Scripting Turtles

Jelle Stiesri

26-11-2020

1. Pro/Cons

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.

Netlogo-Mesa:
- Netlogo is een hele taal opzichzelf, 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.

2. Omschrijving

De simulatie bestaat alleen uit agents(turtles in netlogo), deze agents hebben elk 2 variabelen: ‘Balance’ & ‘Status’. Balance staat voor het huidige saldo($) van de agent en Status geeft aan of de agent word gezien als ‘Poor’, ‘Neutral’ of ‘Rich’. Elke agent begint met een balance van 100$ en in de status Neutral. De omgeving bestaat alleen uit agents die uit elkaar staan gebaseerd op hun balance. Elke agent kan op elk moment elke andere agent **zien**, hierdoor kan deze een willekeurige agent uitzoeken om geld aan te geven.

In de `go` functie worden bepaalde acties aangeroepen gebaseerd op de gebruikersinvoer, mocht de gebruiker bijvoorbeeld `stealing` uitzetten, dan zal de functie `steal` nog aangeroepen worden.

De functie `transact` zorgt ervoor dat een agent geld geeft aan een willekeurige andere agent. Als de gebruiker `spend_in_group` aan heeft gezet kijkt de agent alleen naar een bepaald percentage (dit percentage word ook door de gebruiker bepaald) van de agents die het meest dichtbij zijn. Deze functie **kijkt** dus alleen naar xcor van elke agent om te bepalen welke het dichtstbij staan.

De functie `salarys` is een soort extensie van transact. Deze geeft elke agent een salaris, de hoogte hiervan is gebaseerd op hun status. Die status is dus het enige waar deze functie naar **kijkt**.

De laatste ‘normale’ functie is `steal`, Als dit aan staat is er elke tick voor elke ‘poor’ agent een kans dat hij probeert te stelen van een ‘rich’ agent. Er is een kans dat dit mislukt, mocht dat zo zijn dan gaat de agent dood. Als het stelen wel lukt krijgt de arme agent een deel van het geld van een rijke agent.

Bij elke tick word het hele programma **geupdate**. Elke agent loopt alle functies langs (sowieso transact, en ook salarys & steal gebaseerd op de gebruikersinput). Wanneer de balance is geüpdatet bepaald hij opnieuw de status van een agent. Mocht deze over de grens van `rich` heen zijn, dan word de status daar naartoe veranderd. Hier hoort ook meteen een juiste kleur in de wereld bij. Elke functie **kijkt** naar andere eigenschappen van een agent en heeft hier ook andere **handelingen** bij.

3. Omgeving

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

Non-deterministic: 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 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 rede waarom deze het uberhaubt al was