Luleå Tekniska Universitet
Skellefteå

Laborationsrapport

S0006D, Datorspels AI

Laboration: Datorspels AI, Labb 1

Jonas Hansson

ojaaho-6@student.ltu.se

PythonAI_1, 2020 > Release

6/2/2020

Innehållsförteckning

Problemspecifikation

Skapandet av Artificiell Intelligence i en textbaserad miljö där kommunikation med andra agenter utbytes. Detta genom hjälp av State Machines och meddelande hantering för event och task prioritering. Dessutom ska agenterna balansera sina värden så de inte går sitt öde till mötes allt för fort, men samtidigt inte är odödliga i simuleringens värld.

Användarhandledning

För att kunna använda detta program krävs Python3 installerat på datorn. Starta simuleringen genom att dubbelklicka på "PythonAI_1.py". Denna fil hittar man i senaste GitHub commiten, i mappen 2020 och därefter mappen Release. Vid start öppnas en kommandotolk med instruktioner om att ange antal agenter som önskas simuleras, namngivning av agenterna, samt ange värde på uppdateringsfrekvensen för programmets exekevering. Därefter körs simulationen med konversationer, statistik och nuvarande stadie av agenternas liv tills dess att all agenter har mött sitt öde. Simuleringens egna tidsräkning skrivs sedan ut för att summera antalet månader, dagar, timmar och minuter de varit aktiva.

Algoritmbeskrivning

Inga speciella algortimer har använts i projektet. Dock används Finite-State Machine som är uppdelad över olika klasser av "States". Varje State har en Enter(), Execute() och Exit() funktion där varje State specifik kod angetts för att mutera variablers värden och ange byte av States vid specifika händelser och värden.

Systembeskrivning

PythonAI_1.py är den primära filen för simulerings exekevering. I denna fil sköts allt vid start samt uppdatering av agenterna, tidsräkningen samt kontroll av meddelande hanteringen efter ett meddelande har blivit skickat.

GameTime.py är där tidsräkningen beräknas. Denna läses in i **PythonAI_1.py** under simuleringens gång för att uppdatera tiden som ska visas i världen.

BaseGameEntity.py är grunden för alla olika entiteter som ska instansieras i världen. Denna ärvs av **Humanoid.py** för att ge den ett unikt entitets ID och namn, exempelvis "Humanoid_Name". För närvarande används det inte till så mycket mer men kan gynna framtida projekt genom dess mer exklusiva namn och ID kontra Humanoids.

Humanoid.py är filen där all struktur och alla beteenden för agenterna exsisterar. En agent är en instns av Humanoid som ärver av BaseGameEntity. I konstruktorn för Humanoid anges exempelvis namn, ID, nuvarande plats, samt statistik så som hunger, törst, energi, pengar, gång hastighet, med mera. Under simuleringens gång så kommer Humanoid instansen, vår agent, att hoppa mella de olika states som finns definerade för att simulera vardagliga aktiviteter och sinnestillstånd så som vi människor kan uppleva och delta i. Dessa states är som följande:

- **State_Sleep:** Används för att låta agenterna återhämta sig efter dagens spring genom att sova. Förekommer vanligtvis under nattens närvaro, 22:00 – 08:00. Under natten kan också agenterna få en otrevlig mardröm somstressar upp dem för att begränsa överflödig energi efter en lugn dag.
- **State_Eat:** Används för att öka agenternas hunger nivå i utbyte för en liten summa pengar. Ju lägre denna är ju hungrigare är agenten. När den 0 eller går under detta kommer agenten att avlida. Förekommer vid låga värden av hunger, förutsatt att agenten har ekonomi för detta.
- **State_Drink:** Används för att öka agenternas törst nivå i utbyte för en liten summa pengar. Ju lägre denna är ju törstigare är agenten. När den 0 eller går under detta kommer agenten att avlida. Förekommer vid låga värden av törst, förutsatt att agenten har ekonomi för detta.
- **State_WorkOffice:** Används för att öka agenternas nuvarande summa av pengar, samt sänka hunger, törst, socialt behov och energi mätarna. Förekommer under dagens närvaro då agenten inte innehar ett vapen för att kunna jaga älg i skogen som alternativ inkomst källa. Detta varieras dock beroende på behovet av inkomst. Vid hög inkomst kan agenterna välja att jaga istället, då detta är en mindre pålitlig inkomst källa men ger en viss variation i vardagen.
- **State_WorkHunt:** Används för att potentiellt öka agenternas nuvarande summa av pengar beroende på om en älg kommer i sikte samt om skottet träffar. Dessutom sänks, hunger, törst, socialt behov och energi under jaktens gång. I slutet av jakten betalas agenten med 250kr per älg. Detta är dock ekonomiskt riskabelt då det inte är garanterat att en älg kommer under dagen eller att agenten missar sitt skott till och med.
- **State_Socialize:** Används för att öka värdet på agenternas sociala behov. Ju lägre denna är ju ensammare känner sig agenten. När den 0 eller går under detta kommer agenten att avlida. För att socialisera sig måste en agent boka ett möte med hjälp av **Telecom.py**, (förklarning av **Telecom.py** finns längre ner i rapporten). När agenterna väl träffas så ökar deras sociala behov mätare. Beroende på plats för mötet så kan även andra värden ökas, exempelvis hunger om mötet sker på restaurangen. Det finns dock risker så som att agenterna måste äta, sova eller dricka just innan mötet, eller helt enkelt är döda. Dessa gör att mötet blir annullerat.

