Simulation Stimulation

Verkeerssimulatie

Berry Hijwegen & Ype Bezema

19-12-2019

Introductie

De snelheid op de snelwegen, in Nederland momenteel een veelbesproken onderwerp. De stikstof- en CO2-uitstoot houdt de politiek flink bezig. Door de maatregelen die de politiek in eerste instantie hebben genomen ontstonden er verschillende boerenprotesten, mede hierom is besloten om ook de maximumsnelheden op de snelweg overdag te verlagen naar maximaal 100 km/h. Er wordt veel gezegd over de mogelijke gevolgen van deze maatregel; we zullen bijvoorbeeld later thuis zijn dan normaal is het argument. Klopt dit ook?

Dit experiment heeft hier deels betrekking toe. In dit experiment wordt onderzoek gedaan naar de oorzaken van een spookfile. Hierin zit ook een stukje snelheid meegenomen. Wat is bijvoorbeeld het effect van constant 100 km/h rijden in plaats van 130 km/h? Met dit experiment zal zulke data in een simulatie verzameld kunnen worden. Met deze data kan vervolgens gekeken worden wat nou de effecten zijn van een andere snelheid op de doorstroom van de weg.

Samenvatting

In Nederland heerst een stikstofcrisis. Om deze op te lossen moeten ook de automobilisten snelheid inleveren volgens de overheid, daarom wordt de snelheid op de snelwegen verlaagd naar 100 km/h. Hier is veel ophef over, de bewering is dat het langer zou duren om van A naar B te komen.

Het doel van dit onderzoek is om een idee te geven van wat een verandering van de maximumsnelheid kan betekenen voor de daadwerkelijke snelheid op de snelwegen. Hierbij dan ook de onderzoeksvraag:

“Wat is de invloed van een bepaalde maximumsnelheid op de doorstroom van de weg?”

Deze onderzoeksvraag is onderzocht door middel van een experiment in de vorm van een simulatie. Hierbij is gebruik gemaakt van een tweebaansweg waarop de auto’s elkaar inhalen, deze omgeving is vervolgens getest met verschillende snelheden en drukte.

Na het analyseren van de data was de conclusie hierbij dat bij een hogere maximumsnelheid, de gemiddelde snelheid ongeveer gelijk blijft. De verdeling van de snelheden was wel groter hierbij, er waren dus meer uitschieters, en dus meer opstoppingen die ervoor zorgden dat auto’s vaker moesten stoppen en daardoor minder snel op bestemming komen. Hierdoor had de maximumsnelheid relatief weinig invloed op hoe lang het duurt om van A naar B te komen.

Inhoudsopgave

Onderzoeksvraag en hypothese

Onderzoeksvraag

Wat is de invloed van een bepaalde maximumsnelheid op de doorstroom van de weg?

Definities

Maximumsnelheid	De maximumsnelheid verbonden aan een weg die auto's mogen rijden.
File	Een continue waarde berekend door: $\lvert max.\ km/h - gem.\ km/h \rvert$ Die aangeeft in hoeverre er een file is, hoe hoger, hoe meer file.

Hypothese

Om tot onze hypothese te komen hebben we een stukje vooronderzoek gedaan.

Als vuistregel wordt er bij een noodstop de volgende formule aangehouden:

Remweg = ½(snelheid/10)^2*

De remweg gaat dus kwadratisch omhoog aan de snelheid. **Hierbij is onze eerste verwachting dus dat hoe hoger de maximumsnelheid is, hoe minder goed de doorstroom in de weg is.** Hierom zal er ook niet één concreet antwoord zijn op onze onderzoeksvraag, deze zal afhankelijk zijn van de minimum- en maximumsnelheid op een weg. Er zijn dus meerdere factoren die invloed hebben op deze vraag. Dit is dus bijvoorbeeld de snelheid op het moment van beginnen met remmen en de remkracht, de remkracht is hierbij iets wat verschilt per bestuurder en zijn gewoontes. Dit zal dus iets zijn om rekening mee te houden in het experiment.

$H_0 =$ Het verlagen van de maximumsnelheid heeft een negatief effect op de doorloop.
$H_1 =$ Een hogere snelheid zorgt voor een minder goede doorloop in de weg.

