Simulation Tooling Intro to agents

Student: Yannick van Diermen.

Opdracht 1

Ik heb gekozen voor de tool Unity. Unity is geen speciale programmeertaal of specifiek gemaakt voor simulaties, maar het is een 3d Game Engine die al veel handige functies heeft ingebouwd. Het maakt gebruik van een Entity Component System. Een entity is een object in de tool waar components aan vast hangen. Components kunnen scripts zijn of andere functionaliteit hebben. Die Entities kunnen we gebruiken om Agents mee te bouwen. Ook is het makkelijk om de Agents 'sensoren' te geven door middel van colliders en raycasts. Ten slotte kunnen we snel een pathfinder maken die een eindpunt probeert te bereiken met een ingebouwde navigatie functies.

De voordelen van Unity zijn: 3d, ingebouwde Physics Engine, renderer, scènes, navmesh.

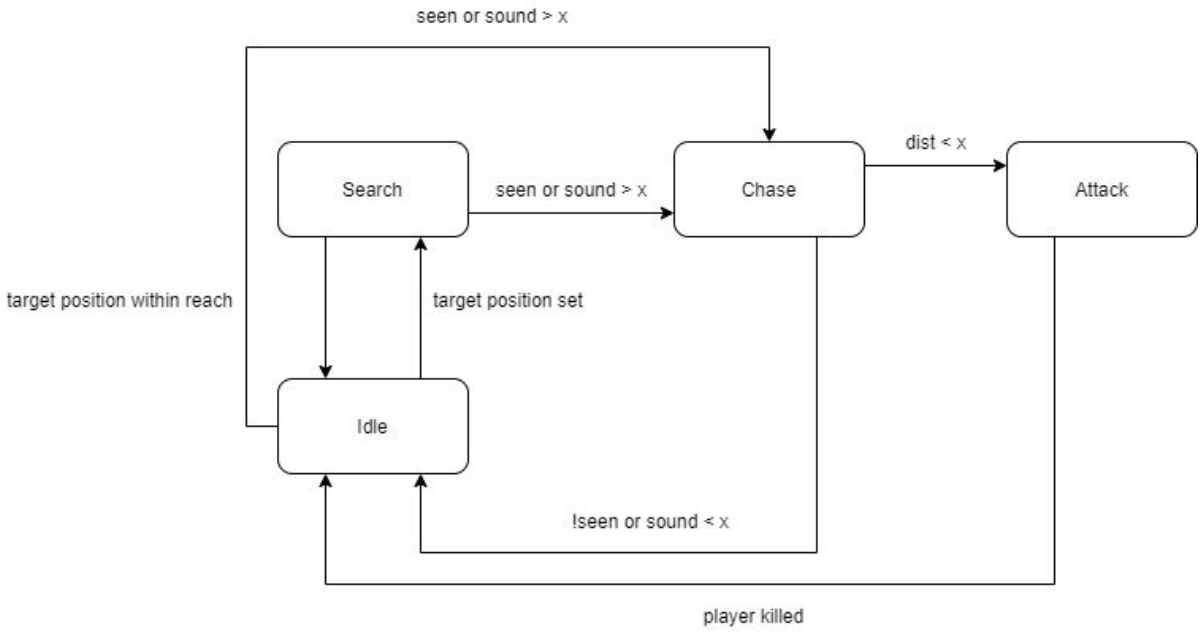
Het grootste nadeel is dat Unity geen main game loop heeft. Daarom is naar mijn mening data heel erg verspreid wat het moeilijk maakt om ècht objectgeorienteerd te programmeren.

Unity is wel een Agent based taal door de Entities en de individuele start en update loops bij elk script.

Opdracht 2

Ik ga voor deze opdracht de AI van Alien Isolation proberen na te maken. Of in ieder geval een versimpelde versie ervan. In volgende bron (https://becominghuman.ai/the-perfect-organism-d350c05d8960) wordt de werking en filosofie achter de AI uitgelegd. Dat heb ik gebruikt om een goed beeld te krijgen hoe de AI werkt.

Het gedrag van de AI wordt gedreven door een Finite State Machine. De FSM bepaalt wat de enemy op dit moment mag/moet doen. De senses die de agent heeft kunnen de interne variabelen (state) aanpassen.



In mijn code bestaat de "perceive" functie uit de volgende drie functies: look, listen en sense Player Position. De look functie gebruikt een raycast om naar voren en naar achteren te kijken in één straal. Van achteren kijken heeft een kleinere range dan naar voren kijken. De listen functie maakt gebruik van een sphere collider die detecteert of er een speler in de buurt is. Als er een speler aanwezig is, dan kijkt de agent of de speler aan het bewegen is. Aan de hand van de afstand tot de speler wordt er dan een geluidswaarde gegeven.

De "update" functie is een FSM die aan de hand van de huidige staat naar een nieuwe staat kan gaan. Misschien is dit in de code wel een beetje verwarrend, maar er is een state van de FSM en er is state in de vorm van variabelen. Hoe dan ook wordt er in de change State functie in mijn code aan de hand van in welke FSM state en variabelen een nieuwe FSM state gekozen. Bijvoorbeeld als de agent in de "search" state zit en de geluidswaarde die de listen functie update hoger is dan een bepaalde threshold, dan gaat de agent naar de "chase state". Het belangrijkste wat je hieruit kan halen is dat de variabelen die in de "perceive" functie worden veranderd niet altijd dezelfde invloed hebben op het gedrag van de agent, dit is volledig afhankelijk van de FSM.

De "act" functie heet in mijn code execute State en hoeft alleen naar de huidige state van de FSM te kijken en dan een (paar) functies uit te voeren.

Opdracht 3

Inaccessible, de agent heeft niet alle informatie tot zijn beschikking uit de simulatie.
non-Deterministic, de agent beweegt naar random coördinaten als hij aan het zoeken is.
Episodic, de agent hoeft niet na te denken over de toekomst.
Dynamic, de positie van de speler kan bewegen wat betekent dat de enviroment verandert door iets anders dan de agent.
Continuous, omdat de simulatie in theorie eindeloos lang kan draaien.

Opdracht 4

Het simuleren van een schaakspel is niet continuous, omdat er maar beperkte stappen mogelijk zijn. De simulatie is ook niet episodic omdat het wel degelijk moet nadenken over toekomstige stappen. Daarnaast is de simulatie accessible omdat het de hele staat in één keer ziet.