Casus: *Rotmaatregel*

Lucas van Wijk, Roeland Oostdam, Freek Gerrits Jans

Abstract

De overheid heeft besloten dat het snelheidslimiet op alle snelwegen wordt vastgezet op 100 km/u. Dit betekend een verschil tot 30 km/u. Hierbij beweert de overheid dat het niet alleen beter voor het milieu is, maar dat het voor de doorstroom van het verkeer niet zoveel zou moeten uitmaken. Sterker nog, de doorstroom zou in sommige gevallen zelfs verbeteren. Wij proberen door middel van het gebruik van een simulatie te achterhalen of dit klopt. Hiermee laten wij zien wat het effect op doorstroom van het verkeer is bij verschillende snelheidslimieten i.c.m. wegbezettingen. Uit deze simulatie blijkt dat een hogere snelheid tot een bepaalde wegbezetting optimaal is, en dat bij een te drukke weg een hoger snelheidslimiet averechts werkt.

Introductie

De theorie van het shockwave effect vertelt ons dat wanneer er een auto remt op bijvoorbeeld een drukke snelweg, er een shockwave effect ontstaat. Dat wil zeggen dat alle auto's achter de remmende auto ook moeten remmen en een opstopping ontstaat. Snelheid speelt bij een shockwave effect een rol. Door middel van een simulatie en een visualisatie wordt het shockwave effect in kaart gebracht. Met deze informatie bepalen we of er een duidelijk verschil zit in het effect van een shockwave en de snelheid. We laten zien dat bij bepaalde weg druktes het handiger is om een lager snelheidslimiet toe te passen dan 130 km/h. Hiervoor zullen wij gebruik maken van het Nagel-Schreckenberg model [1].

Onderzoeksvraag

Onder welke omstandigheden heeft het verlagen van de maximum snelheid een positieve invloed op de doorstroom van het verkeer.

Het plan

Het is de bedoeling om de simulatie in een GUI te maken waarin de omgevingsvariabelen makkelijk aan te passen zijn.
De omgevingsvariabelen zijn als volgt:

- Verkeersdrukte (Auto's per minuut)
- Snelheidslimiet
- Acceleratie
- Deceleratie
- Zichtbereik

Dit zijn veel variabelen. Omdat wij het effect van het aanpassen van de snelheidslimiet willen bepalen moeten dit effect uiteraard geïsoleerd worden. Wat betekent dat wij in eerste instantie de omgevingsvariabelen hetzelfde laten, zodat wij kunnen bepalen dat dit geen effect heeft gehad op het verschil in fille. In een later stadium willen wij kijken of de gevolgen van het veranderen van de snelheidslimiet anders zijn bij andere omgevingsvariabelen. Wij willen dan kijken of wij algemene regels kunnen opstellen wanneer het verstandig is om de snelheidslimiet te veranderen. Bijvoorbeeld in de vorm van een stelling als: "Als de verkeersdrukte lager is dan x is het verstandig om de verkeerssnelheid te verhogen".

Een tijdstap in deze simulatie is de tijd dat een auto erover doet om 10 autolengtes te verplaatsen. (Volgens het Nagel-Schreckenberg model). In elke tijdstap zal elke agent zijn Perceive, Update, Act, Move uitvoeren. (Hierover meer in "opbouw"). De wereld is statisch die zal niks doen bij een nieuwe tijdstap.

In de GUI is het mogelijk om per stap te simuleren en te pauzeren. Tussendoor wordt door middel van grafieken de any state gevisualiseerd. Wij zullen de eigenschappen van elke agent bij elke stap opslaan. Op deze manier kunnen wij zien of er tijdens of na de simulatie opvallende dingen zijn en achterhalen in welke staat de agents zich bevonden en welke acties ze hierdoor hebben ondernomen. De GUI zal ook een grid hebben waar de agens op te zien zijn. Zo kunnen wij tijdens de simulatie de verschillende auto's en de verkeersdrukte zien.

Opbouw

Ons project heeft een autobaan. Op deze weg zijn een aantal auto's gemodelleerd die daar rijden. Deze auto's hebben een bepaalde vision range en een maximum snelheid. Sommige auto's zullen random afremmen. De auto's zullen proberen te voorkomen om te botsen. Bij elke stap zal de auto vier acties uitvoeren:

1. Perceive
   Bij perceive kijkt de auto of er andere auto voor hem is en hoe ver.
2. Update
   Hierin wordt bepaald welke van de hieronder in "verschillende state" beschreven states de auto zal krijgen.
3. Act
   Hier zal de auto zijn snelheid aanpassen naar de snelheid van de huidige state.
4. Move
   De auto verplaatst zich naar voren met de huidige snelheid.

