Opdracht 1
Agent based Simulation

Netlogo

1.1 Tool verschillen

Netlogo, maakt gebruikt van turtles, oftewel agents. Het heeft een duidelijke en simpele manier van werken. Zo heb je een interface (front-end), info (documentation/handleiding) en code (back-end) tab dat je goed onderscheid geeft.
Het is helaas wel erg oud en niet uitbreidbaar. De taal blijft wel erg simpel om vele projecten af te maken,
maar om het voor grote bedrijven professionieel te gebruiken heb ik mijn twijfels, omdat het namelijk vele simpele tasks ook juist heel slecht doet.
Maar sommige onnodige grote tasks juist wel goed doet.
Waarop het lijkt alsof Netlogo gemaakt is om specifiek te werk te gaan en iets minder vrij is om elk onderwerp goed te simuleren.
Daarnaast omdat netlogo een simpel machine systeem is, gaat het voor veel problemen zorgen als het naar een multi machine systeem gaat. (Bron:
https://www.researchgate.net/post/Which-is-the-best-agent-based-modelling-tool-Netlogo-or-RePast-or-What-do-you-recommend-to-me)

Mesa, maakt ook gebruik van agents.
Je code understanding moet wel op een bepaald level zitten wil je goede code kunnen schrijven in python met mesa.
Voor vrij simpele modellen moet je toch al aardig wat code schrijven om een model te visualizeren met JavaScript en Python.
Dus browser based vizualisation.
Het is wel van goeie kwaliteit en heel erg uitbreidbaar.

Over unity:
De learning curve is vrij hoog, opties eigenlijk te uitgebreidt.
Engine is meer gemaakt voor games en niet echt voor data visualisation, al helemaal als je het meerdere malen achter elkaar zou moeten doen.

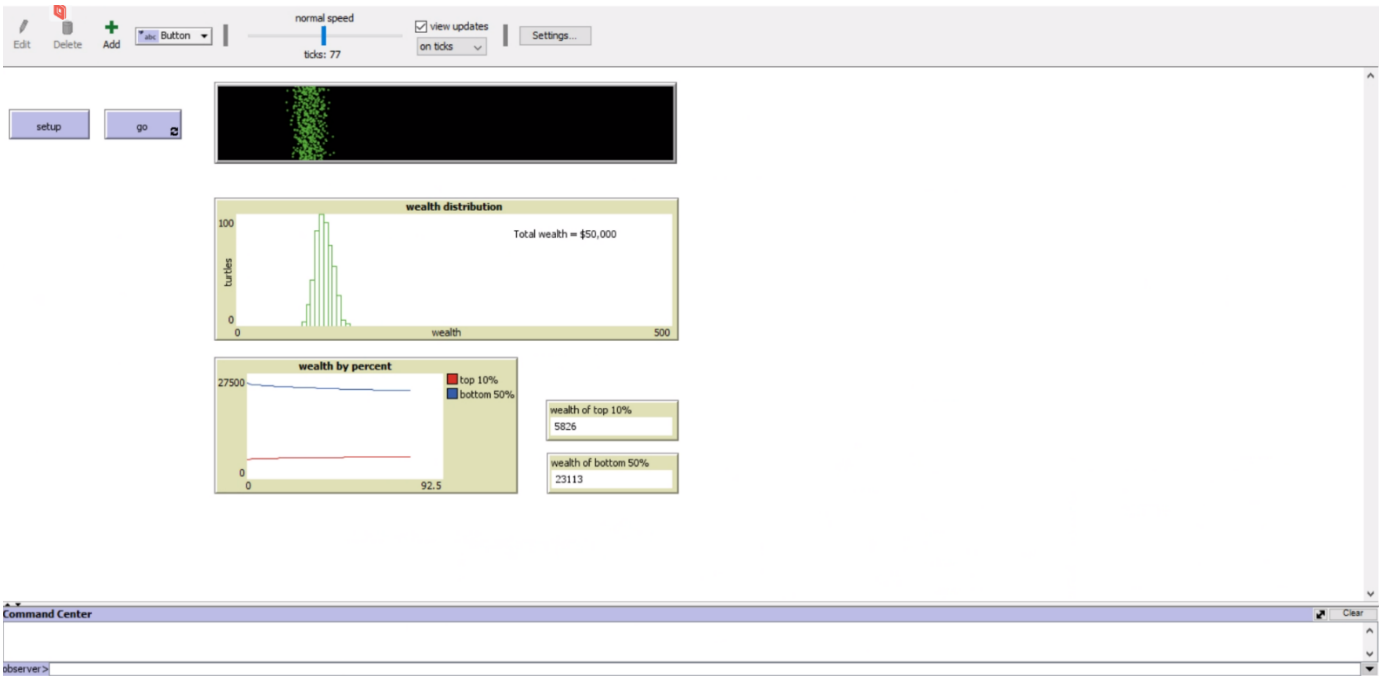
Model Simple Economy:

