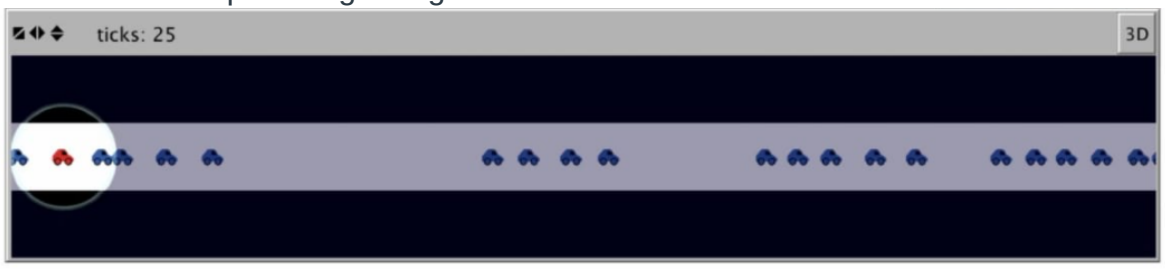
Opdracht 1

Voordelen
Netlogo is een eenvoudige taal om te leren. Het programma heeft een “plug and play” programmeertaal & omgeving waardoor je snel een simulatie kan bouwen. Ondanks dat Netlogo eenvoudig in elkaar zit, kan je met deze programmeertaal complexere modellen bouwen. Ook wordt het makkelijk om de code te delen met anderen, er is weinig uitleg nodig om geprogrammeerde modellen uit te lezen die anderen gebouwd hebben. Netlogo lijkt erg veel weg te hebben van object-oriented programming, wat het uitbreiden van code makkelijker maakt. Ook kan men gebruik maken van objects binnen een agent. Hier worden gegevens bijgehouden van de agent.

Nadelen
Netlogo is gemaakt voor 2D simulaties. Bij het bouwen van een simulatie van hoe een planeet om een andere planeet beweegt is Netlogo dus niet handig. Ik heb zelf het bestaan van de volgende functie nog niet kunnen vinden: om alle agents te laten bewegen gebruik ik momenteel een for-loop. Daardoor kunnen niet alle agents tegelijkertijd een stap vooruit zetten. In het echt zou in een kamer met slangen en muizen alles tegelijkertijd bewegen. Nu zal een slang op 500 muizen moeten wachten voordat het weer een stap kan zetten. Natuurlijk is het visuele aspect bij Netlogo ook erg simpel. Je kan een figuur wel aanpassen met “set shape mouse”, maar ook deze muis ziet er niet realistisch uit. Netlogo werkt met patches. Een blok is een patch. Zo zie je bij onderstaande foto de rode auto. Tussen deze rode auto en de eerste volgende blauwe zit 1 patch. Wanneer de auto 1 stap naar voren doet zit er dus een patch tussen. Bij het bouwen van een model wat in kaart wil brengen hoe netjes auto’s binnen de lijnen van de weg rijden, zal dit met Netlogo niet lukken. Het is daarvoor niet precies genoeg.



Is mijn programma agent-oriënted?

Hieronder staat een citaat uit een artikel dat gaat over Agent Oriënted programmeertalen.

Shoham (1993) suggereert dat een AOP-systeem de volgende drie elementen nodig heeft om compleet te zijn:

- *Een formele taal met duidelijke syntaxis voor het beschrijven van de mentale toestand. Dit omvat constructies voor het verklaren van overtuigingen en hun*

structuur (bijvoorbeeld gebaseerd op predikatenrekening) en het doorgeven van berichten.

- *Een programmeertaal waarmee agents kunnen worden gedefinieerd. De semantiek van deze taal moet nauw verband houden met die van de formele taal.*

- *Een methode om neutrale applicaties om te zetten in agents, zodat een agent kan communiceren met een niet-agent door intenties toe te kennen.*

Het programma wat ik heb geschreven laat duidelijk zien dat er twee groepen zijn. De muizen en de snakes. De bewegingen van de snakes zijn duidelijk. Ze jagen op de muizen. Doordat de snakes door hebben wanneer er een muis in de buurt is en dan de achtervolging inzet, is er interactie tussen de twee groepen.

2.1
Deze formule laat zien dat state0 element is van iedere set die zal volgen. (dus set (I0) heeft als vervolgset (I0, I1) daarna (I0, I1, I2) enzovoorts) de set (I0) geeft weer welke informatie we hebben aan het begin van de simulatie. In het geval van mijn programma dat er 0 muizen zijn opgegeten.

2.2
Deze formule zegt dat wanneer hij iets ziet hij er een waarneming van maakt. Zoals de snake vooruitgaat en merkt dat er een muis voor zich staat.

2.3
Deze formule zegt dat wanneer er een muis wordt waargenomen de functie eat wordt aangeroepen in mijn code. En de muis verdwijnt.

2.4
I staat voor een snake en P voor of een muis op dezelfde plek zit als de snake of niet, hierin zal er meerdere acties geselecteerd kunnen worden. (bijvoorbeeld naast de actie wanneer er een mouse op dezelfde eet hij hem op er ook een actie wanneer hij een olifant ziet de snake een stap terug doet.) wanneer er geen actie geselecteerd word omdat hij niks ziet gaat hij weer terug naar het percieve onderdeel en begint de cyclus opnieuw.

3.1 Accessible vs inaccessible
Mijn Programma is accessible. Dat wil zeggen dat de agents makkelijk informatie kunnen krijgen van de environment. In het geval van mijn programma krijgt de snake een input vanuit de environment op het moment dat hij een muis voor zijn bek heeft of als een muis in de buurt is.

3.2 Deterministic vs non_deterministic
Mijn programma is deterministic. Dit houdt in dat wanneer een actie wordt uitgevoerd het gevolg vast staat. Bij mijn code zal een muis opgegeten worden door een snake wanneer deze precies voor zijn bek beland. Is er geen muis dan gebeurt er niks. Zijn er een of meerdere muizen in de buurt gaat hij altijd voor een van die muizen.

3.3 Episodic vs non-episodic
Mijn programma is episodic. Dat betekend dat een actie die in een episode wordt gemaakt niet de toekomstige actie bepaald. Wanneer een muis opgegeten wordt veranderd er eigenlijk niks in het environment wat invloed heeft op het gedrag van de muis of snake. Echter zal de snake op een gegeven moment geen eat action meer kunnen uitvoeren. De muizen raken namelijk op.

3.4 Static vs dynamic
Mijn programma is static. Wat betekend dat de environment alleen wordt beïnvloed door de acties van een agent. Je ziet bij de simulatie dat er alleen dingen gebeuren die worden veroorzaakt door agents. Bijvoorbeeld de snakes die eten.

3.5 Discrete vs continuous
Mijn programma is Discrete. Dit verteld ons dat er een vooraf een aantal acties of ontvangmogelijkheden zijn vastgesteld. De snakes kunnen niet meer muizen opeten dan dat er gecreëerd worden.

4

Een programma dat continious, dynamic en non-episodic

Wanneer we een programma hebben waarin agents elkaar geld geven wanneer een andere agents minder geld heeft en krijgen geld van agents die meer geld hebben. Dit gebeurt dan alleen als de andere agent binnen een straal van 1 eenheid staat. Er worden ook random briefjes geld gecreëerd in de environment welke opgepakt wordt door een agent wanneer deze erop komt te staan.