Simulation Tooling - Opdracht 4

1. Formulate your experiment

Wij willen een spookfile, of shockwave traffic jam, simuleren in Unity. De centrale onderzoeksvraag waar wij benieuwd naar zijn is in hoeverre je zou kunnen remmen voordat je een spookfile veroorzaakt. Hierin definieren we een file als een situatie waar een groep autos geforceerd wordt om onder de maximumsnelheid te rijden om botsingen te voorkomen. Het remmen wordt gedefinieerd als een percentage afwijking van de snelheid die gemiddeld in die baan wordt aangehouden.

2. Plan

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 SQL database.

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)
- Kansberekening 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

- Simulatie is inaccessible voor agents; ze zullen niet alles over de omgeving weten.
- De simulatie is deterministisch. We willen slechts shockwave traffic jams simuleren, dus we gaan er van uit dat agents altijd zullen remmen wanneer dat nodig is. Het word niet mogelijk voor een agent om door te rijden en een ongeluk te veroorzaken.
- De simulatie is non-episodic. De agents handelen puur op wat ze op dat moment observeren, niet op wat er eerder is gebeurd.
- De environment van de simulatie is static. De agents beïnvloeden slechts elkaar, en niet de omgeving.
- De simulatie kan oneindig door blijven gaan, en is dus continuous.

Onderzoeksvraag

Wat is het effect van een afwijkende snelheid van een bestuurder op de kans op spookfiles?

- Wat is het verband tussen het verschil van de snelheid en het verschil van de kans op een spookfile?
- Wat is het verband tussen de remweg en de kans op een 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 snelheid is, hoe meer kans er is op een spookfile.** 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.

$H_0 =$ Er is geen verband tussen een afwijkende snelheid van een bestuurder en de kans op het ontstaan een spookfile.

$H_1 =$ Het rijden van een afwijkende snelheid heeft invloed op op de kans van het ontstaan van een spookfile.

3 Tool Selection

Algemeen (1 point, max 5 points per point)

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**

Does the tool support coding the modules? (1 point, max 5 points per point)

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 SQL database.	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 worden opgeslagen in een Vector3 class, wat relatief eenvoudig werkt.	4. Netlogo zal hetzelfde bevatten.
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**