Samenvatting:

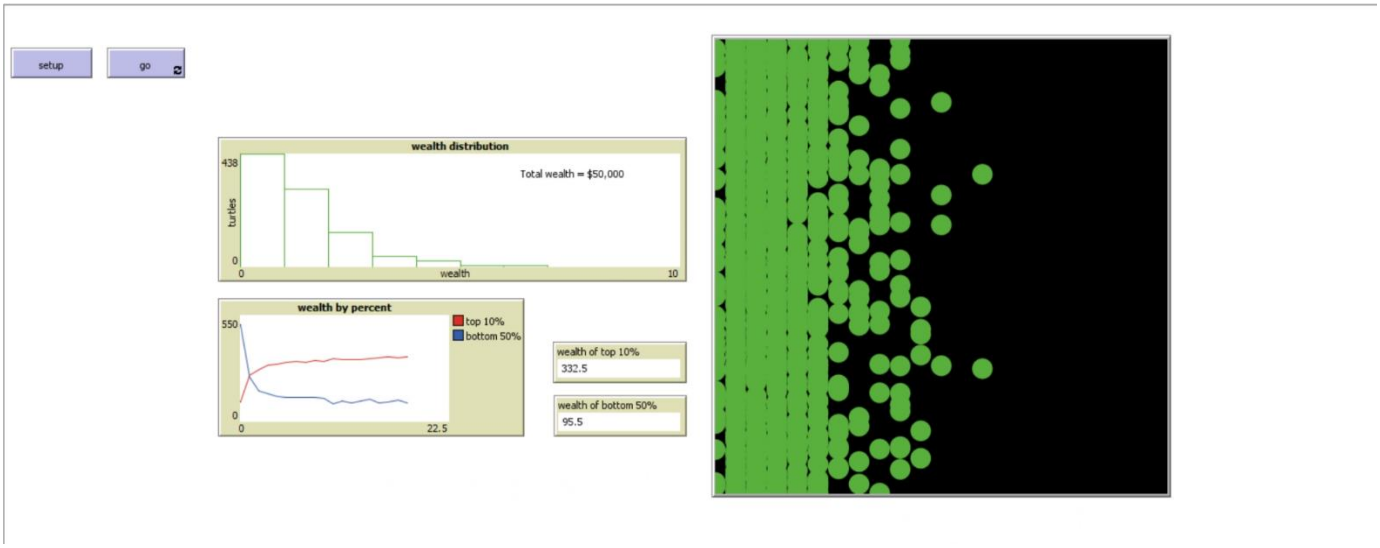
Er worden 500 verschillende kandidaten(turtles) gebruik die ieder 1 "geld" geeft aan een willekeurig persoon.
Ieder heeft ook 100 "geld" en kan dus 100 "geld" geven.

Ieder verliest ook het gegeven geld, waardoor ze blut kunnen worden. Dan kunnen ze geen geld geven en moeten ze wachten totdat ze weer geld krijgen.