- **State_Idle:** Används för att hålla agenterna i ett sorts viloläge. Detta förekommer exempelvis när agenterna är hemma och inte har bristande ekonomi eller är nöjda både med hunger, törst, socialt behov och energi. Dessutom när agenterna väntar på att medparten anländer till den bestämda mötes platsen.
- **State_WalkTo:** Används för att förflytta agenterna mellan de olika platserna i världen. Under färden minskar även energi, hunger och törst. Agenterna förflyttas samma avstpnd som dess gång hastighte per uppdatering. Vilket innebär att en resa på 8m vid 2 i gång hastighet leder till 4 uppdateringars innan agenten nått sin destination. När agenten har anlänt ändras dess nuvarande plats till målet. Vissa States kräver att agenteerna befinner sig på vissa ställen för att genomföras, exempelvisn **State_Eat** kräver att agenterna befinner sig på restaurangen.
- **State_Die:** Används för att terminera en agent och informera de andra vid dödsfallet. Ett meddelande skrivs ut som förklarar vad agenten dog utav. Om ett möte är bokat och agenten dör, kommer mötet att annulleras.

Telecom.py är där alla meddelanden processeras för att skickas eller bli mottagna. **Telecom** klassen har också ett register över alla agenter som är vid liv och möjliga mötesplatser i simuleringen. Agenterna kan nå **Telecom** klassen genom sin referens kallad “Phone”. Detta simulerar att agenterna skickar SMS mellan varandra för kommunikation. Ett meddelande har variablerna **Avsändare, Mottagare, Klass, Innehåll, Mötestid, Plats, Tidpunkt för leverans** samt **Extra information.** Beroende på vilken klass ett meddelande har så behövs ej alla variabler ha ett värde. Exempelvis dödsnotis för en agent behöver enbart Avsändare, Mottagare, Klass och Innehåll, detta då meddelandet skickas instant och inte planerar något. De meddelande klasser som finns är följande:

- **MSG_MeetUp:** Meddelande klass för att boka ett möte med en agent. Detta meddelande skickas genom att en agent med lågt värde av socialt behov skriver en plats och tid till en slumpmässig agent. Meddelandet kan ej skickas till sig självt. När meddelandet skickats

kommer mottagande agent att kontrollera om denna redan har ett bokat möte eller inte känner för det beroende på energi, hunger eller törst. En agent kan enbart boka ett möte åt gången.

- **MSG_AcceptMeeting:** Meddelande klass för att acceptera mötes förfrågan. Detta meddelande skickas om agent som mottagit **MSG_MeetUp** har möjlighet att träffas. Meddelandet skickas till frågande agent. När meddelandet har skickats eller tagits emot så skapar agenterna som det berör ett meddelande om att de ska bege sig mot mötesplatsen en stund innan avlagd mötestid.
- **MSG_DeclineMeeting:** Meddelande klass för att neka mötes förfrågan. Detta meddelande skickas om agent som mottagit **MSG_MeetUP** inte har möjlighet att träffas. Meddelandet skickas till frågande agent och mötet annulleras.
- **MSG_GotoMeeting:** Meddelande klass för att aktivera **State_WalkTo** för de agenter som bokat möte. Meddelandet skickas till båda parter som tidigare fått förfrågan eller accepterat mötes bokning, men enbart när det är 20 minuter kvar till avlagd mötestid. Är de involverade agenterna avlids slängs meddelandet vid leverans.
- **MSG_AbortMeeting:** Meddelande klass för att avbryta bokat möte. Meddelandet skickas från den agent som inte kan komma till mötet efter att denna har mottagit ett validerings meddelande och mötet annulleras. Validerings meddelande beskrivs längre ner.
- **MSG_ValidateMeeting:** Meddelande klass för att validera agenternas möjlighet att delta i kommande möten. Meddelandet skickas 30 minuter innan avlagd mötestid. Kan agenterna inte delta i mötet skapas ett meddelande av klass **MSG_AbortMeeting** och skickas till den agent som de skall mötas med och mötet annulleras. Är de involverade agenterna avlids slängs meddelandet vid leverans.
- **MSG_Grief:** Meddelande klass för att berätta för återstående agenter att en agent har avlidit. Meddelandet säger åt agenterna att skriva ett sorgset meddelande samt kontrollera om den avlidne agenten var delaktig i deras framtida möte.

