Opdracht 3

Intro to agents and agent-tools
Individual assessment
Sinem Ertem

1. Tutorial NetLogo

Om NetLogo beter te begrijpen heb ik de gehele tutorial gevolgd van *Complexity Explorer* [1] over Agent-Based Modelling op Youtube.

1.1 Conway's Game of Life

Game of Life is gebaseerd op een cellulaire automaat wat uit de automatentheorie komt. Het is een model wat bestaat uit een één of meer dimensionaal raster van cellen met elk een eindig aantal toestanden. Voor elk van de cellen zijn er regels die vertellen wat er moet gebeuren voor de volgende toestanden, als we kijken naar *Game of Life* bijvoorbeeld. Als de cel wit is, dus levend, en de cel heeft 3 levende buren dan gaat de cel dood.

Figuur 1: cel X is dood en heeft 3 levende buren aangegeven met 0. De cel komt tot leven in de tweede raster.

0		
0	X	0

0		
0	0	0

1.1.1 Regels

- Als een levende cel door 2 of 3 levende buurcellen omgeven wordt, blijft deze ook levend door balans.
- Als een levende cel meer dan 3 buurcellen omgeven wordt, gaat de cel dood door overpopulatie.
- Een dode cel komt tot leven als het omgeven wordt door 3 levende cellen zoals in *figuur 1* door voortplanting.
- Een levende cel gaat dood als er minder dan twee buren niet levend zijn door onderpopulatie.

1.2 Agent-oriented

ABM (Agent-based modelling) is een techniek om complexere systemen te modelleren om zo het gedrag van deze systemen beter te begrijpen. Acties en interacties worden op zo'n manier bestudeerd.

Agent-oriented programming is een programmeer model wat meer een sociale kijk heeft op berekeningen die worden gemaakt. Zo zijn de belangrijkste eigenschappen:

- Vermogens (dingen die de cellen kunnen doen)
- Overtuigingen (over de wereld, henzelf en andere cellen)
- Verplichtingen (de cellen moeten zich houden aan de regels)

De simpele regels die vastgesteld zijn voor de cellen van *Conway's Game of Life* zorgen eigenlijk voor complexe patronen (de verplaatsing van de cellen door interactie tussen de cellen). Het schijnt dat ze zo hun eigen leven leiden. Wezens die hun huidige staat updaten aan de hand van lokale informatie. Dit heeft dus veel weg van Agent georiënteerd programmeren.

[1] Bron: https://www.youtube.com/watch?v=YOGKSL7GuOE&list=PLF0b3ThojznRKYcrw8moYMUUJK2Ra8Hwl

1.3 Voor- en nadelen

De voordelen van NetLogo liggen vooral bij het feit dat het een tool is wat je snel op een gemakkelijke manier kunt leren. De tool is ook bedoeld voor agent-based simulaties en is daarom ook geschikt voor dit kleine project. De gebruiksvriendelijke interface zorgt ervoor dat je als beginner snel bij kunt leren. De 'Models Library'[2] bevat veel simulaties die al gemaakt zijn door gebruikers die je gemakkelijk kunt inladen en runnen. Er is een aparte kolom voor interface waarin de simulatie zichtbaar is, info, waarin er verdere uitleg staat over de code zelf en als laatst een kolom voor de code waarin je de code kunt schrijven.

Mesa heeft soortgelijke voordelen als we kijken naar het gebruiksvriendelijke. Het biedt visualisatie tools die je kunt aanpassen naar je eigen wensen, Unity gaat een stap verder in 3D simulaties, wat het toch wat moeilijker maakt qua omgang maar voor grote simulaties is dit wel voordelig.

Omdat de voordelen van NetLogo ook meteen minimalistisch zijn is een nadeel dat je complexere simulaties niet kunt nabootsen. Zo kun je bijvoorbeeld niet communiceren met databases, mocht je deze willen implementeren in de simulatie. Dan kun je beter gaan voor bijvoorbeeld de framework van Mesa die dit beter ondersteunt.