Observatie:
Je ziet uit dat de kandidaten toch meer dan 100 "geld" kunnen krijgen, omdat er kans is dat 1 persoon 2 geld krijgt en dus 1 geld winst heeft. Daardoor kunnen er ook kandidaten, blut worden. Als je uiteindelijk dit een tijdje laat werken zie je dat de hoeveelheid aan kandidaten geld erg gespreid ligt. De lijngrafiek laat zien hoe dat te werk gaat met de top 10% en de bottom 50%. Je zou verwachten dat de 50% veel meer heeft, maar toch zie je bij de lijngrafiek dat er toch steeds meer rijkere kandidaten komen en armere mensen. Uiteindelijk zelf dat de top 10% meer geld heeft dan de bottom 50%



Extra element:
~Elk kandidaat begint met 1 "geld", de gift wordt verdeeld tussen 2 random personen (0.5 per persoon dus)
Bij de (kleiner gemaakte) grafiek van bottom 50% en top 10% is de top 10% meteen al meer dan de bottom 50% mede, omdat bij het begin vele kandidaten blut zijn, wat dus overeenkomt met de grafiek van wealth distribution. De grafieken van alle kandidaten als cirkels, wordt minder duidelijk dan bij 100 "geld" omdat alle kandidaten tussen kleinere waardes gaat liggen (0 en 10) ipv (0 en 500) waardoor die grafiek niet handig meer is.



1.2

Algemeen begrip per deelvraag omschrijving

1. De initiele staat LaTeX: i_0\in Ii 0 ∈ I, waarbij I alle mogelijk interne staten zijn.
Betekent welke staten elk agent zich kan bevinden, dit kan handig zijn om te checken of het voor de agent wel nodig is om de See functie te laten runnen.
Zo kan bijvoorbeeld een blutte agent geen geld geven aan anderen en hoeft het dus niet naar andere agents te kijken

2. Een functie "See" of "Perceive", die een mapping maakt van elke staat in de omgeving tot een staat die de perceptie van de agent van de omgeving aangeeft.
Betekent wat elke agent ziet in zijn omgeving (S) voordat het zijn actie onderneemt, wat kan aangeven of hij de actie doet of niet. Deze waarneming zet hij om in de letter P (erceptie)

3. Een functie "Act" die een perceptie van de omgeving neemt en een toepasselijke actie kiest.
Betekent wat de agent met P (uitkomst van See) doet en de juiste actie doorloopt en dan naar de volgende staat gaat zoals:
S0 met A0 -> S1 en dan met A1 -> S2 etc.

4. Een functie "Update" die een staat I neemt (soms D genoemd) en perceptie P, en een nieuwe staat I oplevert Betekent wat de agent na een actie onderneemt. Stopt het? Verandert het een waarde (aka update)? Of gaat het met/zonder aanpassing verder?

Deze begrippen worden in 2 simulaties uitgelegd. Beide simulaties hebben ieder een andere structuur, waarbij bij economy project de act functie belangrijk is en bij het fire project de see functie. Waardoor ze beide erg van elkaar verschillen

Demo Economy Project:
1. Initiële staat: Bestaat uit wealth en agents. Elke agent kan wel of geen wealth hebben. Een agent zonder wealth gaat niet de See functie gebruiken

2. See Functie: Elke agent kan elke andere agents zien, maar kan maar naar 1 willekeurige agent iets geven. Elke agent kijkt of het nog wealth heeft en zo ja dan hebben ze geld en zitten in een "have wealth" staat en kan het een actie doen.

3. Act Functie: Bij de actie krijgen ze minder wealth en geeft deze wealth aan een willekeurige andere agent, waarbij het niet uitmaakt welke staat de andere agent is (wealth of niet).

4. Update Functie: Deze acties en perceives wordt bij elke agent gedaan. De wealth wordt dan per agent geupdate en opnieuw gecheckt bij de perceive staat. En kijkt het vervolgens of het blut is of niet. Deze update functie blijft runnen totdat het crashed(aan aantallen) of handmatig gestopt wordt

Demo fire project:
1. Initiële staat: Bestaat uit greentrees(unburned) en burningtrees(burning) en burnedtrees. Elke tree kan dus verbrand of niet zijn. Een tree dat groen is gaat niet de See functie gebruiken

2. See Functie: Elke tree kan alleen zijn naasten buurman(nen) zien. Wanneer een tree burning is, kijkt het naar zijn buren of ze groen zijn, wat betekent nog niet burned. Als dat het geval is dan doet het de actie functie

3. Act Functie: Als de burning tree een buurman heeft die nog groen is, maakt het zijn buurman(nen) ook burning. Na deze burning state wordt de tree burned en zit het dus in de "burned" state en wordt dus zwart. In deze state gaat het niet meer kijken naar de buren.