Plan van aanpak en toolkeuze

Opzet environment

Simulatie

- De simulatie zal 3D zijn. In unity is dit relatief niet veel lastiger om te programmeren dan een 2D omgeving. Er is gekozen voor een 3D omgeving omdat dit visueel aantrekkelijker is, en het ons interessanter leek om mee te werken dan een 2D omgeving.
- Er moeten veel agents tegelijk gesimuleerd worden (variabel).
- Stochastic: Er zijn variabelen en deze moeten individueel en per agent aangepast kunnen worden.
- Auto's moeten gespawnd worden op bepaalde momenten, aangegeven in verstelbare variabelen in de Unity UI.
- De real-time data wordt opgeslagen in een CSV bestand.

Environment

Om de onderzoeksvragen te kunnen beantwoorden is er een simulatieomgeving opgezet zoals beschreven in het eerder genoemde plan van aanpak, hierbij is gekozen om alleen de essentiële onderdelen van de simulatie te implementeren, het gaat hier vooral om de snelheid, en het houden aan de regels (afstand houden, rechts rijden).
Er is gekozen om een rechte tweebaansweg te gebruiken omdat dit een realistisch beeld geeft van waar veel vertraging kan oplopen. Hierbij is het ook zo dat dit ervoor zorgt dat er minder factoren zijn die invloed hebben op de resultaten van de simulaties.
Auto's worden aan het begin van de baan aangemaakt met een willekeurige snelheid rondom de maximumsnelheid, en aan het einde van de baan vernietigd worden deze weer vernietigd.

- Een lange weg met een aantal banen.
- Geen bochten, kruispunten, of verkeerslichten
- Er zal een min/max snelheid zijn, die overschreden kan worden.
- Autos moeten op de weg blijven, en kunnen niet door elkaar heen.

Agent

Om een rijbaan strak te kunnen volgen, wordt gebruik gemaakt van twee lijnen aan navigatie punten. De agents volgen deze lijnen, en kunnen van baan switchen door de andere lijn te gaan volgen. De navmeshagent van unity heeft al functionaliteit om andere objecten te vermijden. Dit zal ertoe leiden dat de agents snelheid gaan verminderen als een andere agent in de weg zit. Deze snelheidsvermindering wordt gebruikt voor de logica wanneer van baan gewisseld moet worden.
Ook wordt er gebruikt gemaakt van raycasten. Agents kunnen een stuk voor zich uit kijken, om te kijken of ze een andere agent naderen, dan kunnen ze vaak beter van baan wisselen. Het zien van andere agents wordt gedaan door middel van boxcasting.

- Positie berekenen op basis van snelheden (rijden)
- Kans voor remkracht
- Volgafstand
- Auto's moeten elkaars locatie weten, kan door middel van raycasting (boxcasting)
- Regels
    - Rechts rijden
    - Links inhalen

GUI

- Real-time data (tijd, remmende auto's, snelheden op bepaalde plekken).
- Live snelheid, state (remmend/niet-remmend) van auto inzien door erop te klikken
- Reset omgeving

Toolkeuze

Om eerdergenoemde omgeving op te zetten is er gekozen voor de tool Unity, gebaseerd op onderstaande SFA tabel (en een interesse om Unity te leren):

Algemeen (1 point, max 5 points)

