Code, Experiment, Evaluate, Repeat

Verkeerssimulatie

Berry Hijwegen & Ype Bezema

13-12-2019

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.

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 exponentieel 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 veranderen van de maximumsnelheid heeft geen invloed op de doorloop van de weg.
$H_1 =$ 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 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:

- 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:

- Positie berekenen op basis van snelheden (rijden)
- Kans voor remkracht
- Volgafstand
- Auto's moeten elkaars locatie weten, kan door middel van raycasting, of puur elke auto locatie checken.
- 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:

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?	(4) 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.	(3) 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.	(2) Netlogo is user friendly, 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	**17**	**10**

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.	5. 3D in Unity is niet moeilijker dan 2D.	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 databases 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 rijden en niet onnodig links inhalen.	5. Door middel van raycasting in Unity kunnen deze regels gecheckt en afgehandeld worden.	2. Raycasting is in Netlogo een stuk lastiger.
Agent - Positie berekenen op basis van snelheden (rijden)	4. Met de Scripting API van Unity kan eenvoudig met posities gewerkt worden. Deze	4. Netlogo zal hetzelfde bevatten.

	worden opgeslagen in een Vector3 class, wat relatief eenvoudig werkt.	
Agent - Kansberekening voor remkracht	5. Dit is puur wiskundig berekenen.	5. Dit is puur wiskundig 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	**65**	**49**

Uitleg experiment

Voor het experiment zijn de volgende onderzoeksvragen opgesteld:

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

Omgeving

Om deze onderzoeksvragen te kunnen beantwoorden is er een simulatieomgeving opgezet zoals beschreven in het eerder genoemde plan van aanpak, hierbij is gekozen om vooralsnog 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). Hierbij is het inhalen deels geïmplementeerd in verband met technische problemen, dit zal later worden benoemd in de discussie.
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 gespawnd met een willekeurige snelheid rondom de maximumsnelheid, en aan het einde van de baan vernietigd.

Agents

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. Raycasten op navmeshagents bleek echter onbetrouwbaar te zijn, wat dus iets is om rekening mee te houden.

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

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 2500 ticks lang getest.

Deze data wordt opgeslagen in twee CSV-bestanden.

- Eén bestand met één rij per simulatie met simulatienummer, 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.

Resultaten experiment

Resultaten van het experiment te vinden in het volgende notebook:

https://github.com/berryhijwegen/Traffic_Simulation/blob/master/docs/data/Notebook_results.ipynb

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 -0.86 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.

Hierbij zijn er wel een aantal punten die de betrouwbaarheid van het onderzoek niet goed doen:

- De simulatie was niet realistisch genoeg om een uitspraak te doen over het daadwerkelijke verkeer. Zo bleven auto’s zodra ze in gingen links rijden, terwijl dit in het echte verkeer niet zo is, het is de bedoeling dat een bestuurder zo rechts mogelijk blijft rijden.
- Elke combinatie is één keer uitgevoerd. Aangezien er variabelen gebruikt zijn die willekeurige waarden mee hebben gekregen, zou de simulatie vaker uitgevoerd moeten worden dan één keer om de willekeurigheid eruit te halen.

In hoeverre dit experiment dus iets zegt over de daadwerkelijk situatie wanneer men zachter moet gaan rijden is dus discutabel. Hiervoor zullen er meer tests moeten worden gedaan in een complexere omgeving.

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. Onze simulatie is uiteindelijk dus wat simpeler geworden, om het zo bug-free mogelijk te houden.

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 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

In de simulatie die gebruikt is bij het experiment mist nog een essentieel deel van het verkeer, wanneer auto's namelijk inhalen, gaan ze niet meer terug naar de rechterbaan, wat in het echte verkeer natuurlijk wel zo is. Dit zorgt ervoor dat wanneer een relatief trage auto gaat inhalen, en deze op de rechterbaan blijft, een versperring gaat zijn voor snelle auto's. Dit is nog iets om te verbeteren aan de simulatie om een betrouwbaarder en realistischer experiment uit te voeren.