog kunnen verbeteren als we de simulatie nog realistischer willen maken.