S006D – Datorspels AI

Laboration 1

Ludwig Lindmark

[ludlin-8@student.ltu.se](mailto:ludlin-8@student.ltu.se)

FSM – (INSERT GITHUB REPO)  
2020-02-21

Innehållsförteckning

Problemspecifikation

Uppgiften går ut på att programmera en FSM (Finite State Machine). FSM ska implementeras som en agent med ett vardagligt liv där den kan äta, dricka, sova och jobba. Agenten skall också ha ett socialt behov samt ha möjlighet till att vara social med andra agenter.

Kommunikationen mellan agenterna ska lösas med hjälp av ett meddelande system som ger agenterna funktionen att hantera samt skicka meddelanden till varandra vid behov.

Problemlösningen uppnår betyg 3 då det inte existerar en visuell implementation av agenternas liv.

Uppgiften ska implementeras i Python 3

Användarhandledning

FSM Ai laboration 1 finns tillgängligt på GitHub.

Programmet körs via main.py filen, det inte krävs någon användarinmatning för parametrar. Tiden i programmet påverkar hur effektiva agenternas handlingar är. Om användaren vill ändra tiden, ändra parametern för update i main.py

Algoritmbeskrivning

# Meddelandesystem

Meddelande funktionen i programmet skickar meddelanden till andra agenter eller till sig själv. Ett meddelande består av en sändare, mottagare, typen av meddelande samt en fördröjning. Fördröjda meddelanden lagras tills de skickas. Meddelanden utan fördröjning skickas direkt

Innan ett meddelande skickas lagras det efter fördröjningstid. Detta gör det enklare för systemet att hitta det meddelande som ska skickas.

Systembeskrivning

Alla tillstånd ärver från ett grundtillstånd som inkluderar alla metoder för en FSM. Alla agenter ärver från en basentitets klass som innehåller övergripande info t.ex entitetens ID inom systemet. Bilder på ett simpelt klassdiagram finns under bilagor.

# Klasser

Beskriver programmets klasser:

* Telegram - Telegram lagrar all data för ett meddelande, sändare, mottagare, meddelandetyp, fördröjning och extra information vid behov.
* MessageTypes – Beskriver olika typer av meddelanden som enum
* Locations – Beskriver agenternas plats i form av en enum
* BaseEntity – En abstraktklass som innehåller id för alla agenter.
* EntityManager – Entity manager uppdaterar alla agenter inom programmet. Entity manager ansvarar också för meddelandesystemet. Alla agenter lagras i en dictionary i entity manager. Innehåller metoder för att lägga till nya agenter.
* Agent – Agent ärver från klassen BaseEntity. Agent klassen används för att skapa en ny agent. Klassen innehåller den grundläggande strukturen för en FSM. Agent klassen har olika behov utifrån värden som hunger, törst, socialt behov och trötthet.
* State – En abstraktklass som alla andra tillstånd ärver från. Klassen innehåller 3 metoder för Enter, Exit och Execute.

# Tillstånd

Beskriver agenternas tillstånd:

* EatFood – Agenten äter mat och nollställer sin hunger. Detta tillstånd tillhör GlobalState.
* QuenchThirst – Agenten dricker vatten och nollställer sin törst. Detta tillstånd tillhör GlobalState.
* GoHomeAndSleep - Agenten går hem och skickar ett fördröjt meddelande till sig själv som agerar väckarklocka. Agenten sover och sänker sin trötthet tills agenten tar emot sitt fördröjda meddelande.
* GoToWorkAndLabour - Agenten går till sin arbetsplats där den skickar ett fördröjt meddelande till sig själv. Agenten arbetar och samlar pengar tills agenten tar emot sitt fördröjda meddelande.
* Leisure – Agenten har ett tillfälligt tillstånd efter sitt arbetspass där agenten frågar vid behov om dennes vänner är tillgängliga för att umgås.
* Social – Om Agentens vänner har tid för att umgås så socialiserar dem med varandra. Agentens sociala behov sänks.
* Shopping - Om Agentens vänner inte har tid eller något behov för att umgås så handlar går agenten och handlar istället.

Lösningens begränsningar

Lösningen har för tillfället hårdkodade värden för agenternas behov samt antalet agenter programmet ska generera.

Koden är inte optimerad för prestanda då det inte var en prioritering.

Diskussion

I början uppstod ett problem då meddelande systemet implementerades med min första lösning. Detta är på grund av dålig design och planering, vilket resulterade i mycket slösad tid och kod.

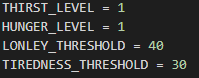
Det var svårt i början att förstå vad agenterna faktiskt gjorde. Det tog längre tid än förväntat med felsökning och problemlösning då majoriteten av tiden gick åt till att titta på agenternas beteende i konsolen.

Laborationen gav en tydlig bild av en grundläggande FSM. Det fanns flera moment och utvecklingsmöjligheter. Personligen tyckte jag att meddelandesystemet var ett av de mer lärorika momenten.

Kursmaterialet var till stor hjälp i början då detta är den första laborationen i kursen. Boken beskrev teorin väldigt utförligt med ett bra exempelscenario och kod.

Testkörningar

Vid vissa fall av extremt obalanserade inställningar av agenternas parametrar kommer resultera i att vissa tillstånd aldrig exekveras. Följande konfiguration resulterade i att agenten drack och åt istället för att vara social.



Bilagor