Verschillende states

Er zijn drie verschillende states die een auto kan hebben. Alle states hebben invloed op de snelheid van een auto. Wanneer de state wordt uitgelezen en uitgevoerd zal er het volgende gebeuren per state.

- acceleration : Bij deze state zal de snelheid per step een vooraf gegeven eenheid in snelheid stijgen.
- braking : Bij deze state zal de snelheid per step een vooraf gegeven eenheid in snelheid dalen.
- randomBraking : Bij deze state zal een auto random per step zoveel in snelheid afremmen dat er altijd minimaal 1 eenheid in steps tussen de auto's blijft.
- cruising : Bij deze step zal de auto de snelheid aanhouden van de vorige step. anders gezegd de snelheid veranderd niet.

User stories en taakverdeling

Projectstructuur

er zijn drie grote onderwerpen in het in de simulatie. “Agent behavior”, “Environment”, “GUI, visualization and data collection”. Voor structuur hebben wij besloten om deze drie onderwerpen globaal te verdelen onder de drie teamleden. Elk lid wordt dan expert op het gebied van dat onderwerp. User stories van dat onderwerp kunnen nog wel door andere gedaan worden zodat niet een iemand al het werk doet maar een team lid blijft de expert en de persoon verantwoordelijk voor het inzicht houden wat er nog gedaan moet worden voor dat onderwerp.

De rollen zijn als volgt verdeeld
Chief Agent behavior: Roeland
Chief Environment: Freek
Chief GUI, visualization and data collection: Lucas

De tijd voor sommige user stories die gepland is lijkt soms wat ruim. Dit komt omdat het voor ons nu moeilijk te zeggen is welke obstakels we tegen gaan komen. Het genereren van een grid voor de visualisatie kan 10 minuten duren als alles goed gaat. Maar het kan ook 6 uur duren als alles fout gaat. Een gedeelte van dit risico hebben we dus meegenomen in de tijdsbepaling.

User stories

Agent behavior	Who	Time in hours
Function to move agent based on it's speed	Roeland	1
Determining speed for current step	Roeland	1.5
Function to determined distance to car ahead	Roeland	0.75
Function to prevent collision	Roeland	0.75
Function to determine state according to NSM	Roeland	2

Environment	Who	
Apply correct agent behavior on tick change	Freek	3
Generate and populate grid	Freek	0.25
track world information	Freek	0.5

GUI, visualization and data collection	Who	
Generate a grid on which you see the agents move	Lucas	1
Add user input fields to change variables for the simulation	Lucas	1
Ensure simulation is restarted in user input	Lucas	0.2
Determine how to express traffic in a numeric variable/expression/characteristic	All	2
Collect determined variable to express traffic over time	Lucas	2
Plot determined variable to express traffic in the GUI	Lucas	1

Overhead		
Making the Poster	All	6*
Preparing presentation	All	6*

*2 hours per person

Tool keuzen

Suitability

Tool supports coding the modules described in part 2. For each of your modules, rate how well your tool is suited.

GUI

Unity: 7. Unity heeft veel variabelen al in een format staan. Denkend aan zwaartekracht, lengte van een auto of de versnelling van een auto. Deze variabelen zijn handig om de beïnvloedende factoren te beschrijven bij een dergelijk simulatie. De simulatie kan in 3D worden weergegeven wat erg mooi oogt, echter hebben we genoeg aan een 2D simulatie voor deze opdracht. Om de tool te snappen hebben we erg veel tijd nodig, iets wat we niet hebben voor dit project.

Mesa: 8. de GUI die Mesa ons biedt is goed voor deze opdracht. De tool geeft de mogelijkheid in 2D een simulatie te bouwen. Erg handig dat Mesa met Python werkt, want iedereen uit de groep heeft ervaring met Python. Daarnaast kunnen door middel van andere modules data goed gevisualiseerd worden. Denk hierbij bijvoorbeeld aan de modules matplotlib, holoview of seaborn.

Netlogo: 6. De GUI van deze tool is ook voldoende. Het laat een 2D beeld zien van de simulatie. Echter is er alleen een ingebouwde module welke grafieken kan laten weergeven. De vrijheid van keuze is daardoor kleiner dan de andere tools.

Time

Unity: 6. Heeft verschillende manieren om objecten te laten bewegen op tijd.

Mesa: 8. Door de module Simultaneous Activation te gebruiken kunnen alle auto's eerst te horen krijgen wat een deze agent moet doen dat tijdstip en vervolgens als alle auto's een opdracht hebben gekregen doen zij allen tegelijkertijd wat hen is opgedragen.

Netlogo: 3. Bij Netlogo kan men ask turtles (in ons geval auto's) gebruiken. Hiermee krijgen alle auto's een commando. De auto's kunnen per auto worden aangesproken of allen tegelijkertijd met allen hetzelfde command.