Vraag	Unity points + why	Netlogo points + why
Performance efficiency: How long does running your simulation take? Is the tool fast enough?	(3) Unity is een relatief zwaar programma. Dit is overigens alleen voor de developer zo. Simulaties kunnen geëxporteerd worden naar verschillende formaten, als standalone Windows-applicatie, of WebGL. Om hier simulaties in te doen en dit iteratief te doen is wel een opgave.	(4) Netlogo is een relatief licht programma. Een nadeel is dat simulaties niet te exporteren zijn; je zult dus geen stand-alone kunnen runnen.
Compatibility: Are you using external data, and does your tool support this use?	(5) Het is erg makkelijk om assets van andere in een Unity project te implementeren, en even zo makkelijk om die assets aan te passen. Unity heeft een store om makkelijk deze assets te vinden en direct in een project te implementeren.	(4) Er zijn assets online te vinden, (bijvoorbeeld deze voor simpele verkeerswegen), en deze zijn ook modificeerbaar.
Are the skills of the developers sufficient to use the tool (user-friendliness)?	(4) Unity is een complexe tool, maar voor deze simulatie hebben wij genoeg kennis. Hiernaast willen wij graag leren om met Unity te werken.	(3) Netlogo is gebruiksvriendelijk, maar wij hebben er geen aanraking mee gehad.
Is it technically feasible to create an MVP in two weeks with the current tool?	(4) De MVP is simpel in assets, dus we kunnen al onze tijd steken in het modelleren van de simulatie. Tevens verwachten wij dat de skills die wij nog niet bezitten (GUI's in Unity) makkelijk geleerd zijn.	(1) Gezien wij geen ervaring hebben met Netlogo, en dus waarschijnlijk de helft van het project bezig zijn met bekend raken, verwachten wij niet dat we een MVP af kunnen krijgen.
Totaal	**16**	**12**

Biedt deze tool ondersteuning om de modules te maken? (1 point, max 5 points)

Vraag	Unity points + why	Netlogo points + why
Simulatie - De simulatie zal 3D zijn.	4. 3D in Unity is niet moeilijker dan 2D. Hiermee zal de simulatie wel zwaarder zijn.	3. Volgens Netlogo zelf is 3D minder supported dan 2D.
Simulatie - De real-time data wordt opgeslagen in een CSV bestand.	4. In C# werken met bestanden werkt redelijk eenvoudig.	1. Werkt vrijwel niet.
Simulatie - Er moeten veel agents tegelijk gesimuleerd worden (variabel).	4. Unity is relatief zwaar. Grafisch is Unity erg sterk, maar dit komt natuurlijk met een hoop rekenkracht. Dit zorgt ervoor dat het maximum aantal agents lager zal zijn dan bij Netlogo, al denken wij niet aan dit maximum te komen.	5. Netlogo is relatief lightweight, waardoor er meer agents tegelijk in de simulatie kunnen leven. Hierbij komt natuurlijk dat het grafisch minder sterk is dan Unity.
Simulatie - Er zijn variabelen en deze moeten individueel en per agent aangepast kunnen worden.	5. In Unity is het zeer eenvoudig om verstelbare parameters in te stellen, waardoor dat niet meer in de code gedaan hoeft te worden en dus in de UI aangepast kan worden.	5. Netlogo biedt dezelfde opties als Unity in het instellen van parameters in de UI.
Simulatie - Auto's moeten gespawnd worden op bepaalde momenten, aangegeven met verstelbare variabelen.	5. Idem, daarnaast is het makkelijk om in Unity een object te gebruiken die alle auto's mooi op 1 plek in de lijst van objecten houdt.	3. Idem, daarnaast is het in Netlogo lastiger om overzichtelijk te houden wat en waar je objecten zijn.
Environment - Een lange weg met een aantal banen.	4. Er zijn veel assets beschikbaar voor Unity. De meeste betaald maar ook veel gratis.	2. Er is weinig te vinden over de 3D components binnen Netlogo.
Environment - Er zal een min/max snelheid zijn, die overschreden kan worden.	5. Goed te doen met de Scripting API en verstelbare parameters.	5. Idem.
Environment - Autos moeten op de weg blijven, en kunnen niet door elkaar heen.	5. Door in Unity gebruik te maken van colliders kan er makkelijk gecheckt worden op collisions.	5. Checken op collisions in Netlogo is relatief eenvoudig.
Agent - Er zijn een aantal verkeersregels die gesimuleerd worden, zoals niet onnodig links	5. Door middel van raycasting in Unity kunnen deze regels gecheckt en afgehandeld	2. Raycasting is in Netlogo een stuk lastiger.