Alla meddelanden som skickas och har ett värde i variabeln **Tidpunkt för leverans** som inte är None, plaseras i en leverans kö där de väntar på att skickas tills att simuleringens klocka stämmer med meddelandets t**idpunkt för leverans.** När ett meddelande har skickats så raderas det ur leverans kön.

Lösningens begränsningar

De begränsningar som finns i detta projekt är som följande:

- Projektet är textbaserat och kan därmed inte ge en mer nogran och visuell representation i form av 2D eller 3D. 2D representation skulle vara möjligt vid användandet av ASCII tecken för att representera agenterna och världen.
- Antalet agenter är limiterade till ett minimalt värde av 1 samt ett maximalt värde av 8 simultalna agenter. Detta är enbart ett design val.
- States som i verkliga livet skulle kunna utföras på olika platser är ej möjliga i simulatiuonen. Exempelvis att agenterna kan enbart äta på restaurangen, drick på puben och sova i hemmet.
- Agenterna kan enbart boka ett möte åt gången, möte kan återigen bokas efter att möte har blivit annullerat eller de mötande agenterna har sagt farväl och skiljts åt efter sociala interaktioner.

Diskussion

Enligt mina egna åsikter så är detta projekt roligt, intressant och planerings krävande. Balanserings delen av mitt projektet är inte det bästa, men det fungerar gott och väl för att variera resultaten som slutgiltigt uppstår. Det som skulle kunna förbättras är hur agenterna prioriterar sina behov och integrera dem i en 3D miljö. Den svåraste delen av projektet tyckte jag var att balansera värdena så att agenterna inte instant dör men ändå inte lever för alltid. Det skulle vara intressant att se detta i en miljö då exempelvis en spelare skulle kunna blockera agenterna från att uppnå sina mål genom att förflytta objekt eller punkter som krävs för att slutföra objektivet. Exempelvis blockera vägen för att komma till destinationen eller att förstöra maten så den inte går att förtära.

I framtiden skulle jag vilja testa att göra en AI som försöker smälta in bland andra AIs som enbart går omkring i exempelvis en stad. Därefter ska den försöka att fånga en specifik AI utan att bli upptäckt genom att strategiskt gömma sig och närma sig målet. I princip som “Rött ljus, grönt ljus”.

Testkörningar

Vid de tester som genomförst har jag exprimenterat med olika mängder av agenter, från 1 – 8, samt varierat variablerna som sätts vid start och ökar eller minskar i de olika stadierna. Vid ett tidigare tillfälle innehöll **Humanoid** en variabel kallad **Risk_Of_Sickness**, vilket kunde göra att agenterna förlorar extra mycket energi, hunger, socialt behov samt törst beroende på State och nivå av sjukdom. Dock var det alldeles för obalanserat och ledde till att om en agent blev sjuk så dog den med 99% sannolikhet under samma dag. Efter borttagandet av **Risk_Of_Sickness** så varierar den beräknade livstiden för agenterna mellan 10 dagar till flera månader. Det visar sig efter mina testkörningar att majoriteten av gångerna så dör agenterna av ensamhet då timing för avbokade möten och nästa gång det får en förfrågan är alldeles för stora i intervall. Det leder till att det sociala behovet sjunker till 0 eller under. För övrigt är de övriga värderna alltid justerade efter behov så som hunger, energi och törst, så pass att agenterna klarar sig fint så länge de har ekonomisk inkomst

Referenser

- “Programming Game AI by Example” av Mat Buckland, PDF-format