Agents and Agents- tooling

Amaryllis Lee
Studentennummer : 1735290
Gekozen tutorial : Behave in Unity

Vraag 1
Volg de tutorial en schrijf daarna in één paragraaf om wat deze tool anders maakt dan andere programmeertalen, wat zijn de voor- en nadelen? Leg ook uit waarom je programma wel of niet agent-based is.
In deze tutorial wordt de *sensory systems* / zintuigen van een Non-Player Character .
Deze tool maakt gebruikt van Unity om de ontwerp van de simulatie (*scene)* te realiseren. Bovendien wordt er gebruikt gemaakt van C# om onze tank en AI te laten bewegen en andere acties uitvoeren.

Unity is een tool op een hoger niveau waar gebruikers een simulatie kunnen implementeren zonder de game-objecten en sensoren, enz. Helemaal opnieuw te hoeven maken (ex Game-objecten, raycast.) Verder houdt de tool de tijd bij en krijg je een duidelijke visualisatie van de simulatie

Deze tool is *agent-based*, omdat de game-objecten (Tank en AI) voeren acties uit op basis van een bepaalde perceptie.
De voordelen en nadelen van het gebruik van deze tool (Unity en C#):
Voordelen:
- U hoeft bepaalde objecten en functionaliteiten niet helemaal opnieuw te creëren
- Duidelijke visualisatie van je simulatie

Nadelen:
- Unity is een geschikt voor het bouwen van games, dus er zal meer code betrokken zijn bij het maken van de simulatie

Vraag 2
We definieren een staat en drie functies waarmee we een stateful agent abstract omschrijven:

1. **Een initiele staat i 0 ∈ I , waarbij *I* alle mogelijk interne staten zijn**
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. Dus: *See:* S → P , waar *S* de staat of serie van staten van de omgeving is en *P* de perceptie van die omgeving**
3. **Een functie "Act" die een perceptie van de omgeving neemt en een toepasselijke actie kiest. Dus *Act:* I → A , waar *I* een interne staat is en *A* een actie.**
4. **Een functie "Update" die een staat *I* neemt (soms *D* genoemd) en perceptie *P*, en een nieuwe staat *I* oplevert, dus *Update:* I × P → I**

Dit is een korte samenvatting van wat er staat in: "An introduction to Multi-Agent Systems" chapter 1.1 up to 1.4. Die paragrafen kun je hier vinden. Natuurlijk worden deze functies stukken complexer wanneer je agenten complexer worden. Beschrijf nu in je eigen woorden wat deze functies zijn in de context van de simulatie uit je tutorial.

I = internal states
I zal de *transform.positions* van onze Game-objecten zijn. Dit geeft ons waar onze Game-objecten zich in onze omgevingen bevinden en in welke richting (*target position*) het Game-object zal bewegen

I0 - initial state
De initiële staat is de positie waar de Game-objecten (AI en tank) zich aan het begin van de simulatie bevinden.

See : S(state(s) in the environment) -> perceive
See voor AI zal de AI zijn die de aspect typen (P) definieert wanneer de AI een Game Objects tegenkomt met de tag PLAYER of wanneer de collider van de AI wordt aangeraakt met de collider van een ander GameObject. (S)

Act : I -> A
Wanneer de AI een Game Objects tegenkomt met de tag PLAYER of wanneer de collider van de AI wordt aangeraakt met de collider van een ander GameObject en het aspect van dit andere Game-object ENEMY is, zal de AI een bericht (actie) afdrukken

Update: I x P -> I
P is wanneer de afstand tussen de positie van AI of Tank en de doelpositie (*target position)* groter is dan de afstand tolerantie (*DistanceTolerance)*. Op basis van deze perceptie en de interne toestand (*i)* zal het Game-object naar de volgende positie gaan.

Vraag 3
Beschrijf je opgeving op basis van de dichotomies die hier op pagina 6 beschreven staan, en licht toe (dus niet alleen termen opsommen)

1. **Accessible vs inaccessible**
   De omgeving is *accessible*, doordat de agents de positie en het aspect type van een ander Gameobject in de omgeving kunnen achterhalen.
2. **Deterministic vs non-Deterministic (Stochastic)**
   De omgeving is deterministisch. De agent kan een bericht afdrukken als het aspect type van het andere object niet gelijk is aan het aspect type van de agent. Bovendien kunnen de agenten alleen van positie veranderen als de afstand groter is dan de Afstand Tolerantie(*DistanceTolerance)*

3. **Episodic vs non-episodic (Sequential)**
   Deze omgeving is episodisch, omdat de actie van de agents is gebaseerd op de waarneming (*perception)* in de huidige staat in onze simulatie

4. **Static vs Dynamic**

5. **Discrete vs continuous**
   In deze omgeving kunnen onze agenten zich alleen verplaatsen als de afstand groter is dan de tolerantie. Bovendien zijn onze AI-agenten in staat om een bericht af te drukken wanneer het zich in het zichtbare bereik bevindt (*see*) of het aanraakt (collider wordt geactiveerd) met een ander GameObject dat het aspect type ENEMY heeft.
6. **Vraag 4**
   Bedenk een voorbeeld waarbij minimaal 3 dichotomies precies tegenovergesteld zijn en beargumenteer waarom dit wel of niet van belang is om je simulatie nuttig te maken.

Een voorbeeld zou zijn dat op basis van het aspecttype van een Game-object, het AI-personage verschillende acties kan uitvoeren (bericht afdrukken, kleur wijzigen, terugkeren naar de oorspronkelijke positie, enz.). Dit maakt de omgeving niet deterministisch.

Op basis van eerdere acties kunnen de agenten een andere speler ontwijken of specifieke acties ondernemen wanneer ze een vijand tegenkomen. Deze aanpassing zal de omgeving niet-episodisch maken.

De agenten kunnen de positie of een ander onderdeel van een ander object in de simulatie niet krijgen. Dit maakt onze simulatie ontoegankelijk. Onze agenten kunnen niet naar een bepaalde positie in onze omgeving gaan en kunnen het type object waarmee het wordt geconfronteerd, niet detecteren.

Het ophalen van de component van een ander object is essentieel voor deze simulatie om een bepaalde waarneming te krijgen (de aspect typen definiëren, de afstand verkrijgen) en acties te ondernemen (verplaatsen naar een andere positie / interne toestand wijzigen, bericht afdrukken).