rijden en niet onnodig links inhalen.	worden.	
Agent - Positie berekenen op basis van snelheden (rijden)	4. Met de Scripting API van Unity kan eenvoudig met posities gewerkt worden. Deze worden opgeslagen in een Vector3 class, wat relatief eenvoudig werkt.	4. Netlogo zal hetzelfde bevatten.
Agent - Kansberekening voor remkracht	5. Dit is puur wiskundig te berekenen.	5. Dit is puur wiskundig te berekenen.
Agent - Volgafstand	5. Eerdergenoemde raycasting.	2. Eerdergenoemde raycasting.
Agent - Auto's moeten elkaars locatie weten, kan door middel van raycasting, of puur elke auto locatie checken.	5. Idem.	2. Idem.
GUI ● Real-time data (tijd, remmende auto's, snelheden op bepaalde plekken). ● Live snelheid, state (remmend/niet-remmend) van auto inzien door erop te klikken ● Reset omgeving	4. Door middel van de Scripting API kunnen GUI buttons en overlays in de interface laten zien worden. Er is geen optie om dit in de visuele developersinterface te doen.	4. Netlogo bevat opties om buttons en overlays toe te voegen.
Totaal	**64**	**49**

Uitleg experiment

Dataverzameling

Om de bovenstaande vragen te beantwoorden zal de volgende data verzameld moeten worden in de simulatie:

- Per uitgevoerde simulatie
    - Simulatienummer
    - Maximumsnelheid van de weg
    - Spawn rate
    - Maximaal aantal auto's in de simulatie

- Per Auto
    - Autonummer
    - Simulatienummer
    - Stapnummer
    - Huidige snelheid van de auto

Deze data wordt opgeslagen in twee CSV-bestanden.

- Eén bestand met één rij per simulatie met simulatienummer, combinatienummer, spawn rate, maximumsnelheid, en maximaal aantal auto's.
- Een tweede bestand waarin voor elke auto elke tick in een simulatie de snelheid opgeslagen wordt, hier zijn de kolommen dus: simulatienummer, autonummer, huidige tick, huidige snelheid.

Uitvoering

De uitvoering van het experiment gebeurd door middel van het iteratief uitvoeren van de simulatie. Hierin worden alle combinaties van de volgende parameters gebruikt:

- Spawn rate [60, 80, 100, 120] (delay in ticks)
- Maximumsnelheid [3.5, 4, 4.5, 5] (Unity units per tick)
- Maximaal aantal auto's [10,20,40,60]

Elke combinatie wordt drie keer 2500 ticks lang getest. Dit wordt gedaan door middel van een script die de simulatie beheerd. Iteratief worden de parameters van de run veranderd en opgeslagen. Om op deze informatie terug te kunnen refereren wordt er per auto ook een simulatienummer bijgehouden. Elk simulatienummer heeft een rij in een tweede csv-bestand waar alle drie de parameters bij staan. Op deze manier kunnen die in de analyse gekoppeld worden. Auto’s spawnen met een willekeurige snelheid tussen 1.75 en 5.25. Deze worden vervolgens gecorrigeerd als deze snelheid te snel is. Na elke run worden de auto’s vernietigd.

Resultaten experiment

Na het uitvoeren van het experiment zijn de verzamelde gegevens geanalyseerd om verbanden te vinden die antwoorden geven op de onderzoeksvraag.

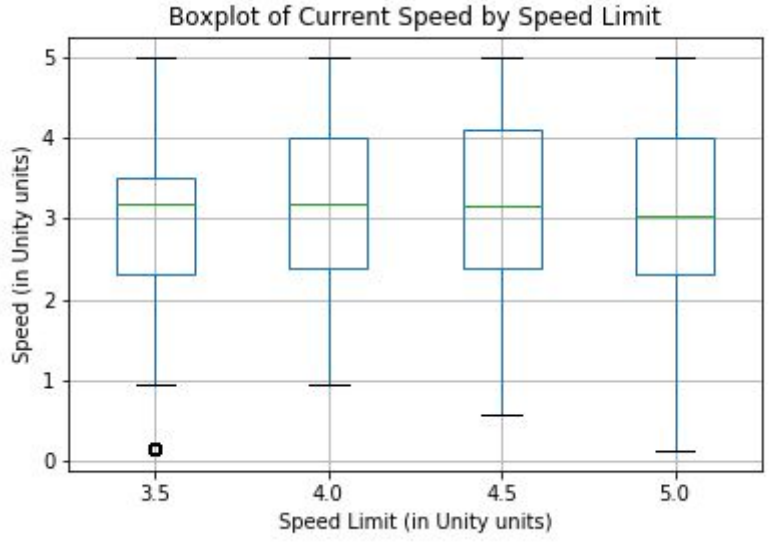
Cijfers

In het experiment zijn er drie verschillende parameters gebruikt waarvan vermoed werd dat deze invloed zouden hebben op de doorstroom van de weg:

