

Scripting Turtles

in netlogo



Jelle Stiesri

26-11-2020

1. Taal – Pro&Con

Het is gelukt om de tutorial te volgen, Ik heb hier ook nog wat dingen aan toegevoegd als stelen, salaris, en alleen maar geld uitwisselen met agents die ongeveer even rijk zijn. Netlogo is puur en alleen bedoeld om agent based simulations mee te maken. Dit is een groot verschil ten opzichte van andere talen/tools waar dit eventueel in zou kunnen. Zo kan unity voor veel meer dan simulaties gebruikt worden en is Python natuurlijk ook voor ontzettend veel bruikbaar. Dit maakt Netlogo heel fijn voor ABS, omdat er niet allemaal onnodige functies bij komen kijken die je niet nodig gaat hebben voor je simulatie.

Pro:

- Makkelijk te leren
- Niet veel computerkracht nodig
- Duidelijke resultaten
- Makkelijk interactief maken
- Makkelijk opslaan & delen met andere onderzoekers

Con:

- Beperkte visualisatie (in de 2d tool die wij gebruiken)
- Er zijn wel mogelijkheden tot het koppelen met andere talen maar dit zal minder werken dan het direct in die taal maken (mesa bijvoorbeeld)
- Hulp zoeken moet vooral in de eigen gebruiksaanwijzing omdat er niet mega veel forum berichten over zijn

Vergelijking met de andere tools: (gebaseerd op uitgewisselde ervaring van groepje)

Unity:

- Mooi om te gebruiken bij ingewikkelde en grote simulaties
- Goed voor Game(simulaties)
- Pittig om zomaar mee te beginnen en te snappen

Mesa:

- Als je python kan hoef je niet een hele nieuwe programmeertaal te leren.
- Makkelijk samen te gebruiken met andere python modules als pandas en numpy
- Alleen een 2D grid, minder makkelijk en mooi te visualiseren.

Netlogo-Unity:

- Unity is een mooie tool om verder in te gaan op mooiere / uitgebreide visualisatie
- Voor simpele simulaties veel eerder aan Netlogo beginnen. Makkelijk in elkaar te draaien en bekijken.
- Unity is een stuk moeilijker om te leren en mooi te maken

Netlogo-Mesa:

- Netlogo is een hele taal op zichzelf, python zullen veel gebruikers al kennen/beheersen
- Netlogo is sneller en duidelijker in het visualiseren, je ziet meteen resultaat en hoeft niet moeilijk te doen met het direct bijstellen van instellingen.
- Je zult wanneer je mesa goed beheerst er waarschijnlijk mooiere simulaties mee kunnen maken.
- Je kan met python heel veel extra dingen toevoegen, berekenen & uitbreiden ten opzichte van Netlogo.

2. Omschrijving

De simulatie bestaat alleen uit agents(turtles in netlogo), deze agents hebben elk 2 variabelen: `Balance` en `Status`. Balance staat voor het huidige saldo(\$) van de agent en Status geeft aan of de agent word gezien als 'Poor', 'Neutral' of 'Rich'. De initiele staat van een agent bestaat uit een balance van 100\$ en de status 'Neutral'.

Een agent kan op elk moment met de **see** functie alle agents en hun variabelen waarnemen. De agent kunnen gebaseerd op deze waarnemingen geld uitwisselen of proberen te stelen van rijke agents.

Dit gebeurt in de **Act** functie die zich onderverdeelt in de `steal` en `transact` functie. Elke agent met geld zal elke **tick** de transact functie uitvoeren, hier kies hij een willekeurige andere agent om geld aan te geven(als `spend_in_group` aan staat kies hij een turtle die op een bepaald % dichtbij staat) . De steal functie is gebaseerd op kans(kans of deze word aangeroepen door een arme agent, kans op meer of minder geld, kans dat het mislukt) en maakt gebruik van de status van de agents.

Om de 1000 **ticks** krijgt een agent salaris, de hoogte hiervan is gebaseerd op hun status.

De **Update** gebeurt elke tick, dus de `transact` & `steal` functies worden elke tick aangeroepen(steal word niet altijd uitgevoerd) En nadat deze functies zijn uitgevoerd is de balance aangepast en word er **opnieuw** bepaald welke status een agent moet krijgen.

3. Omgeving

Accessible: De omgeving & agents zijn simpel. Elke agent heeft een saldo en een status. En elke agent kan alle andere agents zien + hun variabelen.

Non-Deterministisch: Het uitwisselen van geld gaat compleet random, elke agent kies willekeurig een andere agent om geld aan te geven. Maar als een agent geen geld meer heeft kan hij dit ook niet meer uitgeven. Ook gaat het stelen compleet random, hierbij zijn een paar kansen mogelijk:

- 1 op 5000 dat een arme turtle probeert te stelen van een rijke.
- 1 op 10 dat het stelen mislukt en de turtle dood gaat.
- Een arme turtle steelt altijd van een willekeurige rijke turtle
- De turtle kan tussen de 0 en 50% van wat de gebruiker als rijk gedefinieert heeft stelen.

Al deze kansen zorgen ervoor dat er niet te voorspellen valt wat er met een specifieke agent gebeurt en zal elke simulatie anders uitpakken.

Non-episodic: Elke tick heeft invloed op de volgende. Omdat er niet te zeggen is hoeveel geld een agent elke tick krijgt en aan wie hij het geeft maakt het dus erg veel uit welke actie de agents uitvoeren. Wanneer een agent bijvoorbeeld geen geld meer heeft kan hij het ook niet meer uitgeven. Elke actie heeft invloed.

Dynamic: De omgeving is zoals eerder gezegd zeer simpel, er zijn alleen agents aanwezig die heen en weer bewegen gebaseerd op hun balance. Echter gebeuren er ook dingen met een agent die niet gebaseerd zijn op zijn eigen acties, zoals het krijgen van geld van andere agents, of bestolen worden. Hierdoor word de omgeving dynamic.

Discrete: De omgeving is gesloten. Wanneer de balance van een agent groter word dan de grid word zijn locatie niet meer geupdate (zodat deze agent dus in de omgeving blijft). Er kunnen geen 'onverwachte' dingen gebeuren, alle mogelijke functies zijn gedefinieert en een agent zal altijd de actie uitvoeren welke opgegeven word. Elke agent blijft dus binnen de omgeving en kan geen balance lager dan 0 bereiken.

4. Andersom!

- Inaccessible
- Static
- Continuous

Voorbeeld: In deze simulatie zijn een paar dingen anders. Elke agent kan alleen maar de agents in zijn eigen status (poor, neutral, rich) zien (**Inaccessible**). De omgeving heeft geen invloed meer op de agents, de agent kan geen geld meer aan andere agents geven maar kan wel geld geven aan de overheid (balasting) (**Static**). Ook zou deze bijvoorbeeld kunnen gaan gokken, het winnen/verliezen van geld zal dan een gevolg zijn van de acties van die agent. Hierdoor zou de agent ook in de schulden kunnen raken. De grid word oneindig, een agent kan dus in de schulden raken en oneindig veel geld verliezen, of juist heel veel verdienen. (**Continuous**) De simulatie blijft non-deterministic, gokken valt niet te voorspellen. En tot slot blijft de simulatie ook Non-episodic, om dezelfde reden waarom deze het überhaupt al was