Verder maakt NetLogo gebruik van een eigen taal om scripts mee te schrijven. Het oogt erg simplistisch en is ook redelijk goed te begrijpen maar neemt nog steeds wat tijd in beslag wil je een goed script gaan schrijven. Met Python (Mesa) gaat dit bijvoorbeeld veel gemakkelijker, maar t.o.v. C# (Unity) is het wel 'makkelijker'. De library van Python is zo uitgebreid dat je voor moeilijke simulaties met de juiste library al heel ver komt. Dit mis ik toch met NetLogo.

[2] Bron: https://ccl.northwestern.edu/netlogo/models/

2. Beschrijvingen van functies van een abstracte agent

In dit hoofdstuk wordt er verder ingegaan op de simulatie *Conway's Game of Life*. De simulatie wordt beschreven aan de hand van functies die wat vertellen over een abstracte agent.

In een agent-based model hebben de agents entiteiten. Dit betekent dat die agents een state hebben. In *Conway's Game of Life* is vertelt de state iets over of ze levend zijn (witte cel) of dood (rode cel). Zo krijgen de agents ook informatie over andere agents, de buren. Elk cel is namelijk omgeven door 8 buren en van deze 8 buren krijgen ze dus informatie.

2.1 Functie 'See' en 'Perceive'

De *'See'* (S) functie van de agent is het vermogen om zijn omgeving te observeren. De uitkomst van de *'See'* functie is vervolgens hetgeen wat is waargenomen, namelijk *'Percept'* (P):

See: S → P

Bv: de dode cel kijkt of hij levende buren heeft.

2.2 Functie 'Update'

Gebasseerd op wat de agent heeft waargenomen, verandert de *'internal state'*. Zijn huidige state wordt dan geüpdatet en de cel krijgt een nieuwe state.

Update: I x P → I

Bv: de dode cel heeft waargenomen dat het 3 levende buren heeft en komt tot leven

2.3 Functie *'Act'*

Gezien de waarneming van de cel wordt er voor een geschikte actie gekozen.
I is hierin de '*internal state'* en A is de actie zodat:

Act: I → A

3. Beschrijvingen van de omgeving

In dit hoofdstuk zal de omgeving van *Conway's Game of Life* worden beschreven aan de hand van dichotomie zoals staat vermeld in het verslag van Woolridge & Jennings[3] die op Canvas staat.

3.1 Accessible vs. inaccessible

In een 'accessible' omgeving krijgt de agent complete en accurate informatie over de staat van de omgeving, deze wordt dan up to date gehouden. Elk levende en dode cel zou dan op de hoogte zijn van de staat van alle andere cellen in de wereld. Dit is niet helemaal het geval want het ontvangt alleen informatie van de buren, en dat zijn 8 cellen.

3.2 Deterministic vs. non-Deterministic

In een deterministische omgeving heb je de zekerheid dat de uitkomst altijd hetzelfde zal zijn. Als de beginvoorwaarden gelijk zijn, dan zal de uitkomst ook altijd gelijk zijn. Omdat de regels voor de cellen ook altijd hetzelfde zijn kun je enigszins van determinisme spreken. Als de setup voor de cellen altijd hetzelfde is, zal je ook altijd hetzelfde effect gaan krijgen. *Game of Life* is immers een zero-player game en vraagt verder om geen input die de uitkomst kan beïnvloeden.

3.3 Episodic vs. non-episodic

We spreken bij *Conway's Game of Life* van een 'Episodic' omgeving omdat de huidige cel alleen kijkt naar de staat waarin het zich op dit moment bevindt. Er wordt niet gekeken naar vorige staten of naar toekomstige staten.

3.4 Static vs. Dynamic

Een statische omgeving wordt alleen beïnvloed door de acties van een agent, in een dynamische omgeving zijn dit meerdere factoren. We spreken daarom van een statische omgeving omdat er alleen maar cellen zijn en verder geen andere invloeden.

3.5 Discrete vs continuous

Als laatst hebben we nog een discrete omgeving waarin er beperkte acties en percepties zijn. *Conway's Game of Life* heeft in dit geval dan ook een discrete omgeving. Elk cel begint met een staat dat ze of levend zijn of dood. De regels zullen continu worden toegepast op de cellen voor het creëren van nieuwe generaties. Zoals genoemd in hoofdstuk 1.1.1[4] zijn er slechts 4 regels waaraan de cellen zich moeten houden en daarmee een beperkt aantal acties.