- Bezetting van de weg (Aantal auto's)
- Maximale snelheid
- Hoe snel er nieuwe auto's bijkomen

Verdeling snelheden

Om een beeld te krijgen van de belangrijkste data, hieronder een weergave van de verdelingen van de snelheid die auto's rijden bij een bepaalde maximumsnelheid.

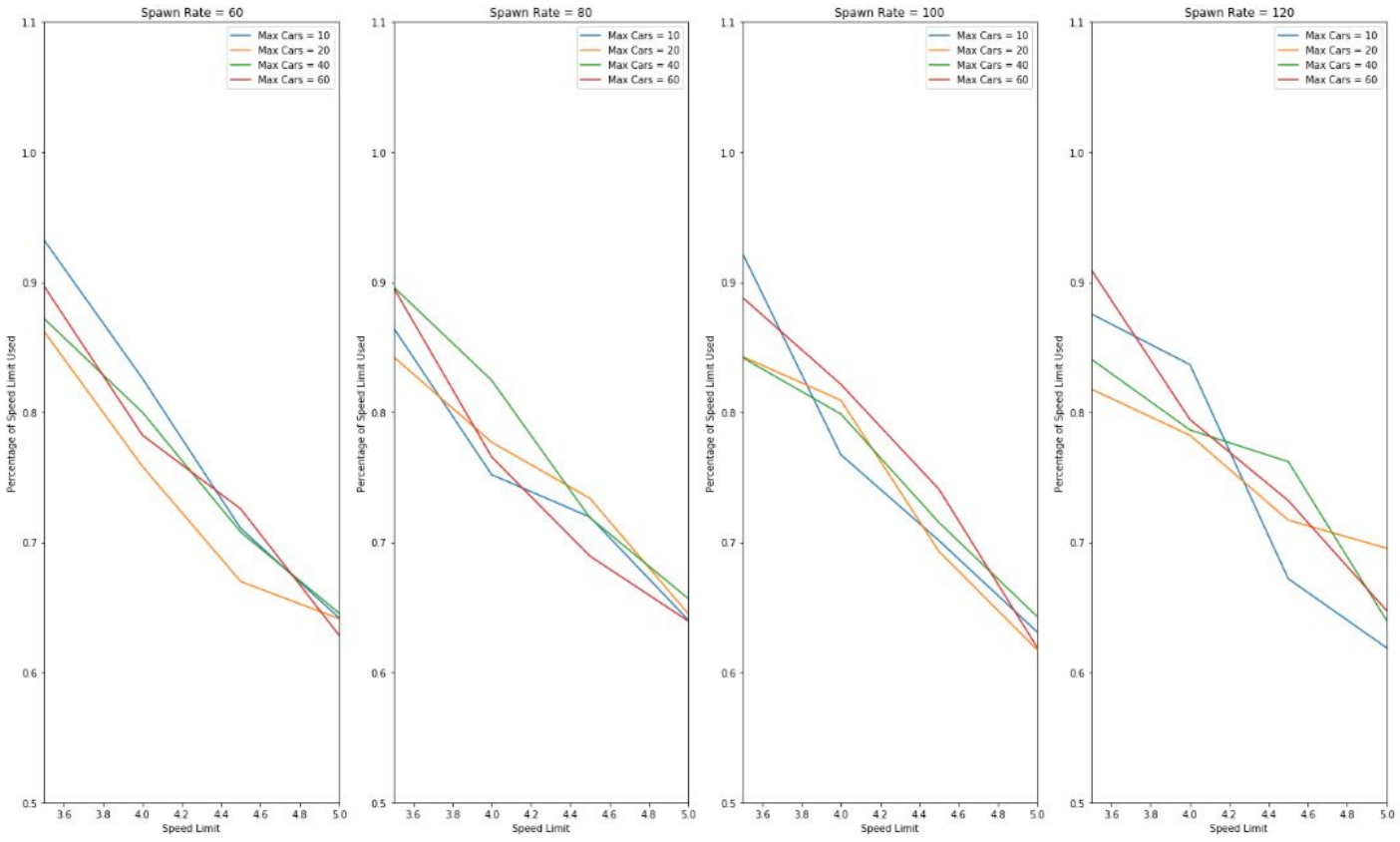


Grafiek met boxplotten om verdeling te weergeven. X-as weergeeft de maximumsnelheid, Y-as weergeeft de daadwerkelijk gereden snelheden.

Hieruit blijkt al dat de gemiddelde snelheid lager ligt bij een hogere snelheid. De verdeling wordt wel steeds ruimer naarmate de snelheid hoger is.

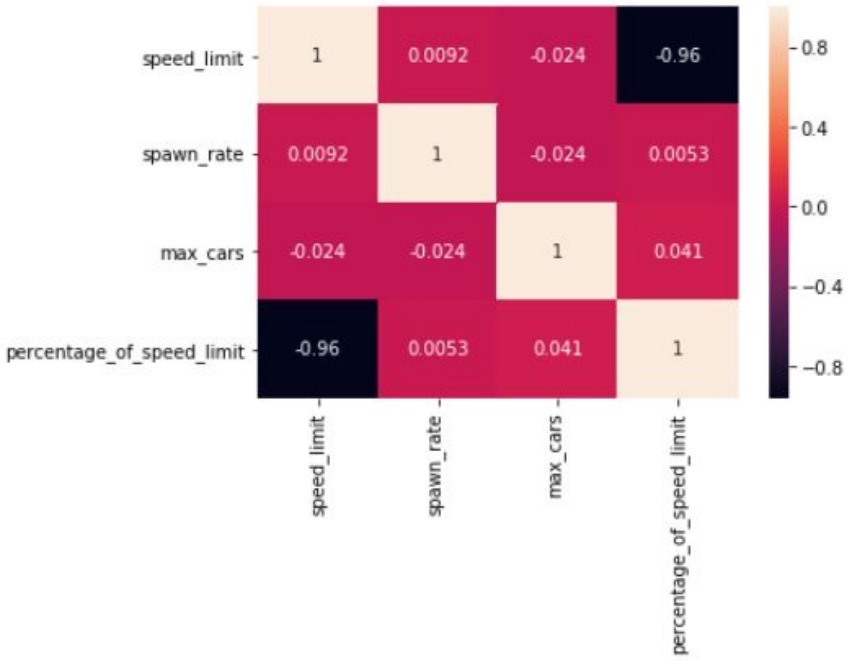
Verbanden

In onderstaande grafieken zijn alle factoren verwerkt. Hieruit Is al te zien dat er een sterk negatief verband is tussen de maximumsnelheid en het percentage van de maximumsnelheid die ook daadwerkelijk gereden wordt. Er is te zien dat bij maximumsnelheid = 5.0, alle lijnen tussen de 60% en 70% uitkomen, dit betekent dat er hier gemiddeld maar tussen de 3.0 en 3.5 gereden werd, een heel verschil met de snelheid die gereden mag worden.



Grafiek om verbanden te weergeven. Elke grafiek heeft een aparte spawn rate, elke lijn staat voor het aantal maximale auto's. X-as bevat de maximumsnelheid, y-as weergeeft het percentage van de maximumsnelheid die gereden is.

Om deze verbanden in nummers te weergeven, is hierbij een correlatiematrix.



Correlatiematrix op verbanden beter in nummers te weergeven.

Hierin wordt het verband wat al gezien kon worden bevestigd. Een negatieve correlatie van -0.96 tussen de maximumsnelheid en het percentage wat daarvan ook daadwerkelijk gereden wordt. In de boxplots was dit al te zien, de gemiddelde snelheid bleef ongeveer gelijk ondanks de hogere snelheid.

Conclusie

De onderzoeksvraag van het experiment was als volgt:

“Wat is de invloed van een bepaalde maximumsnelheid op de doorstroom van de weg?”