4. Update Functie: De volgende run gaat het systeem kijken naar de burning (state) trees en laat het ook die zijn buurmannen aansteken, waarna deze bomen in de burned state komen en niet meer kunnen aansteken. Dit blijft doorgaan totdat elke boom unburned(groen) of burned is (geen elke tree burning).

1.3 Dichotomies van de Omgeving Simple Economy:
Beschikbaar vs ontoegankelijk,
De agenten in simple economy geven en nemen geld. Ze weten daarbij niet hoeveel geld de ander heeft, alleen wat hijzelf heeft. Hij weet niks van de omgeving dus zijn de agenten ontoegankelijk

Deterministisch vs niet-deterministisch,
De agenten geven willekeurige mensen geld, wat betekent dat elke agent elke keer een andere hoeveelheid aan geld/wealth heeft, waarbij de uitkomst nooit voorspelbaar is en is dus de simulatie niet-deterministisch
Episodic vs non-episodic,
De agenten geven geld aan willekeurige mensen. Dat betekent niet altijd dat ze ook iets terugkrijgen, dus is namelijk compleet willekeurig. De willekeurigheid bepaalt het systeem en niet de agents. Dus bepalen de agenten waar hun geld/wealth naar toe gaat? Nee, wat betekent dat het episodic is.

Statisch vs dynamisch,
De omgeving wordt bepaald door de agenten, maar deze agenten besturen zichzelf niet, maar worden bestuurd door de randomness. Dit zorgt ervoor welke agenten rijker worden dan anderen. Dus zijn er andere systemen die de agenten behelpen in het systeem? Ja, dus dynamisch

Discreet vs continu,
De omgeving laat door de randomness, het geld aan willekeurige mensen geven. De enige manier dat het systeem deze actie niet kan ondernemen is, wanneer iedereen blut is. Tijdens deze actie gaat er geen geld verloren en is er altijd wel een persoon met geld die het kan doorgeven.
Dus is het systeem oneindig of zijn er zoveel regels dat het net zoals een schaakspel uiteindelijk stopt?
Oneindig, Simple Economy project is continu

1.4 Tegenovergestelde Dichotomies
Een voorbeeldsysteem dat Niet-episodisch is in plaats van episodisch:
Verzekeringssysteem, elk persoon kan een object verzekeren als het kapot/niet werkend wordt. Als dat gebeurt dat wordt een groot hoeveelheid
van de waarde terugbetaalt. In ruil daarvoor betaal je een bepaald bedrag per maand/jaar. Dit moet je natuurlijk niet misbruiken. Je kan namelijk het object
express telkens laten vallen en dan geld terugkrijgen. Verzekeringen hebben daarom voor elk persoon een zogenaamde "reputatie". Dit om "slechte" personen voor
verzekeringen snel van te voren zien aankomen en hun bedrag per maand/jaar te verhogen.

Een voorbeeldsysteem dat Discreet is in plaats van continu:
Schaakspel, elk stuk van een schaakspel heeft een bepaalde patroon dat het beweegt. Naarmate het spel, verminderen
de hoeveelheid van deze stukken en is de bedoeling dat iemand wint en verliest. Het is wel mogelijk om oneindig door te gaan
als er bepaalde stukken over zijn, maar dat zou officieel niet voldoende zijn om het spel te eindigen.

Een voorbeeldsysteem dat Deterministisch is in plaats van niet-deterministisch,
Reken/meetsysteem, een som dat uitgerekend moet worden, maakt niet gebruik van willekeurigheid. Een optel som zoals 1+1 maakt altijd 2 (tenzij binair).
Als dit systeem wel niet-deterministisch zou zijn, dan zouden alle berekeningen onjuist zijn en zou de wereld in elkaar storten. Want als
een goudvis groter is dan een flatgebouw dan kan je beide objecten niet met andere objecten vergelijken omdat het onjuist is.