Collect data

Unity: 8. Bij Unity kan makkelijk states worden opgeslagen en uitgelezen. Met de inspector methode is het mogelijk om te kijken wat een bepaalde auto zijn internal state is.

Mesa: 7. Doordat Mesa de taal Python gebruikt als taal kan er in classes objecten worden opgeslagen waarin dus data makkelijk kan worden opgeslagen en uitgelezen. .

Netlogo: 4. Heeft de mogelijkheid om agents te inspecteren. Echter is het gebruik van deze data lastig.

Interactive GUI

Unity: 6. Unity is een uitgebreid programma gemaakt voor visualisatie en interactie. Dit maakt interactie met de simulatie erg makkelijk. Unity bied een grote scala aan mogelijkheden binnen de interactie met de GUI. Dit zal voor ons project niet gewenst zijn, omdat we een gering aantal opties nodig hebben betreft de interactie met de GUI.

Mesa: 7. Mesa kan door middel van knoppen en een slider makkelijk variabelen veranderen. Dit is precies goed voor onze simulatie.

Netlogo: 7. Netlogo kan ook makkelijk variabelen aanpassen door middel van knoppen en sliders. Ook deze mate van interactie is goed voor onze simulatie.

Agent

Unity: 8. Bij de module agents van Unity kun je erg veel details geven aan een specifieke agent. De basis bevat al erg veel details ten opzichte van wat nodig is voor deze opdracht.

Mesa: 9. De module geeft ons een mogelijkheid om object oriented te werk te gaan. We kunnen zo per agent erg fijn aangeven wat een state kan zijn van elke specifieke agent.

Netlogo: 3. Kan een specifieke agent of groep agents een commando geven. Tot nu toe hebben wij niet kunnen vinden hoe een state aan een agent toewijst met code.

Performance efficiency

How long does running your simulation take? Is the tool fast enough?

We schatten dat we rond de 400 stappen nodig te hebben voor de simulatie.

Unity: Unity kan zware programma's draaien en zal zeker snel genoeg zijn om onze simulatie te kunnen visualiseren. Naar ons idee is Unity te bassaal, er zijn meerdere mogelijkheden om de simulatie in andere programma's te draaien die minder zwaar zijn en minder complex.

Mesa: Mesa is naar ons idee krachtig genoeg om onze simulatie te draaien. De simulatie kan met dit programma goed gevisualiseerd worden. Daarbij zal tussen Unity 2d en Mesa 2d eigenlijk geen verschil zitten in snelheid.

Netlogo: Naar ons idee is Netlogo snel genoeg om de simulatie te runnen.

Feasibility

Are the skills of the developers sufficient to use the tool (user-friendliness)?

Unity: Unity maakt gebruik van de taal C#. Een taal die een beetje bekend is onder ons. Unity is in eerste instantie niet heel gebruiksvriendelijk. Er zou vele tutorials gekeken moeten worden, voordat er een goede simulatie kan worden gebouwd.

Mesa: Mesa maakt gebruik van Python. Een taal die erg makkelijk is in vergelijking tot andere talen naar onze mening. De modules zijn goed beschreven en kunnen makkelijk worden gebruikt.

Netlogo: Netlogo maakt gebruik van een variant van java. Deze taal is ons onbekend maar nog wel te begrijpen. De interface is erg simpel en gebruiksvriendelijk.

Is it technically feasible to create a MVP in two weeks with the current tool?

Unity: De tijd die we nodig zullen hebben om in Unity een MVP te maken zal langer zijn dan twee weken.

Mesa: We denken dat ons het gaat lukken om een Minimum viable product te bouwen binnen twee weken in Mesa met wat extra's.

Netlogo: Voor Netlogo denken we ongeveer aan een MVP te voldoen in twee weken.

Compatibility

Are you using external data, and does your tool support this use?

Voor de basis gebruiken we nog geen externe data, wellicht gaan we dit gebruiken wanneer de basis af is. Mesa ondersteund het gebruik van een externe dataset evenals Unity. Naar ons weten doet Netlogo dat niet. Wij konden geen informatie vinden over het gebruik van externe data binnen Netlogo.

Keuze

Voor de simulatie zoeken wij een programma waarin een 2d model kan worden getest. Daarnaast moet er makkelijk informatie uit de agents en enviroment worden ontleed en verwerkt kunnen worden. Daarnaast willen we een tool die wat meer gebruiksvriendelijk is, maar wel veel mogelijkheden bevat qua visualisatie van de resultaten die voortkomen uit ons project.

Als we kijken naar alle informatie uit het FSA-model zien we dat Mesa eigenlijk vaak het best naar voren komt. Voornamelijk vanwege de flexibiliteit die Mesa biedt in het combineren van modules en het gebruiksgemak ervan. De data uit de simulatie kan makkelijk worden opgeslagen bij Mesa en vervolgens gebruikt worden.