Vanuit de analyse van de resultaten van het simulatie kan geconcludeerd worden dat er een sterke correlatie is tussen de maximale snelheid en de doorstroom van de weg. Vanuit de resultaten bleek dat er een correlatie was van -0.96 was tussen de maximumsnelheid en het percentage van de gemiddelde snelheid ten opzichte van de maximumsnelheid, oftewel, hoe hoger de maximumsnelheid, hoe minder hier gebruik van gemaakt kon worden.

De gemiddelde snelheid kwam als percentage steeds verder af te staan van de maximumsnelheid. Dit kwam doordat de gemiddelde snelheid eigenlijk gelijk bleef. De oorzaak hiervan is terug te zien in grafiek met boxplots, bij de hoogst geteste snelheid is de spreiding veel groter dan bij bijvoorbeeld de eerste plots. Auto’s hebben dus vaker stil gestaan.

Volgens dit onderzoek zou het dus niet uitmaken of de snelheid verlaagd wordt, door minder opstoppingen zou de gemiddelde snelheid toch gelijk blijven en de doorstroming dus beter worden dan bij een hogere snelheid.

Discussie

Waar liepen we tegenaan in het proces?

Unity is een complexe tool. De simulatie bleek lastiger om te bouwen dan we dachten, omdat er vrij vaak bugs kwamen opdagen waar we dan weer een dag mee bezig waren. Raycasten met navmeshagents, om een voorbeeld te noemen, was te onbetrouwbaar, maar was wel nodig om rijstrookwisseling goed te kunnen implementeren. Uiteindelijk is dit opgelost door boxcasting te gebruiken, in plaats van raycasting.

Relatief vaak kwam er gedrag voor vanuit de Unity libraries die naar onze mening onverwacht waren, dit zal grotendeels te maken hebben met dat wij Unity nog niet zo goed kennen en dus langer bezig zijn met het uitzoeken van hoe het nou kan dat zulke dingen ontstaan.

Omtrent performance was er ook nog wel een punt. Het experiment uitvoeren duurde relatief lang. Dit heeft naar ons idee de volgende redenen:

- Wij hebben ons experiment in de editor-omgeving uitgevoerd, en niet in een build. Hierdoor zullen er nog een aantal debug-features van Unity actief invloed hebben gehad op de performance.
- Onze kennis van C# is niet op hetzelfde niveau als dat van Python. Wij zullen dus bepaalde dingen niet goed geoptimaliseerd hebben in de code.
- Het visuele aspect van Unity biedt veel mogelijkheden; maar vraagt ook heel veel berekeningen. Het was daarom misschien beter geweest om in een 2D omgeving te werken.

Gevolgen voor betrouwbaarheid van experiment

De simulatie die gebruikt is in het experiment is niet perfect. Zo komt het soms voor dat auto’s die maar iets sneller rijden dan de auto voor zich, gaan inhalen. Hierdoor blijft de linkerrijbaan soms lange tijd bezet. Dit is een versperring voor snellere auto's. Dit is nog iets om te verbeteren aan de simulatie om een betrouwbaarder en realistischer experiment uit te voeren.

In hoeverre dit experiment dus iets zegt over de daadwerkelijk situatie wanneer men minder hard moet gaan rijden is dus discutabel. Hiervoor zal er een complexere omgeving opgezet moeten worden.

Daarnaast zou er voor een hogere betrouwbaarheid meer data verzameld moeten worden.

Debugging

Unity heeft zeer sterke debugging tools ingebouwd, wat onze code van een hoop test functies heeft bespaard. Met breakpoints konden we makkelijk er achter komen of stukken van onze code nooit bereikt werden, of juist of ze op de verkeerde momenten getriggerd werden. Zo ook konden we door de variabelen publiek te maken makkelijk de waarden checken in de inspector, en zien waar het fout gaat. Door middel van het pauzeren van de simulatie konden we goed inzoomen op colliders en de buitenranden van de objecten om zo ook te zien waar het mis ging met raycasting.

Onze code heeft daarom vrijwel geen debugging functionaliteit. Een uitzondering hiervan is het tekenen van de raycasts (boxcasts om specifiek te zijn).