[3] Bron: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.2204&rep=rep1&type=pdf

[4] Hoofdstuk 1.1.1 Regels pag. 2.

4. Reflectie op de omgeving

Wanneer een omgeving complexer is zal het moeilijk zijn voor de agents om te bepalen welke acties ze moeten nemen. De meest complexe omgevingen zijn dan ook de omgevingen die 'inaccesible', 'non-deterministic', 'continuous', 'non-episodic' en 'dynamic' zijn. Ik zal daarom voor 3 verschillende dichotomieën het tegenovergestelde proberen uit te leggen waarom dit wel of niet gunstig zal zijn voor de simulatie.

4.1 Inaccessible

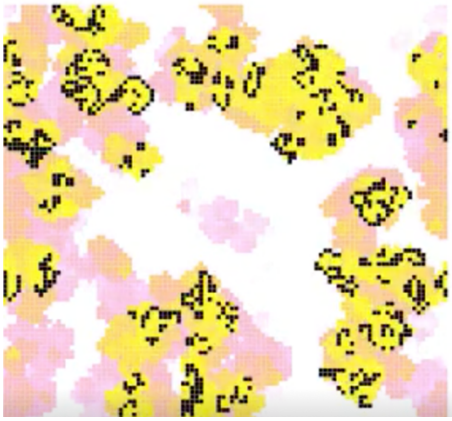
Als de cellen in *Conway's Game of Life* geen accurate en incomplete informatie ontvangen zal het ook moeilijk zijn om er een volgende generatie uit te creëren. De cel moet wel weten of zijn 8 buren levend/dood zijn om zo voor zichzelf te bepalen wat voor actie het moet nemen. Een 'inaccessible' omgeving zal dus ongunstig zijn.

4.2 Non-deterministic

Conway's Game of Life heeft van zichzelf deterministische regels. Als we stochastische regels toepassen hebben we te maken met kansen dat een bepaalde cel tot leven komt of doodgaat:

- 100% van de cellen gaan dood
- 99% van de cellen gaan dood
- 1% gaat dood, 1% komt tot leven
- 99.9% gaat dood, 0,1% komt tot leven
- 99.99% gaat dood, 0.01% komt tot leven

Figuur 2: Stochastische versie van GOL



Figuur 3: Deterministische versie van GOL

Een duidelijk verschil zijn de patronen[5] die bij een deterministisch proces vastgesteld zijn en bij een stochastisch proces niet voorkomen zoals:

- Oscillators
- Gliders
- Still lifes

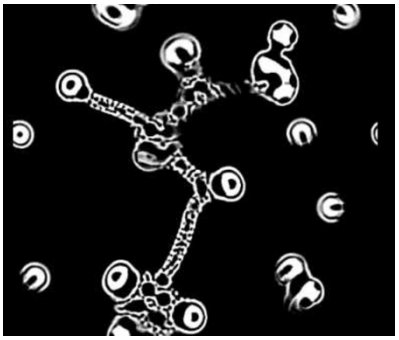
Als je de echte '*Conway's Game of Life'* wilt behouden is het niet gunstig om niet-deterministische elementen toe te passen omdat je de bovengenoemde patronen mist, maar voor een eigen variant maakt het vrijwel niks uit omdat er nog steeds generaties zullen ontstaan.

[5] Examples of patterns: https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

4.3 Continuous

Als we de discrete grid van *Conway's Game of Life* vervangen door een continue grid is de belangrijkste verandering dat het floating-points gebruikt in plaats van integers. De simpele regels die zijn vastgesteld zorgen nu voor complexere uitkomsten omdat de parameters 'real numbers' zijn en het gedrag van de objecten verandert. Maar niet heel veel.

Figuur: 4: Continue Conway's Game of Life



Wat er gebeurt in figuur 4 is dat de cel cirkelvormig wordt en de dichtstbijzijnde buren vestigen zich als een ring eromheen.
Het is gunstiger om het discreet te behouden omdat je toch ziet dat het complexer wordt en dat is niet nodig voor een simpele GOL.